MODELO PARA LA SOLUCIÓN AL PROBLEMA DE DISEÑO DE TERRITORIOS COMERCIALES, MEDIANTE LA METAHEURÍSTICA DE OPTIMIZACIÓN EVOLUTIVA DE ENJAMBRE DE PARTÍCULAS (EPSO)

SILVIA JULIANA ARIZA GARCÍA
EMILY YOSELIN CARVAJAL CALDERÓN

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2014

MODELO PARA LA SOLUCIÓN AL PROBLEMA DE DISEÑO DE TERRITORIOS COMERCIALES, MEDIANTE LA METAHEURÍSTICA DE OPTIMIZACIÓN EVOLUTIVA DE ENJAMBRE DE PARTÍCULAS (EPSO)

SILVIA JULIANA ARIZA GARCÍA
EMILY YOSELIN CARVAJAL CALDERÓN

Trabajo de grado para optar al título de
INGENIERO(A) INDUSTRIAL

Director
Ph.D. HENRY LAMOS DÍAZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO-MECÁNICAS
ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES
BUCARAMANGA
2014

DEDICATORIA

A Dios primeramente le dedico mi Proyecto, mi Carrera y mi Vida, gracias a Él he llegado hasta este punto, su mano que me ha sostenido y me ha regalado la sabiduría necesaria para lograr mis metas.

A mi padre Hernando, en memoria suya ya mi madre Yolanda, su apoyo fue el pilar para recorrer este camino, aun cuando las fuerzas se menguaban siempre hubieron palabras de fortaleza. Los amo infinitamente.

A mi hermano Andrés Felipe que lo amo y agradezco su paciencia para conmigo por confiar en mí y estar siempre a mi lado regalándome felicidad.

Mil gracias doy a Dios por la maravillosa familia que me regalo, cada uno de ellos tienen parte importante en la culminación de este sueño.

A cada una de las personas que hacen parte de mi vida, sin ellos tampoco hubiera sido posible.

Porque de él, y por él, y para él, son todas las cosas. A él sea la gloria por los siglos. Amén.
Romanos 11:36

Silvia Juliana Ariza García

DEDICATORIA

A Dios por su noble compañía, quién me ha guiado por el buen camino, quien me ha dado fortaleza en los momentos difíciles y con toda la humildad que de mi corazón puede emanar, dedico primeramente mi proyecto a Dios.

A mis padres Javier y Luz Marina, que por su apoyo, sacrificio, amor y comprensión; hicieron todo en la vida para que pudiera lograr mis sueños. Por depositar en mí su confianza al estar lejos de casa.

A mis hermanas Getsy y Hillary por estar siempre presentes. Gracias por su paciencia y por estar en otro momento tan importante en mi vida.

A mi querido Jorge por ser el pilar principal para la culminación de mi carrera, que con su apoyo constante y amor incondicional ha sido amigo y compañero inseparable, fuente de sabiduría, calma y consejo en todo momento.

Gracias a todos los familiares que me ayudaron, a que este gran esfuerzo se convierta en realidad.

Y por último: deseo dedicar este momento tan importante e inolvidable; a mí misma, por no dejarme vencer. Impossible is nothing!

Emily Yóselin Carvajal Calderón

AGRADECIMIENTOS

A Dios, por la vida y la sabiduría para lograr sacar adelante este sueño.

Al Ph.D. Henry Lamos, por su apoyo y guía durante el proceso de desarrollo de este proyecto de investigación.

A la Ingeniera Diana G. Ramírez Ríos, por su dedicación y asesorías, que hacen posible la culminación de este proyecto.

Al Grupo de Investigación “OPALO”, por brindarnos la oportunidad de desarrollar este proyecto.

A la escuela de Estudios Industriales y Empresariales y a la Universidad Industrial de Santander, por permitirnos ser parte activa y brindarnos el conocimiento necesario para llevar a finalidad el proyecto.

A nuestras familias y amigos, por su apoyo incondicional.

CONTENIDO

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE TABLAS

Pág.

LISTA DE ANEXOS

Pág.

RESUMEN

TITULO: MODELO PARA LA SOLUCIÓN AL PROBLEMA DE DISEÑO DE TERRITORIOS COMERCIALES, MEDIANTE LA METAHEURÍSTICA DE OPTIMIZACIÓN EVOLUTIVA DE ENJAMBRE DE PARTÍCULAS (EPSO)*

AUTORES: SILVIA JULIANA ARIZA GARCÍA Y EMILY YOSELIN CARVAJAL CALDERÓN**

PALABRAS CLAVES: EPSO, PARTICLE SWARM OPTIMIZATION EVOLUTION, TDP

CONTENIDO:

Este trabajo de investigación consiste en desarrollar una solución al problema de diseño territorial (TDPs), mediante la elaboración de un framework y la construcción de un toolbox en Matlab®, implementando la metaheurística EPSO "Optimización por enjambre de partículas evolutivas". El TDPs consiste en determinar una división de un conjunto de unidades ubicadas en un territorio que cumple con los criterios múltiples como la compactibilidad, la conectividad y el equilibrio en términos de clientes y la demanda del producto, constituyendo un soporte importante que no debe ser ignorado por los responsables de las actividades inherentes a la distribución comercial ya que se convierte en una decisión táctica para la empresa. El problema de diseño de territorios pertenece a la categoría de problemas de optimización combinatoria, y se clasifica como unproblema NP-hard, es decir, no hay ningún algoritmo que garantice encontrar la mejor solución posible (solución óptima) en un tiempo polinomial. La consecuencia de este resultado es que es un problema muy difícil de resolver computacionalmente, ya que el tiempo de ejecución del método de solución crece exponencialmente con el tamaño del problema. Esto representa todo un reto de investigación. Desde la perspectiva práctica, una buena solución es muy importante ya que ayudaría a las empresas a distribuir su área de trabajo de la mejor manera de acuerdo a sus criterios de planeación.

* Proyecto de grado modalidad Investigación

** Facultad Físico Mecánicas. Escuela de Estudios Industriales y Empresariales. Director Henry Lamos

ABSTRACT

TITLE: MODEL FORT HE SOLUTION TO THE PROBLEM OF DESIG TRADE TERRITORIES THROUGH THE META-HEURISTIC EVOLUTIONARY OPTIMIZATION OF PARTICLE SWARM (EPSO)*

AUTHORS: SILVIA JULIANA ARIZA GARCÍA Y EMILY YOSELIN CARVAJAL CALDERÓN**

KEY TERMS: EPSO, PARTICLE SWARM OPTIMIZATION EVOLUTION, TDP

DESCRIPTION:

This work of investigation is to develop a solution to the problem of territorial design (TDPs), through the development of a framework and building a toolbox in Matlab, implementing the meta-heuristic EPSO "Evolutionary Swarm Optimization particles." The TDPs is to determine a division of a set of units located in a territory that meets multiple criteria such as compactness, connectivity and balance in terms of customer and product demand, constituting an important support that should not be ignored by those responsible for the activities related to the commercial distribution as it becomes a tactical decision for the company.The design problem of territory belongs to the category of combinatorial optimization problems, and these are classified as NP-hard that is the problem, there is no algorithm that guarantees to find the best possible solution (optimal solution) in polynomial time. The consequence of this result is that it is a very computationally intractable problem, since the runtime solution method grows exponentially with the problem size. This represents a challenge for research. From a practical perspective, a good solution is very important as it helps businesses to distribute their work area in the best way according to their planning criteria.

* Degree Project-Research modality

** Faculty of Physical-Mechanical Science – School of Industrial and Business Studies – Director Henry Lamos

INTRODUCCIÓN

El diseño de territorios permite a las empresas dedicadas a las ventas de bienes y servicios mejorar la competitividad con el objetivo de optimizar la rentabilidad. En la actualidad las empresas implementan estrategias innovadoras para mejorar su nivel de servicio al cliente. Una actividad importante a considerar es la distribución de territorios comerciales, la cual permite a la organización ajustarse adecuadamente a las necesidades de los clientes, mejorando su cobertura y ayudando a la compañía a mantener su fuerza de ventas, tiempos de viaje y costos bajo control.

El problema de diseños de territorios consiste en determinar una agrupación de manzanas, códigos postales o clientes individuales que se denominan territorios o unidades básicas (UBs), en un área geográfica determinada, en un número fijo de territorios de manera que se cumplan una serie de requerimientos de planificación impuestos por la empresa.

Los territorios satisfacen ciertas características de planeación definidas en la empresa; como la compactibilidad y la conectividad, que significa el equilibrio entre el número de clientes y la demanda. El problema de diseño de territorios pertenece a la categoría de problemas de optimización combinatoria, y se clasifica como un problema NP-hard, es decir, no hay ningún algoritmo que garantice encontrar la mejor solución posible (solución óptima) en un tiempo polinomial. La consecuencia de este resultado es que es un problema difícil de resolver computacionalmente, ya que el tiempo de ejecución del método de solución crece exponencialmente con el tamaño del problema. Esto representa todo un reto de investigación. Desde la perspectiva práctica, una buena solución es muy importante ya que ayudaría a las empresas a distribuir su área de trabajo de la mejor manera de acuerdo a sus criterios de planeación.

En este trabajo de investigación se propone elaborar un framework para la solución del problema de diseño territorial (TDPs) y construir un toolbox en Matlab® para la solución del problema de diseño territorial comercial mediante la metaheurística EPSO “Optimización por enjambre de partículas evolutivas” constituyendo un soporte importante que no debe ser ignorado por los responsables de las actividades inherentes a la distribución comercial ya que se convierte en una decisión táctica para la empresa.

1. REVISIÓN DE LA LITERATURA

Como antecedente al presente trabajo, en la literatura se encuentra el trabajo desarrollado por Hess et al., (1965)[1], Fleischmann y Paraschis (1988)[2], Hojati (1996)[3], Garfinkel y Nemhauser (1970)[4], Mehrotra et al., (1998)[5], Bozkaya et al., (2003)[6] y Kalcsics et al., (2005)[7]. En la mayoría de estos trabajos, los autores consideran los modelos de un solo objetivo. Entre las pocas obras que se ocupan de los problemas de distritos multi-objetivo se encuentra Bowerman et al., (1995)[8], Scott et al., (1996)[9], Guo et al., (2000)[10], Wei y Chai (2004)[11], Tavares-Pereira et al., (2007)[12] y Ricca y Simeone (2008)[13].

[1] HESS, S.W.; WEAVER, J.B.; SIEGFELDT, H.J.;WHELAN, J.N. y ZITLAU, P.A. Nonpartisan political redistricting by computer. Operations Research 13 (6), (1965); p. 998–1006.

[2] FLEISCHMANN, By PARASCHIS, J.N. Solving a large scale districting problem: a case report. Computers & Operations Research 15 (6), (1988); p.521–533.

[3] HOJATI, M. Optimal political districting. Computers & Operations Research 23 (12), 1 1996; p. 147–1161.

[4] GARFINKEL, R.S., NEMHAUSER, G.L. Solving optimal political districting by implicit enumeration techniques. Management Science 16 (8), (1970); p. B495–B508.

[5] MEHROTRA, A.; JOHNSON, E.L. y NEMHAUSER, G.L. An optimization based heuristic for political districting. Management Science 44 (8), (1998); p. 1100–1113.

[6] BOZKAYA, B.;ERKUT, E. y LAPORTE, G. A tabu search heuristic and adaptive memory procedure for political districting. European Journal of Operational.Research 144 (1), (2003); p.12–26.

[7] KALCSICS, J.; NICKEL, S. y SCHRÖDER, M. Towards a unified territorial design approach: applications, algorithms, and GIS integration. Top 13 (1), (2005); p. 1–56.

[8] BOWERMAN, R.; HALL, B. y CALAMAI, P. A multi-objective optimization approach to urban school bus routing: formulation and solution method. Transportation Research Part A 29 (2), (1995); p. 107–123.

[9] SCOTT, M.N.; CROMLEY, R.G. y CROMLEY, E.K. Multi-objective analysis of school district regionalization alternatives in Connecticut.The Professional. Geographer 48 (1), (1996); p. 1–14.

[10] GUO, J.; TRINIDAD, G. y SMITH, N. MOZART: a multi-objective zoning and aggregation tool. In: Proceedings of the Philippine Computing Science Congress. (PCSC), (2000); pp. 197–201.

[11] WEI, B.C. y CHAI, W.Y.A multiobjective hybrid metaheuristic approach for GIS-based spatial zone model. Journal of Mathematical Modelling and Algorithms 3, (2004); p. 245–261.

[12] TAVARES-PEREIRA, F.; FIGUEIRA, J.R.; MOUSSEAU, V. y BERNARD, R. Multiple criteria districting problems.the public transportation network pricing system of the Paris region. Annals of Operations Research 154 (1), (2007); p. 69–92.

[13] RICCA, F. y SIMEONE, B. Local search algorithms for political districting. European Journal of Operational Research 189 (3), (2008); p. 1409–1426.

Bowerman et al. (1995)[14] presenta un enfoque multi-objetivo para la solución de un problema de enrutamiento de un autobús escolar. El Minmax VRP es una variante del clásico VRP en el que el objetivo es minimizar la duración de la ruta más larga. Este modelo ha aparecido en algunas situaciones reales, especialmente en el contexto de transporte escolar en áreas rurales. En este trabajo se trata una variante del VRP con 2 objetivos: minimizar la duración de la ruta más larga (propio del Minmax VRP) y minimizar la distancia total recorrida (propio del VRP). El "trade-off" está entre el nivel de servicio, representado por el primer objetivo: el tiempo máximo que pasa un escolar en el autobús y el costo de las operaciones, representado por el segundo. Se diseña un algoritmo para obtener conjuntos de soluciones no dominadas cercanas a la curva de eficiencia.

Scott et al. (1996)[15] hace un análisis multi-objetivo de distribución escolar en un estudio de caso de Connecticut, EE.UU. Propone un modelo de programación de metas entera mixta, donde la función objetiva es minimizar las disparidades en: matrícula de minorías, la relación alumno-profesor, y la inscripción general. El número de distritos no es fijo y el criterio de contigüidad no está formulado de manera explícita. Los trabajos experimentales utilizando diferentes escenarios de ponderación, revelan que la reducción al mínimo de la distancia está en conflicto con el resto de objetivos de la equidad y la calidad de las oportunidades educativas.

Guo et al. (2000)[16] propone una zonificación multi-objetivo y una herramienta de agregación (MOZART por sus siglas en inglés Multi-Objective Zoning and AggRegation Tool). MOZART es una integración de un motor gráfico de partición con un sistema de información geográfica (SIG) a través de una interfaz gráfica de usuario. Validan el desempeño de MOZART al resolver dos problemas de zonificación de tres áreas del gobierno local en Victoria: Kingston, Bayside, y Glen

[14] BOWERMAN, Op. cit., p.16
[15] SCOTT, Op. cit., p. 16
[16] GUO, Op. cit., p. 16

Eira. La primera parte de su trabajo experimental se realiza tomando en cuenta un solo objetivo de igualdad en el tamaño de la población. En cambio, en la segunda parte de su trabajo experimental, tanto la equidad de la población y la compactibilidad son tratados como funciones objetivos. Reportan un caso con 577 distritos y 20 zonas. La inclusión de la compactibilidad como el segundo objetivo zonificación produce zonas con mejores formas.

Wei y Chai (2004)[17] presentan un enfoque metaheurístico híbrido multi-objetivo para un modelo de zonificación espacial basado en el SIG. Su procedimiento heurístico es una combinación de búsqueda tabú y búsqueda dispersa. Ellos muestran el desempeño procedimiento mediante la resolución de un problema de distribución político con 55 unidades básicas y 3 distritos. La equidad en la población, la compactibilidad y homogeneidad socioeconómica son tratados como objetivos.

Tavares-Pereira et al. (2007)[18] estudia un problema multi-objetivo de distribución de servicio público. Consideran criterios múltiples, como la ubicación de la zona con respecto a la red, la estructura de la movilidad dentro de una zona, la zona correspondiente a las estructuras administrativas, centros de atracción de la zona, la naturaleza social y la naturaleza geográfica. Propone un algoritmo evolutivo con la búsqueda local y lo aplican a un caso real de la región de transporte público de París. Se discuten resultados para el caso bi-objetivo teniendo en cuenta las diferentes combinaciones de criterios.

Ricca y Simeone (2008)[19] abordan un problema político de distribución de múltiples criterios. Tales criterios son la conectividad, la igualdad de la población, la compactibilidad y la conformidad con los límites administrativos. Transforman el modelo multi-objetivo en un modelo de un solo objetivo mediante una combinación

[17] GUO, Op. cit., p. 16
[18] TAVARES-PEREIRA, Op. cit., p. 16
[19] RICCA, Op. cit., p. 16

convexa de tres funciones objetivo (desigualdad, no compactibilidad y no conformidad con los límites administrativos); la conectividad se considera como una restricción. Ellos comparan el comportamiento de cuatro metaheurísticas de búsqueda local: descenso (descent), búsqueda tabú (tabu search), recocido simulado (simulated annealing) y OBA (old barchelor acceptance). La aplicación se realiza sobre una muestra de cinco regiones italianas donde la metaheurística OBA produce los mejores resultados en la mayoría de los casos.

Zhen Ping et al., (2007)[20] emplea técnicas de solución exactas y destacan la modelación del problema de distritos políticos como un problema de partición de grafos de doble ponderación, el cual es formulado como un modelo de programación cuadrática y es resuelto con software de optimización comercial.
Flores Rivas y Ríos Mercado, (2009)[21] resuelven el problema de diseño territorial, mediante un modelo de programación matemática. El problema se resuelve mediante el método de ramificación y acotamiento y realizan además, un análisis de la sensibilidad del modelo a la variación de diversos parámetros.

Solís García et al., (2009)[22] presenta la modelación de un problema de diseño de territorios de atención comercial como un problema lineal entero mixto relajado resuelto mediante el método de ramificación y acotamiento.

Correa et al., (2010)[23] presenta y describe una heurística bi-objetivo de dos etapas para el rediseño de territorios de venta, inspirada por un problema real de una empresa que vende productos en México. La heurística busca reducir al mínimo la

[20] ZHEN PING, L.; RUISHENG, W. y YONG W. A quadratic programming model for political districting problem.Optimization and Systems Biology. Beijing, China: World Publishing Corporation. Lecture Notes in Operations Research 7, (2007); p. 427-435.
[21] FLORES RIVAS, R. y RÍOS MERCADO, R. Estudio computacional sobre un problema de división de territorios comerciales. Ingenierías. XII (42), (2009); p. 41-47.
[22] SOLÍS GARCÍA, N.; RÍOS MERCADO, R. y ÁLVAREZ SOCARRÁS, M., Modelado de sistemas territoriales con programación entera. Ingenierías. XII(44), (2009); p. 7-15.
[23] CORREA, J.G.; RUVALCABA, M.L.; OLIVARES BENÍTEZ E; AGUILAR, J.A. y MACÍAS, J. Biobjective model for redesigning sales territories. En: 15th Annual International Conference on Industrial Engineering Theory, Applications & Practice, México, D.F., october (2010).

distancia total recorrida y la variación del volumen de ventas de cada vendedor con respecto a la situación actual.

1.1 MARCO DE ANTECEDENTES

El problema de Diseño Territorial Comercial no ha sido abordado como tema de investigación en la Escuela de Estudios Industriales y Empresariales de la UIS, por consiguiente, es una oportunidad de explorar los métodos de solución aproximados. En este trabajo se usa la metaheurística EPSO (por sus siglas en inglés Evolutionary Optimization of Particle Swarm).

En primer lugar, se tiene que en febrero de 2011 fue presentado en la Facultad de Ingeniería Físico-Mecánica de la Universidad Industrial de Santander, Escuela de Estudios Industriales y Empresariales el trabajo especial de grado: Aplicación del algoritmo metaheurística optimización evolutiva por enjambre de partículas (EPSO) a la alternativa de asignación de energía eléctrica: minimización de pagos finales por Oscar Javier Muñoz Sarmiento[24], como requisito para optar el título de ingeniero industrial.

La investigación presenta la implementación del algoritmo metaheurístico Optimización Evolutiva de Enjambre de Partículas, como herramienta para la solución al problema de asignación de energía eléctrica, con el fin de minimizar los tiempos de ejecución. Para este trabajo se tomó como base los supuestos de competencia perfecta, demanda de energía inelástica y se despreciaron los servicios complementarios, las restricciones del sistema de transmisión y las pérdidas.

[24] MUÑOZ SARMIENTO, O.; Aplicación del algoritmo metaheurístico optimización evolutiva por enjambre de partículas (EPSO) a la alternativa de asignación de energía eléctrica: minimización de pagos finales. Bucaramanga, Colombia. (2011)

El análisis de los resultados permitió determinar un cambio del sistema de asignación energética tipo BCM al sistema de asignación PCM, en el cual generaría un ahorro en los costos de la energía eléctrica, beneficiando de esta forma a los operadores del mercado, a los distribuidores y por ende al consumidor final.

En segundo lugar se encuentra el artículo realizado en el 2013 por Silvia Galván, Javier Arias y Henry Lamos de la Universidad Industrial de Santander, en el que presenta el framework SIM-EPSO para la "Optimización por simulación basado en EPSO para el problema de ruteo de vehículos con demandas estocásticas"[25] con descarga preventiva para el caso de un solo vehículo, desarrollando la metaheurística híbrida Optimización de Enjambre de Partículas Evolutivo (EPSO) y Simulación Monte Carlo para la evaluación de la función objetivo. Adicionalmente, se usó un diseño experimental con el propósito de determinar el impacto de los parámetros del VRPSD sobre la función objetivo, y se construyó un banco de pruebas con el objetivo de medir la calidad de las soluciones encontradas en el SIM-EPSO, las cuales fueron contrastadas con la versión básica de la metaheurística Optimización de Enjambre de Partículas (PSO). Los resultados computacionales obtenidos evidencian la eficiencia del framework propuesto para encontrar mejores soluciones respecto al PSO en un tiempo computacional competitivo.

En tercer lugar, se encuentra el artículo aprobado en el 2013 por Henry Lamos, Silvia Galván, Ludy González y Camilo Cruz de la Universidad Industrial de Santander, en el que presenta la metaheurística de Optimización de Enjambre de Partículas (PSO) para la solución del problema de ruteo de Vehículos con Entrega

[25] ARIAS, J.; GALVÁN, S. y LAMOS, H. Optimización por simulación basado en EPSO para el problema de ruteo de vehículos con demandas estocásticas. Dyna, 179. Colombia, Mayo (2013).

y Recolección Simultáneas (VRPSPD)[26]. Aplicaron una representación de la solución y un método de decodificación para implementar el PSO al VRPSPD. El método de decodificación inicia transformando una partícula en una lista de prioridades de clientes para entrar a las rutas y en una matriz de prioridades de vehículos para servir cada cliente. Las rutas de los vehículos son construidos con base en la lista de prioridad de clientes y en la matriz de prioridad de vehículos. El algoritmo es válido usando 18 instancias disponibles en la literatura para problemas de 100, 200 y 400 clientes. Del análisis de resultados obtenidos, se puede concluir que el tamaño del problema influye significativamente en la eficiencia del algoritmo para encontrar una buena solución. Para problemas de gran número de clientes, el algoritmo tuvo mayor dificultad en encontrar soluciones que tuvieran menor desviación en su costo, respecto a los valores de referencia.

1.2 PLANTEAMIENTO DEL PROBLEMA

En la actualidad se abordan problemas de toma de decisiones que surgen en el campo del diseño de territorios. A continuación se describe algunos ejemplos de diferentes problemas.

En una empresa distribuidora de bebidas embotelladas se busca que se cumplan una serie de requerimientos de planeación. También se presenta en la actualidad problemas políticos de distribución de múltiples criterios, tales como conectividad, la igualdad de la población, la compactibilidad y la conformidad con los límites administrativos. Problemas de rediseño de territorios de venta que busca reducir al mínimo, la distancia total recorrida y la variación del volumen de ventas de cada vendedor con respecto a la situación actual.

[26] LAMOS, H. GALVÁN, S; GONZÁLEZ, L. y CRUZ, C. Algoritmo PSO-híbrido para solucionar el problema de ruteo de vehículos con entrega y recolección simultáneas. Revista Facultad de Ingeniería, UPTC, Julio – Diciembre, (2013); Vol 22, No 35. pp. 75-90.

Teniendo en cuenta las situaciones presentadas anteriormente, hace útil y necesario el desarrollo de este proyecto. Sin olvidar que el desempeño de la distribución territorial comercial es fundamental en la rentabilidad de la empresa, por tanto desde la perspectiva práctica, una buena solución es muy importante, ya que ayudaría a las empresas a distribuir el área de trabajo de acuerdo a los criterios de planeación.

Además, dado que el problema de diseño territorial comercial es un problema de optimización combinatoria, hace que sea un reto hallar mejores soluciones por medio de nuevos algoritmos para la solución aproximada. Un método que ha sido bastante promisorio para la solución de problemas combinatorios es el algoritmo de optimización Evolutionary Particle Swarm Optimization (EPSO)[27], siendo esta una herramienta revolucionaria, que combina las características de Estrategias Evolutivas y Optimización por Enjambre de Partículas (PSO).

La motivación de desarrollar la presente herramienta para la solución de problemas de diseño territorial comercial deriva del hecho que en competencia con otros algoritmos metaheurísticos, este ha logrado mejores resultados en muchos casos, posicionando el método como una opción para resolver problemas de optimización complejos.

El problema estudiado en este trabajo fue introducido por R.Z. RÍOS-MERCADO, M.A. SALAZAR-AGUILAR y M. CABRERA-RÍOS[28]. En ese trabajo se aborda un problema de toma de decisiones que surge en el campo de diseño territorial como una aplicación de una empresa distribuidora de bebidas embotelladas, en el cual el problema consiste en determinar una agrupación de unidades básicas, dentro

[27] PRINGLES, Rolando; MIRANDA, Vladimiro y GARCES, Francisco. Expansion óptima del sistema de transporte implementando EPSO. VII Latin American Congress on Electricity Generation & Transmission, October 24-27 2007, Paper C074.

[28] RÍOS-MERCADO, R.Z; SALAZAR-AGUILAR, Maria Angelica y CABRERA-RÍOS, M..“New Models for Commercial Territory Design” Published online: 7 January 2011. Springer Science Bussiness Media, LLC 2011.

de un área geográfica objetivo, en un número fijo de territorios de tal manera que se cumplan una serie de requerimientos de planificación.

1.3 DESCRIPCIÓN DEL PROBLEMA

El problema de diseño de territorios se basa en un conjunto de unidades básica (UBs) - manzanas en este caso- y la cercanía entre las manzanas. Cada manzana posee unos nodos j (C^1j; C^2j), los cuales representan los clientes y el volumen en ventas. Hace parte fundamental del problema el uso de una distancia Euclidiana, *dij*, esta se calcula para cada UBs *i* y *j*. Las UBs deben ser agrupadas en *p* territorios de manera tal que cada nodo pertenezca sólo a un territorio. Esta empresa desea dividir las Unidades Básicas de la mejor manera, teniendo en cuenta criterios tales como económicos y geográficos, logrando así una distribución equitativa en las fuerzas de ventas.

Una restricción significativa es la de conexidad, es decir, entre las UBs debe existir una ruta entre ellas y estas deben pertenecer al mismo territorio. De igual forma, se debe lograr la compactibilidad entre las UBs del mismo territorio, es decir, se deben encontrar tan cerca una de la otra como sea posible.

Para esto en necesario lograr minimizar la medida de dispersión, maximizando la compactibilidad entre las unidades básicas. En este trabajo estudia dos medidas, una basada en el objetivo del problema de p-centro (pCP) y la otra basada en el objetivo del problema p-mediana (pMP). Esto conduce a dos modelos diferentes que se describen a continuación.

1.3.1 Modelos de Optimización. Los modelos que se estudian en este trabajo, fueron introducidos por[29]

Para presentar el modelo, se define primero al conjunto N^i, para todoi en *V* como el conjunto de todas las UBs adyacentes a la UB ^{i}, es decir,

$$N^i = \{j \in V : (i,j) \in E \vee (j,i) \in E\}$$

Las variables de decisión se definen como:

$$x_{ij} = \begin{cases} 1 \text{ si la UB } i \text{ es asignada al territorio con centro en } j \\ 0 \text{ en otro caso} \end{cases}$$

Note que $x_{ii} = 1$ implica que la UB i es un centro territorial. El modelo matemático MTDP (TDP basado en p-mediana) se define a continuación.

(MTDP)

$$\text{minimizar} z = \sum_{j \in V} \sum_{i \in V} d_{ij}\, x_{ij} \qquad (1)$$

$$\text{sujeta a} \sum_{i \in V} x_{ii} = p \qquad (2)$$

$$\sum_{i \in V} x_{ij} = 1 \quad j \in V \qquad (3)$$

$$\sum_{j \in V} w_j^a\, x_{ij} \geq (1 - \tau^a)\mu^a x_{ii} \quad i \in V, a \in A \qquad (4)$$

$$\sum_{j \in V} w_j^a\, x_{ij} \leq (1 - \tau^a)\mu^a x_{ii} \quad i \in V, a \in A \qquad (5)$$

[29] R.Z. RÍOS-MERCADO, M.A. SALAZAR-AGUILAR y M. CABRERA-RÍOS en el artículo de investigación “New Models for Commercial Territory Design” Published online: 7 January 2011. Springer Science Bussiness Media, LLC 2011.

$$\sum_{j \in \cup_{v \in S} \frac{N^v}{S}} x_{ij} - \sum_{j \in S} x_{ij} \geq 1 - |S|$$

$$i \in V, S \subset \frac{V}{N^i \cup \{i\}} \quad (6)$$

$$x_{ij} \in \{0,1\} \quad i,j \in V \quad (7)$$

El objetivo (1) representa una medida de dispersión basada en el objetivo del *p*MP. En este sentido, minimizar dispersión es equivalente a maximizar compactibilidad. La restricción (2) garantiza la creación de exactamente p territorios. Las restricciones (3) representan la asignación exclusiva de las UBs.

Las restricciones (4)-(5) representan el balance con respecto a cada actividad y establecen que el tamaño de cada territorio debe estar dentro de un rango de variación (determinado por τ^a) con respecto al tamaño promedio. La conexidad de los territorios está dada por las restricciones (6). Estas últimas son similares a las restricciones de eliminación de subrutas (subtours) en el problema del agente viajero.

La cantidad de estas restricciones es un número exponencial por lo cual escribirlas explícitamente resulta prácticamente imposible. El método de solución propuesto genera de manera iterativa las restricciones de conexidad necesarias para encontrar una solución óptima del problema. Este modelo puede ser visto como el problema *p*MP con múltiples restricciones de capacidad y restricciones adicionales (4)-(6).

Cuando la medida de dispersión utilizada es el objetivo del *p*CP, la función objetivo (1) es reemplazada por la función (8). El modelo resultante es llamado CTDP y fue introducido en[30].

$$z = max_{i,j \in V}\{d_{ij}x_{ij}\} \qquad (8)$$

Estos modelos pertenecen a la clase NP-hard. Pruebas de complejidad para problemas similares en el contexto de distritos políticos pueden encontrarse en Altman[31].

Estos modelos pertenecen a la clase NP-duro. Cabe señalar que aunque en teoría, las restricciones de conexidad pudieran ser escritas explícitamente, esto no tendría ningún sentido práctico debido a su número exponencial.

Denominemos como R_MTDP al modelo relajado que se obtiene de relajar las restricciones (6) del MTDP. De manera similar definimos el modelo relajado R_CTDP como el modelo resultante al eliminar (6) en CTDP.

Se introduce nuevas formulaciones matemáticas del problema utilizando optimización cuadrática entera (IQP). El número de variables se redujo de n^2 a $2np$. En el modelo cuadrático, hacemos uso de los mismos parámetros utilizados en el modelo lineal. Se define un conjunto adicional $Q = \{1,2,\dots,p\}$ de índices de territorios y un conjunto de variables binarias y_{ip} para identificar los centros de los territorios y z_{jq} para representar la asignación de UBs a territorios. Las variables de decisión para el modelo IQP se definen así:

[30] RÍOS-MERCADO, R.Z.Y FERNANDEZ, E.A reactive GRASP for a commercial territory design problem with multiple balancing requirements.Computers & Operations Research, 36(3), (2009); p. 755–776.

[31] ALTMAN, M. Districting Principles and Democratic Representation. Disertación doctoral, California Institute of Technology, Pasadena, EUA, (1998).

$$Z_{jq} = \begin{cases} 1 & si\ la\ UB\ j\ es\ asignada\ al\ territorio\ q \\ 0 & en\ otro\ caso \end{cases}$$

$$Y_{iq} = \begin{cases} 1 & si\ la\ UB\ j\ es\ asignada\ al\ territorio\ q \\ 0 & en\ otro\ caso \end{cases}$$

De acuerdo con las definiciones anteriores, la equivalencia existente entre las variables utilizadas en el modelo lineal y las utilizadas en el cuadrático está dada por:

$$X_{ij} = \sum_{q \in Q} Z_{jq} Y_{iq} \qquad (9)$$

El modelo QMTDP (*quadratic median-based territory design problema*) usa una medida de dispersión equivalente a la utilizada en el modelo MTDP. A continuación se muestra la formulación del modelo QMTDP.

(QMTDP)

$$\text{minimizar} z = \sum_{q \in Q} \sum_{j \in V} \sum_{i \in V} d_{ij} Z_{jq} Y_{iq} \qquad (10)$$

$$\text{sujeta a} \sum_{i \in V} Y_{iq} = 1 \qquad q \in Q \qquad (11)$$

$$\sum_{q \in Q} Z_{jq} = 1 \qquad j \in V \qquad (12)$$

$$Z_{jq} \geq Y_{jq} \qquad q \in Q, j \in V \qquad (13)$$

$$\sum_{j \in V} w_j^a Z_{jq} \geq (1 - \tau^a)\mu^a \qquad q \in Q, a \in A \qquad (14)$$

$$\sum_{j \in V} w_j^a Z_{jq} \leq (1 + \tau^a)\mu^a \qquad q \in Q, a \in A \qquad (15)$$

$$\sum_{q \in Q} \sum_{j \in U_{v \in S} N^v / S} Z_{jq} Y_{iq} - \sum_{q \in Q} \sum_{j \in S} Z_{jq} Y_{iq} \geq 1 - |S|$$

$$i \in V, S \subset \frac{V}{N^i \cup \{i\}} (16)$$

$$Z_{jq} \in \{0,1\} \qquad q \in Q, j \in V \qquad (17)$$

$$Y_{iq} \in \{0,1\} \qquad q \in Q, j \in V \qquad (18)$$

Las restricciones (11) son usadas para garantizar la asignación de un centro por cada territorio. La asignación exclusiva está dada por las restricciones (12). El balance territorial se establece con las restricciones (14)-(15). Las restricciones (13) indican que una UB j no puede ser centro del territorio q si j no pertenece al territorio q. El último conjunto de restricciones (16) garantizan la conexidad de los territorios. Nuevamente tenemos un número exponencial de estas restricciones.

Bajo esta formulación cuadrática, una medida de dispersión basada en el objetivo *pCP* está dada por (19). Entonces, el QCTDP (*quadratic center-based territory design problema*) es el modelo resultante al reemplazar la función objetivo (10) por la medida de dispersión (19).

$$\min \quad z = max_{i,j \in V} \left\{ \sum_{q \in Q} Z_{jq} Y_{iq} \right\} \qquad (19)$$

Cabe recalcar que estas formulaciones IQP son nuevas en la literatura de diseño de territorios. QMTDP es difícil de resolver debido a que posee un objetivo cuadrático y un conjunto de restricciones cuadráticas (restricciones de conexidad). Adicionalmente, no es posible escribirlas explícitamente debido a su número exponencial. Si las restricciones de conexidad son relajadas, el modelo puede resolverse utilizando cualquier optimizador para problemas MINLP.

Similar a la definición de los modelos relajados R_MTDP y R_CTDP, definimos R_QMTDP como la relajación de QMTDP cuando las restricciones (16) son removidas del modelo. Claramente, una solución para R_QMTDP brinda una cota inferior para QMTDP. Hay algunos casos especiales para los cuales el modelo

puede ser reforzado, por ejemplo, cuando no hay soluciones factibles que contengan territorios formados por una sola UB. Es decir, cuando cada solución factible tiene territorios con al menos dos unidades básicas asociadas a él, la siguiente es una desigualdad válida para R_QMTDP.

$$\sum_{i \in N} Z_{iq} \geq Z_{jq} \qquad q \in Q, j \in V \qquad (20)$$

Estas desigualdades evitan la creación de subconjuntos no conexos *S* tales que $|S| = 1$. Existen un número polinomial de estos subconjuntos, así que estas desigualdades pueden incorporarse fácilmente al modelo. Note que, para las formulaciones MILP las desigualdades válidas equivalente están dadas por:

$$\sum_{i \in N} X_{il} \geq X_{ij} \qquad i \in V, j \in \frac{V}{(\{i\} \cup N^i)} \, (21)$$

En contraste con las desigualdades dadas en (20) que sólo son válidas cuando permanece la condición de territorios formados por más de una unidad básica, las restricciones (21) son válidas para cualquier instancia. Definamos entonces a R1_QMTDP como la relajación conformada por R_QMTDP más las restricciones de adicionales (20). De manera similar se puede definir modelos relajados para el modelo QCTDP. No referimos a éstos como R_QCTDP y R1_QCTDP, respectivamente. Igualmente, para los modelos MTDP y CTDP, se obtienen nuevos modelos relajados agregando (21) en los modelos relajados R_MTDP y R_CTDP. Los denominamos como R1_MTDP y R1_CTDP, respectivamente.

2. JUSTIFICACIÓN DEL PROYECTO

El modelo TDPs (por sus siglas en inglés, Territory Design Problems), fue formulado en el artículo “Experiences with a sales districting model: Criteria and implementation” publicado en Management Science, 18(4), P41-P54; por los investigadores HESS, S.W. & SAMUELS, S.A. (1971); pioneros al proponer un modelo matemático aplicado al diseño de territorios, denominado GEOLINE. Siendo de utilidad en los diferentes problemas que se presentan en las empresas para el manejo del diseño de territorios comerciales con el objetivo de vender y prestar servicios u otras aplicaciones como: los distritos escolares, territorios para equipamientos sociales, servicios de invierno y recogida de residuos sólidos, territorios de servicio de emergencia y asignación de energía eléctrica.

El problema de diseño de territorios consiste en determinar una partición de un conjunto de unidades básicas ubicadas en un territorio específico, de acuerdo a su logística, marketing y requisitos de planificación. Dado que en el sector de consumo masivo, el desempeño de la distribución comercial es una actividad que impacta directamente en la rentabilidad, es de interés para el sector del comercio contar con herramientas computacionales que le permitan tomar decisiones tácticas. Además, siendo el TDPs un problema de optimización combinatoria es un reto para la comunidad científica construir nuevos algoritmos que ayuden a determinar la solución aproximada del problema en tiempo computacional polinomial. Por eso es necesaria la implementación de metaheurísticas para solucionar problemas de asignación de recursos, constituyendo un soporte importante que no debe ser ignorado por los responsables de las actividades inherentes a la distribución comercial.

3. OBJETIVOS

3.1 OBJETIVO GENERAL

Elaborar un modelo para la solución al problema de diseño de territorios comerciales, mediante la metaheurística de optimización evolutiva de enjambre de partículas (EPSO).

3.2 OBJETIVOS ESPECÍFICOS

- Realizar una revisión de la literatura de los métodos utilizados en la solución a problemas de diseño de territorios.
- Elaborar un documento introductorio sobre EPSO y su aplicación en problemas de optimización discreta y combinatoria, aplicados a los problemas de diseño de territorios (TDPs).
- Elaborar un framework para la solución del problema de diseño territorial (TDPs).
- Construir un toolbox en Matlab® para la solución del TDPs.
- Validar la herramienta desarrollada mediante experimentos numéricos, que permita medir su eficiencia.
- Elaborar un artículo de carácter publicable.

4. MARCO REFERENCIAL

4.1 OPTIMIZACIÓN COMBINATORIA

La optimización combinatoria es una rama de la matemática que conjuga disciplinas como la Investigación de Operaciones, la teoría de algoritmos y la complejidad computacional. Los problemas combinatorios tiene la particularidad que son fáciles de formular pero difíciles de resolver. A continuación se presentan una serie de problemas que dan mejores luces a lo anteriormente discutido:

- Un excursionista quiere llenar su mochila de la forma más eficaz, aunque en ella no le quepan todos los objetos que desearía llevar, decidir cuales llevar es un ejemplo de optimización combinatoria.
- Un escritor trata de colocar en orden las distintas palabras de una sentencia.
- Una empresa necesita ejecutar una serie de tareas; cada persona puede realizar una parte de las tareas, con un costo determinado. El objetivo de la empresa es contratar un conjunto de personas que le resuelvan todas las tareas a mínimo costo.
- Un vendedor desea realizar un viaje, comenzando y acabando en la misma ciudad, visitando un conjunto de lugares con el mínimo consumo de recursos.

Dos clases de elementos son comunes a todas las situaciones propuestas: (i) Un conjunto de objetos, casos, palabras, etc., que se han de colocar en distintas posiciones, (ii) Una familia de lugares en los que se deben colocar dichos objetos.

Cada colocación de objetos en sus lugares, se denomina una configuración. La combinatoria se dedica al estudio de las configuraciones. Un problema de optimización combinatoria es la búsqueda de la mejor configuración. La construcción de una o más funciones de valor sobre el espacio de las configuraciones permite ordenar éstas para saber cuál es la mejor.

Un problema de optimización combinatoria es uni-objetivo cuando sobre el espacio de configuraciones se construye una sola función de valor; y multi-objetivo cuando es necesario más de una. Aunque por la finitud, en teoría, estos problemas no ofrecen dificultas para su resolución, en la práctica, suele faltar tiempo para encontrar la solución; por ello, se necesitan otros métodos para resolver estos problemas; siendo imprescindible para su resolución los computadores.

Este tipo de problemas de optimización exhibe las siguientes características:

a) El objetivo es encontrar el máximo/mínimo de una determinada función objetivo sobre un conjunto finito de soluciones, indicado como S.
b) Existe, sobre las variables de decisión de las que depende la función objetivo, un conjunto de restricciones, en general, que permiten identificar un subconjunto de S, Sf, que contiene las denominadas soluciones factibles. Según el contexto, S o Sf son referidos, alternativamente, como espacio de búsqueda.
c) No se exige ninguna condición o propiedad sobre la función objetivo o sobre la definición del conjunto S.
d) Como S es finito, las variables resultarán, en general, discretas, restringiendo su dominio a una serie finita de valores.
e) El número de elementos en S (Sf) es muy elevado, haciendo impracticable la evaluación de todas sus soluciones para determinar la óptima.

Estas características hacen aún más compleja la búsqueda de una solución, cuando se está frente a una optimización multi-objetivo.

4.2 NP HARD

Los problemas computacionales se pueden dividir en dos conjuntos:

Problemas Tratables, para los cuales existe un algoritmo de complejidad polinomial. Se les conoce también como P (de orden polinomial), siendo el conjunto de problemas para los cuales existe una solución algorítmica determinística de complejidad polinomial.

Problemas Intratables, para los que no se conoce ningún algoritmo de complejidad polinomial. Se les llama NP (de orden no determinístico polinomial), siendo el conjunto de problemas para los que existe una solución no determinística de complejidad polinomial.

Entre estos últimos se pueden encontrar los siguientes problemas:

- Agente Viajero
- Caminos Hamiltoniano. Encontrar un camino en un grafo que pasa una sola vez por cada nodo.
- Caminos Eulerianos. Encontrar un camino en un grafo que pasa una vez por cada arco.

La idea intuitiva de problema “difícil de resolver” queda reflejada en el término científico NP-hard utilizado en el contexto de la complejidad algorítmica. En términos coloquiales se puede decir que un problema de optimización difícil es aquel para el que no podemos garantizar el encontrar la mejor solución posible en un tiempo razonable. La existencia de una gran cantidad y variedad de problemas difíciles, que aparecen en la práctica y que necesitan ser resueltos de forma eficiente, impulsó el desarrollo de procedimientos eficientes para encontrar buenas soluciones aunque no fueran óptimas.

En teoría de la complejidad computacional, la clase de complejidad NP-hard es el conjunto de los problemas de decisión que contiene los problemas H, tales que todo problema L en NP puede ser transformado polinomialmente en H. Esta clase

puede ser descrita como conteniendo los problemas de decisión que son al menos tan difíciles como un problema de NP. Esta afirmación se justifica porque si podemos encontrar un algoritmo A que resuelve uno de los problemas H de NP-hard en tiempo polinómico, entonces es posible construir un algoritmo que trabaje en tiempo polinómico para cualquier problema de NP ejecutando primero la reducción de este problema en H y luego ejecutando el algoritmo A.

4.3 MÉTODOS DE SOLUCIÓN

Con frecuencia se resuelven pequeños problemas de optimización, como el camino más corto de ir de un lugar a otro, en general, éstos son lo suficientemente pequeños y pueden ser resueltos sin recurrir a elementos externos al cerebro, pero cuando estos problemas aumentan hay que recurrir a otras herramientas que ayuden a resolverlos, conforme se hacen más grandes y complejos los problemas el uso de los ordenadores es inevitable para su resolución.

Debido a la gran importancia de los problemas de optimización a lo largo de la historia de la matemática aplicada se han desarrollado múltiples métodos para tratar de resolverlos. Una clasificación muy simple de estos métodos o técnicas de optimización es la siguiente [F. Glover 1986].

- Exactas (o enumerativas, exhaustivas, etc.)
- Aproximadas (Constructivas, Búsqueda local, Metaheurísticas)

4.3.1 Métodos Exactos. Estos métodos garantizan encontrar la solución óptima para cualquier instancia de cualquier problema en un tiempo acotado. El inconveniente de estos métodos es que el tiempo necesario para llevarlos a cabo, aunque acotado, crece exponencialmente con el tamaño del problema. Esto provoca en muchos casos que el tiempo necesario para la resolución del problema sea inabordable (cientos de años).

Entre los métodos exactos se destacan los algoritmos de ramificación y acotación. Este es un algoritmo de propósito general en el que se realiza una sistemática enumeración de las soluciones; subconjuntos de la solución son evaluados respecto a su contribución a la función objetivo, se definen unas cotas inferiores y superiores para cada problema, dependiendo del valor de estas cotas calculado para cada posible subconjunto de solución se decide si se ramifica o no el árbol de soluciones.

Padberg y Rinaldi[32]propusieron un mejoramiento del B&B clásico integrándole el método de corte de planos originando el técnica conocida como Branch and Cut (B&C). Por otro lado, la combinación del algoritmo de generación de columnas con el B&B origina el algoritmo conocido como Branch and Price (B&P).

Bard et. al[33] propusieron un algoritmo de B&C para solucionar el VRPTW, mientras que Christiansen y Lysgaard trató el CVRP con demandas estocásticas (CVRPSD), para solucionar este problema propuso un algoritmo de B&P basado en la descomposición de Dantzing-Wolfe, su metodología consiste en la obtención de un problema maestro como resultado de pequeñas variaciones a la formulación original del problema con el objeto de hacerlo más tratable, sobre el problema maestro se aplica programación lineal, si la solución es entera y todas las restricciones se cumplen con igualdad entonces esta es una solución óptima del problema original, de lo contrario, se ramifica orientado a obtener una solución entera.

[32] PADBERG, M y A. RINALDI, G. “Branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems”.SIAM Review. Vol. 33, (1991); p. 60–100.
[33] BARD, JF et. al. “A branch-and-cut procedure for the vehicle routing problem with time windows”. Transportation Science, Vol. 36, (2002); p. 250–69.

Righini y Salani[34] presentan un algoritmo exacto basado en programación dinámica bidireccional y acotada, con estado de decrecimiento relajación de espacio (DSSR - decremental state space relaxation). Esta relajación se realiza con el ánimo de reducir el número de estados a ser explorados. Se realizaron comparaciones con otros algoritmos, reduciendo el tiempo de cómputo, sin embargo, el tiempo requerido en encontrar la solución óptima es bastante amplio.

R.Z. Ríos-Mercado, M.A. Salazar-Aguilar y M. Cabrera-Ríos[35] una de las contribuciones importantes del trabajo realizado por estos autores, es el desarrollo de un método de optimización exacta que maneja de manera efectiva el numero exponencial de restricciones de conexidad en instancias pequeñas y medianas; se desarrolló un procedimiento de solución exacta (ICGP-TDP) basado en B&B y una estrategia de generación de cortes.

4.3.2 Métodos Aproximados. Debido a la complejidad de un problema optimización combinatoria, entre los que se encuentra el problema DTC (Diseño de Territorio Comercial) diferentes métodos aproximados han sido propuestos, aunque no garantizan encontrar la solución óptima brindan una muy buena solución al problema, estos métodos son conocidos como heurísticos, en el presente trabajo se hace un especial énfasis en estos métodos.

Están recibiendo una atención cada vez mayor para resolver estos problemas por parte de la comunidad internacional a lo largo de los últimos 30 años.

Estos métodos sacrifican la garantía de encontrar el óptimo a cambio de encontrar una buena solución en un tiempo razonable.

---

[34] RIGHINI, G y SALANI, M. “Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming” Computers & Operations Research Vol. 36, (2009); p. 1191 – 1203.

[35] RÍOS-MERCADO, R.Z; SALAZAR-AGUILAR, Maria Angelica y CABRERA-RÍOS, M..“New Models for Commercial Territory Design” Published online: 7 January 2011. Springer Science Bussiness Media, LLC 2011.

Entre los algoritmos no exactos se pueden encontrar tres tipos: los heurísticos constructivos (también llamados voraces), los métodos de búsqueda local (o métodos de seguimiento del gradiente) y las metaheurísticas.

- Heurísticos Constructivos. Suelen ser los métodos más rápidos. Generan una solución partiendo de una vacía a la que se le va añadiendo componentes hasta tener una solución completa, que es el resultado del algoritmo.

Aunque en muchos casos encontrar un heurístico constructivo es relativamente fácil, las soluciones ofrecidas suelen ser de muy baja calidad y encontrar métodos de esta clase que produzcan buenas soluciones es muy difícil ya que dependen mucho del problema, para su planteamiento se debe tener un conocimiento muy extenso del mismo.

Además, en muchos problemas es casi imposible, ya que, por ejemplo, en aquellos con muchas restricciones puede que la mayoría de las soluciones parciales sólo conduzcan a soluciones no factibles. [F. Glover and G. Kochenberger. 2002].

- Métodos De Búsqueda Local. Los métodos de búsqueda local o de seguimiento del gradiente parten de una solución ya completa junto con el uso del concepto de vecindario, recorren parte del espacio de búsqueda hasta encontrar un óptimo local. En esa definición han surgido diferentes conceptos, como el de vecindario y óptimo local que ahora pasamos a definir.

El vecindario de una solución s, que notamos como N(s), es el conjunto de soluciones que se pueden construir a partir de s, aplicando un operador específico de modificación (generalmente denominado movimiento).

Un óptimo local es una solución mejor o igual que cualquier otra solución de su vecindario. Estos métodos, parten de una solución inicial, examinan su vecindario y eligen el mejor vecino continuando el proceso hasta que encuentran un óptimo local. En muchos casos, la exploración completa del vecindario es inabordable y se siguen diversas estrategias, dando lugar a diferentes variaciones del esquema genérico. Según el operador de movimiento elegido, el vecindario cambia y el modo de explorar el espacio de búsqueda también, pudiendo simplificarse o complicarse el proceso de búsqueda.

Finalmente, en los años setenta surgió una nueva clase de algoritmos no exactos, cuya idea básica era combinar diferentes métodos heurísticos a un nivel más alto para conseguir una exploración del espacio de búsqueda de forma eficiente y efectiva. Estas técnicas se han denominado metaheurísticas.

Metaheurísticas (Mh). Este término fue introducido por primera vez por F. Glover 1986. Antes de que este término fuese aceptado completamente por la comunidad científica, estos métodos eran denominados como heurísticos modernos.[C.R. Reeves. 1993]

De las diferentes descripciones de metaheurística que se encuentran en la literatura se pueden destacar ciertas propiedades fundamentales que caracterizan a este tipo de métodos. [C. Blum and A. Roli. 2003]

Las metaheurísticas son estrategias o plantillas generales que guían el proceso de búsqueda.

El objetivo es una exploración del espacio de búsqueda eficiente para encontrar soluciones (casi) óptimas.

Las metaheurísticas son algoritmos no exactos y generalmente son no deterministas.

Pueden incorporar mecanismos para evitar las áreas del espacio de búsqueda no óptimas.

El esquema básico de cualquier metaheurística es general y no depende del problema a resolver.

Las metaheurísticas hacen uso de conocimiento del problema que se trata resolver en forma de heurísticos específicos que son controlados de manera estructurada por una estrategia de más alto nivel.

Las metaheurísticas utilizan funciones de bondad (funciones de “fitness”) para cuantificar el grado de adecuación de una determinada solución.

Resumiendo esos puntos, se puede acordar que una metaheurística es una estrategia de alto nivel que usa diferentes métodos para explorar el espacio de búsqueda.

Por lo tanto es de especial interés el correcto equilibrio (generalmente dinámico) que haya entre diversificación e intensificación.

Algoritmos de Enjambre. Los algoritmos bajo el marco de inteligencia de enjambres, son métodos bioinspirados en el comportamiento de colonias de insectos generalmente, como hormigas, abejas, termitas.

Estos algoritmos replican la sinergia que presentan estos sistemas en los que el comportamiento colaborativo de los integrantes del enjambre les permite desarrollar tareas complejas que exceden la capacidad de un solo individuo.

Las hormigas tienen grandes estructuras sociales que les permiten realizar tareas complejas que exceden la capacidad de una sola hormiga. Una hormiga que se embarca en una ruta para encontrar alimento, deja a su paso un rastro de feromonas que tiende a ser seguido por las siguientes hormigas, es así como se crea un proceso iterativo por el cual la acción individual de una hormiga sirve como estímulo para las acciones de los otros individuos, en síntesis, el comportamiento colectivo de todos los agentes da como resultado una sola entidad, un todo que obtiene mejores resultados que los que lograrían sus elementos trabajando por separado.

Optimización evolutiva de enjambre de partículas (EPSO)[36]. La optimización evolutiva de enjambre de partículas es un novedoso algoritmo de optimización metaheurístico que combina los conceptos de Estrategias Evolutivas y Optimización de Enjambre de Partículas.

Un número significativo de algoritmos han sido desarrollados bajo el concepto de Estrategias Evolutivas. Los algoritmos evolutivos han tenido su inspiración en la biología de evolución de las especies, estos se basan en la selección Darwinística para promover el progreso a través del conocimiento óptimo. Los conceptos de Optimización de Enjambre de Partículas, PSO, se basan en diferentes conceptos de la naturaleza referida al comportamiento de distintos grupos de especies animales. El algoritmo trata de imitar el comportamiento colectivo o social de bandadas de aves, cardúmenes de peces o enjambre de abejas como un conjunto de partículas que evoluciona en el espacio de búsqueda motivado por tres factores: inercia, memoria y cooperación.

El algoritmo de EPSO se basa en un conjunto de partículas que evolucionan en el espacio de búsqueda tratando de encontrar un punto óptimo en el mismo. A

[36] PRINGLES, Rolando; MIRANDA, Vladimiro y GARCES, Francisco. Expansion óptima del sistema de transporte implementando EPSO. VII Latin American Congress on Electricity Generation & Transmission, October 24-27 2007, Paper C074.

diferencia de PSO, la evolución no sólo se ve en el comportamiento de las partículas sino también en los pesos que afectan al movimiento de estas a medida que se avanza en el espacio de búsqueda. Una de las características más importantes es que es un método autoadaptivo, que ajusta automáticamente sus parámetros o comportamiento en respuesta a la manera en que progresa la solución del problema.

La adaptación es típicamente comandada por reglas. En la mayoría de los casos estas reglas son heurísticas y dependen ampliamente del conocimiento ganado por los investigadores en observar cómo un algoritmo se comporta frente a una clase de problema. Un método auto adaptivo requiere que el algoritmo desarrolle por sí mismo un proceso de cambio de su comportamiento, en lugar de obedecer una regla externa.

El mecanismo del algoritmo EPSO se puede describir de la siguiente manera: para una iteración dada existe un conjunto de soluciones o alternativas denominadas partículas. Cada partícula está definida por una posición en el espacio de búsqueda (Xi) y una velocidad (Vi).

En un momento dado, hay al menos una partícula que tiene la mejor posición en el espacio de búsqueda. La población de las partículas reconoce tal posición (bg), entonces las partículas tienden a moverse en esa dirección. Además cada partícula es atraída a su mejor posición anterior (bi).

Las partículas se reproducen y evolucionan a lo largo de un número de generaciones según los siguientes pasos:

REPLICACIÓN: Cada partícula es replicada un número de r veces, dando lugar a nuevas partículas iguales.

MUTACIÓN: Los parámetros estratégicos (Wi) que afectarán al movimiento de las partículas son mutados.

REPRODUCCIÓN: De cada partícula se genera un sucesor según la regla de movimiento de la partícula.

EVALUACIÓN: Cada sucesor será evaluado con una función objetivo.

SELECCIÓN: Por un torneo estocástico u otro proceso de selección, las mejores partículas sobreviven para formar una nueva generación.

La regla de movimiento o reproducción de las partículas es la siguiente: dada una partícula X_i^k, una nueva partícula X_i^{k+1} resulta de

$$X_i^{k+1} = X_i^k + V_i^{k+1}$$
$$V_i^{k+1} = w_{i1}^* V_i + w_{i2}^* (b_i - X_i^k) + w_{i3}^* (b_g - X_i^k) P$$

Dónde:

b_i: Mejor punto encontrado por la partícula i en su vida pasada hasta la generación actual,

b_g: Mejor punto global encontrado por el enjambre de partículas en sus vidas pasadas hasta a generación actual,

X_i^k: Ubicación de la partícula i en la generación k,

V_i^k: Velocidad de la partícula i en la generación k.

W_{i1}: Peso del término inercia,

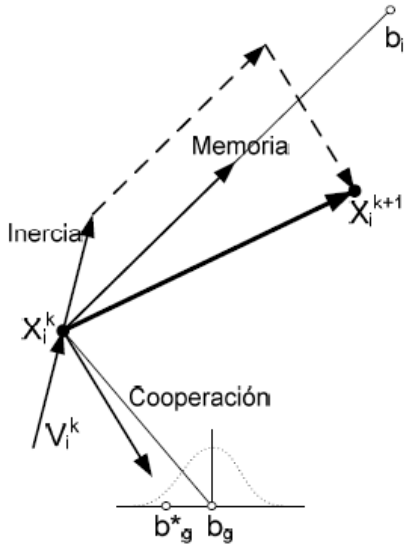
W_{i2}: Peso del término de memoria,

W_{i3}: Peso del término cooperación,

P: Factor de comunicación

La velocidad de la partícula V_i^k; está compuesta por tres términos: El primer término se denomina inercia ya que debido a él la partícula, tiende a moverse en la misma dirección que trae esta partícula, el segundo término se denomina memoria, la partícula es atraída a su mejor posición previa, y el tercer término cooperación, la partícula es atraída por la mejor posición global encontrada por el enjambre. En la figura 1 se ilustra la regla de movimiento de las partículas del algoritmo.

Figura 1. Ilustración de la regla de movimiento de las partículas en EPSO.[37]



En la regla de movimiento, el símbolo * significa que esos parámetros presentarán una evolución producto del proceso de mutación. Esta es una de las diferencias con PSO. La regla de mutación que afecta a los pesos es:

$$w_{ik}^{*} = W_{ik}[\log N(0,1)]^{\tau}$$

[37] PRINGLES, Op. cit., p. 43

Dónde:

$\log N(0,1)$: Es una variable aleatoria con distribución lognormal

τ: Parámetro de aprendizaje, fijado externamente, el cual controla la amplitud de las mutaciones,

En forma aproximada se puede usar como regla de mutación para los pesos:

$$w_{ik}^{*} = W_{ik}[1 + \tau N(0,1)]$$

Dónde:

N (0,1) es una variable aleatoria con distribución Gaussiana, de media cero y varianza 1. Las dos ecuaciones previas son equivalentes siempre que τsea pequeño, de manera que los pesos negativos son descartados.

Además, la mejor solución global b_g es perturbada aleatoriamente como se expresa en la siguiente ecuación:

$$b_g^{*} = b_g + w_{i4}^{*} N(0,1)$$

Donde w_{i4} es el cuarto parámetro estratégico (peso) asociado con la partícula i.

Este controla la amplitud del vecindario de b_g donde es más probable encontrar la mejor solución global o al menos una mejor solución que el b_g actual. El peso w_{i4} es mutado acorde a la regla de mutación general descrita anteriormente.

El factor de comunicación P introduce un sorteo estocástico en la comunicación entre las partículas. Es una matriz diagonal que afecta todas las partículas,

contiene variables binarias de valor 1 con probabilidad p y valor 0 con probabilidad (1-p). El valor de p se fija como parámetro externo y controla el paso de la información dentro del enjambre de partículas, en las formulaciones clásicas se considera p=1. Con este parámetro se controla el flujo libre de información sobre la mejor posición global y permite una búsqueda más local por cada partícula, evitando convergencia prematura a un óptimo local. El valor utilizado en este ejemplo, dando como resultados experimentales[38], la probabilidad de comunicación fue p=0.20, este valor conduce en muchos casos a mejores resultados que un modelo determinista con p=1.

En EPSO tenemos dos mecanismos (evolutivos y auto adaptivo) actuando en secuencia, cada uno con su propia probabilidad de producir no solo mejores individuos, sino también un promedio grupal mejor. La evolución permite que en cada recombinación se induzca un movimiento en dirección al óptimo. Entonces, la selección que actúa sobre una generación que es en promedio mejor que la precedente produce una nueva generación que será mejor que la primera generación de partículas.

Que sea auto adaptivo suma otro interés al método, esta evita en gran medida la necesidad de un ajuste fino de los parámetros iniciales del algoritmo, porque se espera que el procedimiento aprenda (en el sentido evolucionario) las características del espacio de búsqueda y corrija (autoajuste) los pesos en orden a generar una adecuada tasa de progreso hacia el óptimo. Esta característica da robustez al modelo. Esto significa, que independientemente de los valores iniciales, el algoritmo converge al óptimo o un resultado próximo. Una reciente descripción completa de EPSO[39].

[38] HALSE, K. “Modeling and solving complex vehicle routing problems”. PhD thesis, Department of Mathematical Statistics and Operations Research, Technical University of Denmark, (1992).
[39] MIRANDA, V; HRVOJE, Keko y JARAMILLO, Alvaro. “EPSO: Evolutionary Particle Swarms, en Advances in Evolutionary Computing for Systems Design, Serie: Studies in Computational Intelligence, Vol 66, L. Jain, V. Palade, D. Srinivasan, Eds. Springer, (2007); p. 139-168.

5. DISEÑO METODOLÓGICO

5.1 OPTIMIZACIÓN EVOLUTIVA DE ENJAMBRE DE PARTÍCULAS PARA EL DISEÑO TERRITORIAL COMERCIAL

La motivación de desarrollar la presente herramienta para la solución de problemas de diseño territorial comercial deriva del hecho que en competencia con otros algoritmos metaheurísticos, este ha logrado mejores resultados en muchos casos, posicionando el método como una opción para resolver problemas de optimización complejos.

El problema que se estudia se basa en el trabajo de R.Z. RÍOS-MERCADO, M.A. SALAZAR-AGUILAR y M. CABRERA-RÍOS[40]. En el trabajo se aborda un problema de toma de decisiones que surge en el campo de diseño territorial como una aplicación de una empresa distribuidora de bebidas embotelladas, en el cual el problema consiste en determinar una agrupación de unidades básicas, dentro de una área geográfica objetivo, en un número fijo de territorios de tal manera que se cumplan una serie de requerimientos de planificación.

5.1.1 Algoritmo EPSO. En el EPSO cada partícula representa una solución, la cual evolucionara en un espacio de búsqueda para lograr encontrar un punto óptimo dentro de este mismo, cada una de estas partículas tiene asociadas unas velocidades y unos para metros que la llevan siempre hacia la mejor solución.

Es necesario definir la trayectoria, velocidad y movimiento los cuales se pueden definir en términos de las probabilidades de cambio, para que una variable cambie de un estado al otro, es por esto que en el algoritmo binario una partícula se mueve en cada una de sus coordenadas en un espacio restringido de [0 1]

[40] RÍOS-MERCADO, R.Z; SALAZAR-AGUILAR, Maria Angelica y CABRERA-RÍOS, M..“New Models for Commercial Territory Design” Published online: 7 January 2011. Springer Science Bussiness Media, LLC 2011.

dependiendo de su velocidad, donde cada V representa la probabilidad de que una de las variables de decisión de la partícula tome determinado valor. En otras palabras, si para una partícula la velocidad de una de sus variables de decisión es $V = 0{,}20$ entonces hay una probabilidad del veinte por ciento de que esta variabsle tome el valor de 1, es decir, se asigna la UB al depósito y un ochenta por ciento de probabilidad de tomar el valor de 0, no se asigna.

La velocidad en el algoritmo EPSO no está representada como una probabilidad.

La velocidad se rige bajo la siguiente ecuación.

$$V_i^{t+1} = W_{i0}^* v_i^t + W_{i1}^* \left(S_i^t - b_g\right) + W_{i2}^* \left(S_i^t - b_g^*\right)$$

Es por esto que se utiliza la función sigmoide para convertir la expresión de la velocidad en una probabilidad.

$$K_{\left(v_i^k\right)} = \frac{1}{1 + e^{-\left(v_i^t\right)}}$$

Y por último la regla de movimiento de la partícula de una iteración k a una iteración k+1 está dada por la siguiente ecuación:

$$S_i^{t+1} = \begin{cases} 1 \; si \; rand() \leq K_{\left(v_i^t\right)} \\ 0 \; si \; rand() > K_{\left(v_i^t\right)} \end{cases}$$

El algoritmo utilizado para resolver el TDP, se presenta a continuación. En primer lugar se inicializa los parámetros para luego generar los enjambres y se le asigna los pesos de los depósitos para cada unidad básica; de esta manera se genera las partículas para luego su posterior asignación. Consecutivamente se calcula la

distancia total y se actualizan los pesos, la mejor posición, la velocidad y el peso de la velocidad. Se continúa con la siguiente iteración

Sean

$\boldsymbol{n}$: $Número\ de\ UB$

$\boldsymbol{m}$: $Cantidades\ de\ tipos\ de\ productos$

$\boldsymbol{n\_dep}$: $Número\ de\ depósitos$

$\boldsymbol{n\_iter}$: $Número\ de\ iteraciones$

$\boldsymbol{n\_part}$: $Número\ de\ partículas\ por\ iteración$

$\boldsymbol{tao}$: $Grados\ de\ libertad\ del\ aprendizaje$

$\boldsymbol{tol}$: $Tolerancia\ relativa\ con\ respecto\ a\ la\ cantidad\ de\ producto\ a\ manejar$

$\boldsymbol{w_0}$: $Peso\ del\ término\ inercia$

$\boldsymbol{w_1}$: $Peso\ del\ término\ de\ memoria$

$\boldsymbol{w_2}$: $Peso\ del\ término\ de\ cooperación$

$\boldsymbol{w_3}$: $Parámetro\ estratégico\ (peso) asociado\ con\ la\ partícula\ i.$

$\boldsymbol{d_{total}}$: $Suma\ de\ las\ distancias$

$$\boldsymbol{S_{ijqt}} = \begin{cases} 1\ Si\ la\ UB\ es\ asinada\ al\ depósito\ i \\ 0\ de\ lo\ contrario \end{cases}$$

$\boldsymbol{bg}$: $Distancia\ de\ la\ mejor\ posición$

$\boldsymbol{cap_{rest(j)}}$: $Capacidad\ restante\ durante\ la\ asignación\ de\ UB$

$\boldsymbol{cdem(j)}$: $Demanda\ estimada\ por\ UB$

$\boldsymbol{k(i,j)}$: $Peso\ de\ la\ velocidad\ adaptada\ por\ la\ función\ sigmoide$

$\boldsymbol{sbg(i,j)}$: $Posición\ de\ la\ mejor\ solución\ global\ generada\ por\ el\ enjambre$

$\boldsymbol{vinicial(i,j)}$: $Velocidad\ inicial\ de\ las\ partículas$

De acuerdo con estas variables se tiene la siguiente variante del algoritmo EPSO

Tabla 1. Framework EPSO aplicado al Diseño Territorial Comercial (EPSO-TDP)

EPSO-TDP
Inicializar parámetros Elegir *n_dep* depositos y las *n* UB Generación de enjambres Para *t* =1 hasta *n_iter* Asignar *p_dep* para asignacion de UB Generación de partículas Para *q*= 1 hasta *n_part* Asignación de UB: Si k < rand[0,1] Verificar capacidades *s(i,j,q,t)=1* Fin si Calcular distancia total, *dtotal* Actualizar *sb* y *sbg* Actualizar *w0, w1, w2, w3* Generar *sbg** Actualizar *v, k* Fin para Fin para

Para el desarrollo de este algoritmo se construye en Matlab un Toolbox, el cual consiste en encontrar la distancia mínima generada y la asignación final de las unidades básicas en los depósitos.

Para ordenar los depósitos es necesario darle un peso de importancia a cada depósito, en este trabajo el peso de 0.5. Luego por medio de un random asignamos de manera aleatoria, entonces si el random es menor o igual al peso de 0.5 y si ese depósito es igual a cero (es decir que no se ha asignado)

precedemos a asignarlo y guardarlo en orden_dep(i). Finalmente se realiza lo mismo para todos los depósitos.

Para la asignación de las UB, si random es menor o igual a 0.5 y si la UB no ha sido asignada entonces tenemos en cuenta la demanda por medio de la capacidad y la tolerancia dada. Luego de eso se asigna si cumple con el criterio. Finalmente se realiza lo mismo para todas las UB. Recordando que cada UB debe ser asignada a un solo depósito.

TOOLBOX EN MATLAB

Botones empleados:

INICIALIZAR: Permite inicializar parámetros.
GENERAR: Abre las ventanas de las tablas que se deben cargar, en cuanto a la distancia y las capacidades y demandas de las UBs.
CARGAR: Carga los datos a través de tablas que se pueden copiar y pegar de Excel o llenar manualmente.
RESOLVER: Ejecuta el algoritmo EPSO-TDP
LIMPIAR: Permite borrar los resultados previos

INICIO

1. **Inicializar Parámetros**

   n, m, n_iter, n_part, n_dep, tao
   w0, w1, w2, w3
   salida, k, sbg
   dist, sum, v, bg, dem, cap, cap_rest, tol;

2. **Generación de Enjambres**

   for t=1:n_iter

a. Ordenar depósitos

```
cont=0
Mientras cont < n_dep
        i.  for i=1:n_dep
                p_dep(i)=.5;
                Si Rand(0,1); <= p_dep(i) && orden_dep(i)==0
                        cont=cont+1
                        orden_dep(i)=cont
                Fin si
            Fin Para
 Fin Mientras
```

3. Generación de Partículas

```
Para q=1:n_part
Si t==1
    b(q)=sum
Fin
```

4. Asignación UB

```
UB_asig = [1:n]*0
UB_no=[1:n]*0
    a.  Para i = 1:n_dep
        Si Rand(0,1) <= 0.5
                ii.  Para j=1:n
                    UB_no(j)=0
                        Para l=1:m
                            Si ((dem (j,l) <= cap_rest(orden_dep(i),l)*(1+tol))
                            UB_no(j)=1
                            Fin Si
                        Fin Para
                    Si UB_no(j)==0
                        Salida (orden_dep(i),j,q,t) = 1
                        cap_rest(orden_dep(i),l)=cap_rest(orden_dep(i),l)-
                        dem(j,l);
                        UB_asig(j)=1
            Fin si
                        Fin Para
          Fin si
    Fin Para
```

5. Calcular distancia

```
d_total=0
    a.  Para i=1:n_dep
                iii.  Para j=1:n
                            d_total=d_total+salida(i,j,q,t)*dist(i,j)
                      Fin Para
                iv.   Para l=1:m
                            Cap_rest(I,l)=cap(I,l)
                      Fin Para
            Fin Para
```

6. Actualizar valores

```
Si d_total<bg
bg=d_total
    a.  Para i=1:n_dep
                v.   Para j=1:n
                     sbg(i,j)=salida(i,j,q,t)
                     Fin Para
          Fin Para
Fin Si

Si d_total<b(q)
    b(q)=d_total
    Para i=1:n_dep
            Para j=1:n
                    sb(i,j,q)=salida(i,j,q,t)
            Fin Para
    Fin Para
Fin Si
```

7. Actualizar parámetros

```
s=randn(1,4)
w0=w0*(1+tao*s(1,1))
w1=w1*(1+tao*s(1,2))
w2=w2*(1+tao*s(1,3))
w3=w3*(1+tao*s(1,4)
```

a. Alterar mejor solución global

```
Para j=1:n
    vi.  Si Rand(1,1)<=w3
         sbg_alt(i,j)=sbg(i,j)
              Mientras que sbg(i,j)==1
                   sbg_alt(i,j)=0
              Fin Mientras que
    vii.  sbg_alt(i,j)=1
         Fin Si
Fin Para
```

b. Actualizar capacidad restante

```
Para i=1:n_dep
         Para l=1:m
              cap_rest(i,l)=cap(i,l)
         Fin Para
Fin Para
```

c. Actualizar v y k

```
    Si q<n_part
         Para i=1:n_dep
              Para j=1:n
                   v(i,j,q+1)=v(i,j,q)*w0+w1*(salida(i,j,q,t)-sb(i,j,q))+
                   w2*(salida(i,j,q,t)-sbg_alt(i,j))
                   k(i,j)=1/(1+exp(-v(i,j,q+1)))
              Fin Para
         Fin Para
    Mientras que q=1
         Para i=1:n_dep
              Para j=1:n
                   v(i,j,1)=v(i,j,q)*w0+w1*(salida(i,j,q,t)-sb(i,j,q))+
                   w2*(salida(i,j,q,t)-sbg_alt(i,j))
                   k(i,j)=1/(1+exp(-v(i,j,1)))
              Fin Para
         Fin Para
    Fin Mientras que
```

FIN

5.2 RESULTADOS COMPUTACIONALES

El algoritmo EPSO para resolver el problema de Diseño Territorial Comercial fue programado en Matlab versión R2012a y ejecutado en un equipo con procesador Intel Core i3 y 4GB de RAM instalada. Los parámetros asociados al EPSO son:

Tabla 2. Parámetros del EPSO establecidos

Parámetros	**Valor**	
Número de UB	n	11
Cantidad de tipos de productos	m	7
Número de iteraciones	n_iter	5
Número de partículas por iteración	n_part	10
Número de depósitos	n_dep	5
Tao	tao	0,5
Tolerancia	tol	0,5

Ejemplo: Este ejemplo considera un sistema de 11 UB y 5 depósitos. Las distancias dadas entre cada depósito y la UB se presentan en la tabla 3. Cada situación de capacidad y demanda se muestran en las tablas 4 y 5 respectivamente.

Tabla 3. Distancias entre depósitos y UBs

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**CAP**
1	5	6	5	15	19	7	11	13	10	13	10	3100
2	15	12	13	12	6	17	3	6	11	16	5	3100
3	115	5	17	18	12	15	10	5	7	14	6	2800
4	8	10	12	17	6	10	9	14	12	18	13	3200
5	10	11	12	9	7	12	12	7	9	14	9	2530
DEM	942	602	752	568	889	579	656	576	797	731	535	

Tabla 4. Capacidad de productos en depósitos

DEP	1	2	3	4	5	6	7	TOTAL
1	300	650	400	200	350	300	900	3100
2	400	500	300	200	300	400	1000	3100
3	300	450	200	200	400	300	950	2800
4	450	550	350	200	450	400	800	3200
5	280	500	300	200	300	200	750	2530

Tabla 5. Demanda de productos por cada cliente

Clientes	1	2	3	4	5	6	7	TOTAL
1	158	145	20	65	45	59	450	942
2	120	67	25	28	123	39	200	602
3	54	200	35	46	89	98	230	752
4	86	120	45	23	35	59	200	568
5	125	86	55	76	45	72	430	889
6	72	92	75	29	67	94	150	579
7	149	90	45	53	52	67	200	656
8	80	128	30	24	85	49	180	576
9	135	79	35	48	36	104	360	797
10	75	170	55	34	47	100	250	731
11	45	155	25	55	53	52	150	535

Como se hizo mención anteriormente, este ejemplo se corre con el algoritmo EPSO. En la tabla 6. Se observan los resultados obtenidos por algoritmo, la asignación de las unidades básicas a cada depósito se realiza teniendo en cuenta los resultados binarios, cuando el resultado es 1 la UB es asignada al depósito, de igual forma cabe resaltar que para cada UB solo se puede asignar a un depósito.

Tabla 6. Resultados obtenidos por el algoritmo EPSO

bg*	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	1	1	0	1	0	1	0
2	0	0	0	1	0	0	0	0	0	0	0
3	0	0	1	0	0	0	1	0	1	0	0
4	1	0	0	0	0	0	0	0	0	0	1
5	0	1	0	0	0	0	0	0	0	0	0

La asignación de las unidades básicas, teniendo en cuenta los parámetros se realiza de la siguiente manera:

Deposito 1: 5-6-8-10
Deposito 2: 4
Deposito 3: 3-7-9
Deposito 4: 1-11
Deposito 5: 2

Los experimentos numéricos efectuados para evaluar el desempeño del EPSO, se llevaron a cabo aplicando el modelo en un conjunto de instancias desarrolladas para el Diseño de Territorios Comerciales. Se trata de 3 instancias en la cual están conformadas por los siguientes elementos: la matriz de distancia entre los depósitos y las unidades básicas; la capacidad de los 7 tipos de productos en los depósitos; la demanda de los 7 tipos de productos por cada uno de los clientes.

La primera instancia es de 11 unidades básicas y 5 depósitos, la segunda instancia es de 44 unidades básicas y 15 depósitos. Finalmente la última instancia en de 88 unidades básicas y 20 depósitos. Los parámetros para estas 3 instancias están descritos en la tabla 7. De igual forma se detalla los resultados encontrados de las distancias mínimas para diferentes tolerancias.

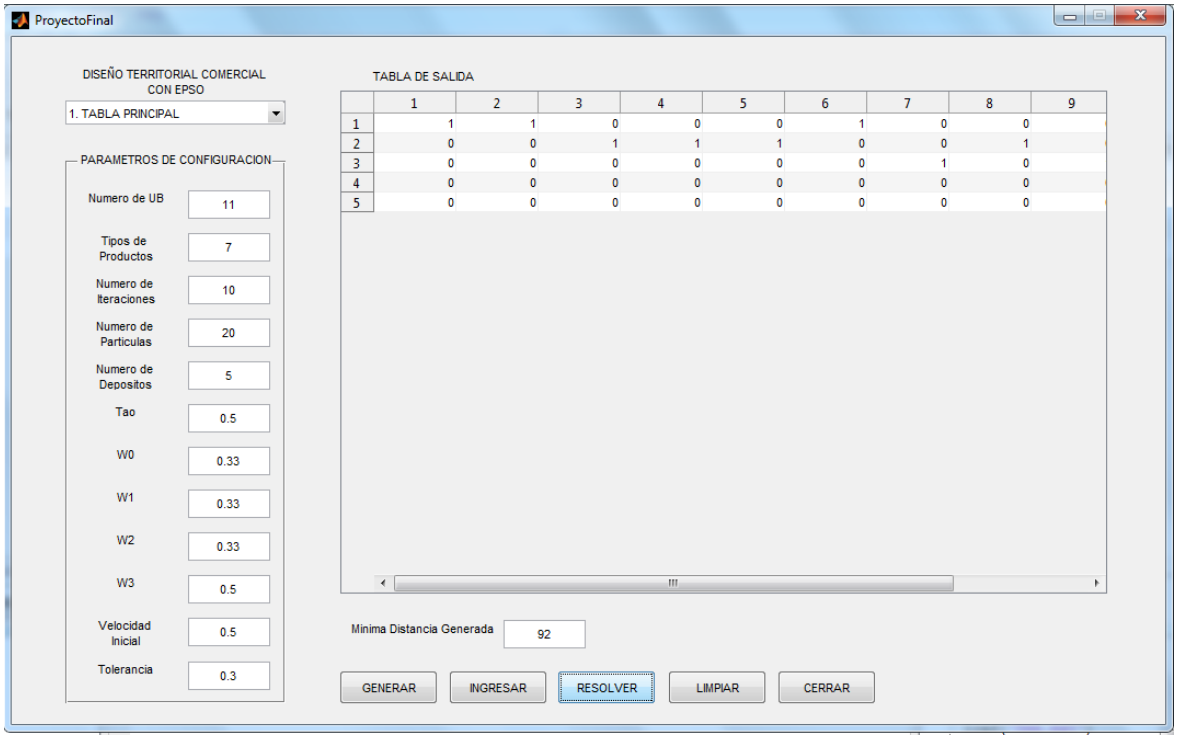
Tabla 7. Parámetros para EPSO con resultados para 3 instancias.

UB		11	44	88
Depósitos		5	15	20
Iteraciones		5	5	5
Partículas		10	10	10
Productos		7	7	7
Tao		0.5	0.5	0.5
W0		0.33	0.33	0.33
W1		0.33	0.33	0.33
W2		0.33	0.33	0.33
W3		0.5	0.5	0.5
Velocidad inicial		0.5	0.5	0.5
		Distancia Min		
Tolerancia	**0.3**	92	421	899
	0.5	95	411	903
	0.9	83	399	903

En los resultados obtenidos para las diferentes instancias se pudo observar que la tolerancia relativa con respecto a la cantidad de productos a manejar es un elemento significativo en el resultado. Sin embargo no se puede generalizar su aumento o disminución, puesto que se refleja resultados diferentes.

Es necesario tener en cuenta que, si el número de depósitos es el ideal para esa cantidad de unidades básicas, es decir, que si la oferta es mayor que la demanda, por lo que en algunas instancias ciertos depósitos no son asignados. Puede ser porque algún depósito se encuentra muy lejos de los demás o que con menos depósitos es suficiente para la asignación de todas las unidades básicas. Se puede observar en la siguiente figura:

Figura 2. Sin asignación de UBs en todos los Depósitos.



Teniendo en cuenta las mismas instancias, se experimenta con 10 iteraciones y 20 partículas, los resultados y los parámetros están descritos en la tabla 8.

Tabla 8. Resultados con 10 iteraciones y 20 partículas

UB		11	44	88
Depósitos		5	15	20
Iteraciones		10	10	10
Partículas		20	20	20
Productos		7	7	7
Tao		0.5	0.5	0.5
W0		0.33	0.33	0.33
W1		0.33	0.33	0.33
W2		0.33	0.33	0.33
W3		0.5	0.5	0.5
Velocidad inicial		0.5	0.5	0.5
		Distancia Min		
Tolerancia	**0.3**	92	393	890
	0.5	93	411	898
	0.9	83	413	873

En las 3 instancias se encuentran mejores resultados en comparación con la tabla 7. Para 88 unidades básicas y 20 depósitos existe una mejor solución de 873; esto quiero decir que al aumentar el espacio de búsqueda las partículas se direccionan a una solución óptima local.

Para un mayor número de iteraciones y partículas se presentó un inconveniente con 88 unidades básicas y 20 depósitos. La memoria del computador no es suficiente para desarrollar el modelo con 100 iteraciones y 200 partículas. Esto refleja una limitante a la hora de aumentar el espacio de búsqueda. Sin embargo para 44 unidades básicas y 15 depósitos se encontró una mejor solución de 377 de distancia. Los resultados están descritos en la tabla 9.

Tabla 9. Resultados con 100 iteraciones y 200 partículas

UB		11	44	88
Depósitos		5	15	20
Iteraciones		100	100	100
Partículas		200	200	200
Productos		7	7	7
Tao		0.5	0.5	0.5
W0		0.33	0.33	0.33
W1		0.33	0.33	0.33
W2		0.33	0.33	0.33
W3		0.5	0.5	0.5
Velocidadinicial		0.5	0.5	0.5
		Distancia Min		
Tolerancia	**0.3**	92	382	-
	0.5	93	386	-
	0.9	83	377	-

Teniendo en cuenta el tiempo de respuesta, Matlab demora alrededor de 15 a 20 segundos para arrojar los resultados con 88 UB y 20 depósitos. Para el resto de pruebas numéricas los resultados son de inmediato.

Finalmente se realiza pruebas numéricas con una velocidad de 0.2 como lo muestra la tabla 10. En esta ocasión no refleja cambio alguno al disminuir la velocidad inicial. Sigue encontrando la misma distancia mínima generada con 10 iteraciones y 20 partículas.

Tabla 10. Cambios en Velocidad

UB		11	44	88
Depósitos		5	15	20
Iteraciones		10	10	10
Partículas		20	20	20
Productos		7	7	7
Tao		0.5	0.5	0.5
W0		0.33	0.33	0.33
W1		0.33	0.33	0.33
W2		0.33	0.33	0.33
W3		0.5	0.5	0.5
Velocidadinicial		0.2	0.2	0.2
		Distancia Min		
Tolerancia	**0.3**	92	393	890
	0.5	93	411	898
	0.9	83	413	873

Se realiza una sensibilidad con la tolerancia donde se puede observar la mínima distancia generada con las diferentes tolerancias. Concluyendo así, que a mayor tolerancia se logra obtener una mínima distancia, lo cual mejora la compactibilidad entre las Unidades Básicas.

Tabla 11. Tolerancia y Distancia Mínima

(n;n_dep)	TOLERANCIA	DISTANCIA MÍNIMA
(11,5)	0	86
(11,5)	0.1	86
(11,5)	0.2	86
(11,5)	0.3	83
(11,5)	0.4	83
(11,5)	0.5	83
(11,5)	0.6	83
(11,5)	0.7	83
(11,5)	0.8	83
(11,5)	0.9	83
(11,5)	1	83

6. CONCLUSIONES

En el presente trabajo se resolvió el problema de Diseños de Territorios Comerciales con la metaheurística EPSO, utilizando un método de decodificación para la asignación de UB a los depósitos. Adicionalmente se efectuaron experimentos numéricos para evaluar el desempeño de EPSO sobre 3 instancias de unidades básicas y depósitos conformado de esta manera (11,5) (44,15) y (88,20).

Es necesario tener en cuenta el número de depósitos para las unidades básicas, ya que si es muy grande la cantidad de depósitos puede que estos no se asignen. Puede darse por el nivel de servicio o por la lejanía de algún depósito con respecto a las demás.

Se presentó inconvenientes con respecto a la memoria del computador para resolver en Matlab® con instancias de 88 unidades básicas y 20 depósitos al aumentar las iteraciones y partículas. Este modelo solo resuelve para instancias pequeñas y con pequeñas iteraciones.

Al aumentar el número de iteraciones, el espacio de búsqueda aumenta y se puede encontrar mejores resultados con respecto a la distancia mínima.

El tiempo de respuesta para pequeñas instancias es casi de inmediato aunque para instancias mayores demora alrededor de 15 a 18 segundos.

7. RECOMENDACIONES Y TRABAJOS FUTUROS

Para futuras investigaciones enfocadas a encontrar método de solución para el Diseño Territorial Comercial se recomienda tener en cuenta estudios sobre la convergencia de las velocidades de las partículas empleadas en las heurísticas y el estudio analítico sobre el comportamiento de sus demás parámetros.

En el campo de las TDPs se está trabajando en la implementación de una vecindad basada en intercambio de unidades básicas entre territorios vecinos para tener una mayor flexibilidad en la fase de pos procesamiento. Otra de las áreas de oportunidad es evaluar la sensitividad del algoritmo con respecto al parámetro de ponderación de la función miope.

Futuros trabajos estarán enfocados en la aplicabilidad de esta metaheurística, logrando demostrar el impacto en el sector, logrando escalabilidad de la herramienta generada y su utilidad para la optimización de la planeación logística como soporte a la toma de decisiones.

BIBLIOGRAFÍA

ALTMAN, M. Districting Principles and Democratic Representation. Dissertation doctoral, California Institute of Technology, Pasadena, EUA, (1998).

BARD, JF et. al. “A branch-and-cut procedure for the vehicle routing problem with time windows”. Transportation Science, Vol. 36, (2002); p. 250–69.

BOWERMAN, R.; HALL, B. y CALAMAI, P. A multi-objective optimization approach to urban school bus routing: formulation and solution method. Transportation Research Part A 29 (2), (1995); p. 107–123.

BOZKAYA, B.;ERKUT, E. y LAPORTE, G. A tabu search heuristic and adaptive memory procedure for political districting. European Journal of Operational. Research 144 (1), (2003); p. 12–26.

CORREA, J.G.; RUVALCABA, M.L.; OLIVARES BENÍTEZ E; AGUILAR, J.A. y MACÍAS, J. Biobjective model for redesigning sales territories.En: 15th Annual International Conference on Industrial Engineering Theory, Applications & Practice, México, D.F., october (2010).

FLEISCHMANN, B y PARASCHIS, J.N., 1988. Solving a large scale districting problem: a case report.Computers & Operations Research 15 (6), 521–533.

FLORES RIVAS, R.; RÍOS MERCADO, R. Estudio computacional sobre un problema de división de territorios comerciales. Ingenierías. XII (42), (2009); p. 41-47.

GARFINKEL, R.S., NEMHAUSER, G.L. Solving optimal political districting by implicit enumeration techniques. Management Science 16 (8), 1970; p. B495–B508.

GUO, J.; TRINIDAD, G. y SMITH, N. MOZART: a multi-objective zoning and aggregation tool. In: Proceedings of the Philippine Computing Science Congress. (PCSC), (2000); pp. 197–201.

HALSE, K. “Modeling and solving complex vehicle routing problems”. PhD thesis, Department of Mathematical Statistics and Operations Research, Technical University of Denmark, (1992).

HESS, S.W.; WEAVER, J.B.; SIEGFELDT, H.J.;WHELAN, J.N. y ZITLAU, P.A., 1965. Nonpartisan political redistricting by computer. Operations Research 13 (6), 998–1006.

HOJATI, M. Optimal political districting. Computers & Operations Research 23 (12), 1 1996; p. 147–1161.

KALCSICS, J.; NICKEL, S. y SCHRÖDER, M. Towards a unified territorial design approach: applications, algorithms, and GIS integration. Top 13 (1), (2005); p. 1–56.

MEHROTRA, A.; JOHNSON, E.L. y NEMHAUSER, G.L. An optimization based heuristic for political districting. Management Science 44 (8), 1998; p. 1100–1113.

MIRANDA, V; HRVOJE, Keko; JARAMILLO, Alvaro.“EPSO: Evolutionary Particle Swarms, en Advances in Evolutionary Computing for Systems Design, Serie: Studies in Computational Intelligence, Vol 66, L. Jain, V. Palade, D. Srinivasan, Eds. Springer, (2007); p. 139-168.

PADBERG, M y A. RINALDI, G. “Branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems”.SIAM Review. Vol. 33, (1991); p. 60–100.

PRINGLES, Rolando; MIRANDA, Vladimiro y GARCES, Francisco. Expansion óptima del sistema de transporte implementando EPSO. VII Latin American Congress on Electricity Generation & Transmission, October 24-27 2007, Paper C074.

RICCA, F. y SIMEONE, B. Local search algorithms for political districting. European Journal of Operational Research 189 (3), (2008); p. 1409–1426.

RIGHINI, G y SALANI, M. “Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming” Computers & Operations Research Vol. 36, (2009); p. 1191 – 1203.

RÍOS-MERCADO, R.Z.Y FERNANDEZ, E.A reactive GRASP for a commercial territory design problem with multiple balancing requirements.Computers & Operations Research, 36(3):755–776, 2009.

RÍOS-MERCADO, R.Z; SALAZAR-AGUILAR, Maria Angelica y CABRERA-RÍOS, M..“New Models for Commercial Territory Design” Published online: 7 January 2011. Springer Science Bussiness Media, LLC 2011.

SCOTT, M.N.; CROMLEY, R.G. y CROMLEY, E.K. Multi-objective analysis of school district regionalization alternatives in Connecticut.The Professional. Geographer 48 (1), (1996); p. 1–14.

SOLÍS GARCÍA, N.; RÍOS MERCADO, R. y ÁLVAREZ SOCARRÁS, M., Modelado de sistemas territoriales con programación entera. Ingenierías.XII(44), (2009); p. 7-15.

TAVARES-PEREIRA, F.; FIGUEIRA, J.R.; MOUSSEAU, V. y BERNARD, R. Multiple criteria districting problems.the public transportation network pricing system of the Paris region. Annals of Operations Research 154 (1), (2007); p. 69–92.

WEI, B.C. y CHAI, W.Y.A multiobjective hybrid metaheuristic approach for GIS-based spatial zone model. Journal of Mathematical Modelling and Algorithms 3, (2004); p. 245–261.

ZHEN PING, L.; RUISHENG, W. y YONG W.A quadratic programming model for political districting problem.Optimization and Systems Biology. Beijing, China: World Publishing Corporation. Lecture Notes in Operations Research 7, (2007); p. 427-435.

ANEXOS

Anexo A. Pasos para el Uso del Software Desarrollado en MATLAB

Pasos para el Uso del Software Desarrollado en MATLAB

Este capítulo pretende dar a conocer el paso a paso para la aplicación del algoritmo EPSO en MATLAB y el desarrollo de la misma.

1. Instalación

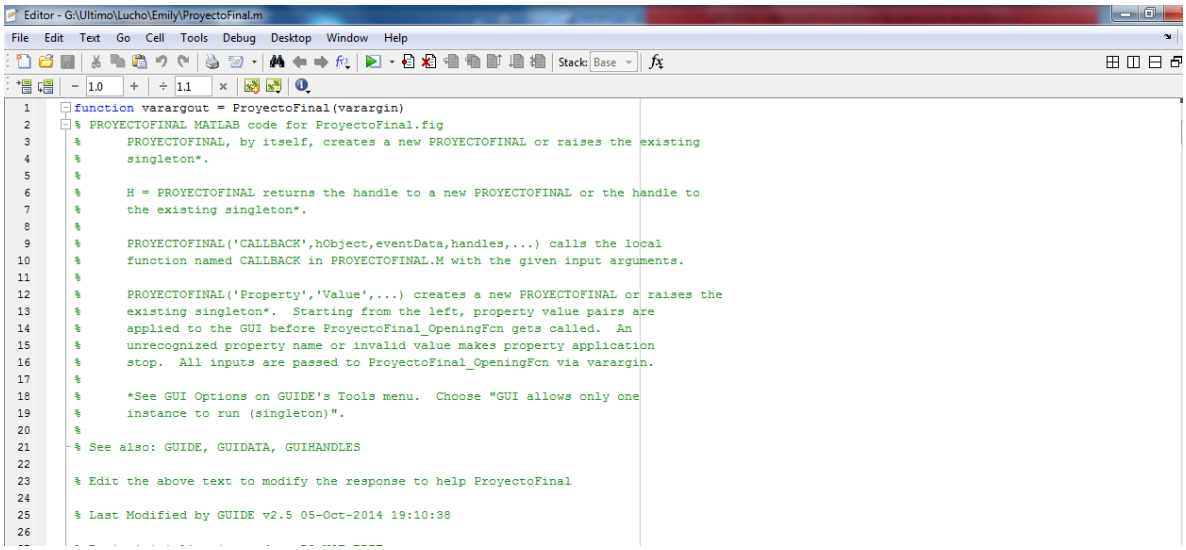
Para poder correr el software es necesario tener instalado previamente MATLAB versión 2012 o superior. Una vez se cumpla con este requisito, se copia la carpeta, “TDP-EPSO” en el destino que el usuario desee. Esta carpeta contiene el entorno amigable y las funciones necesarias para el buen funcionamiento del software.

2. Datos

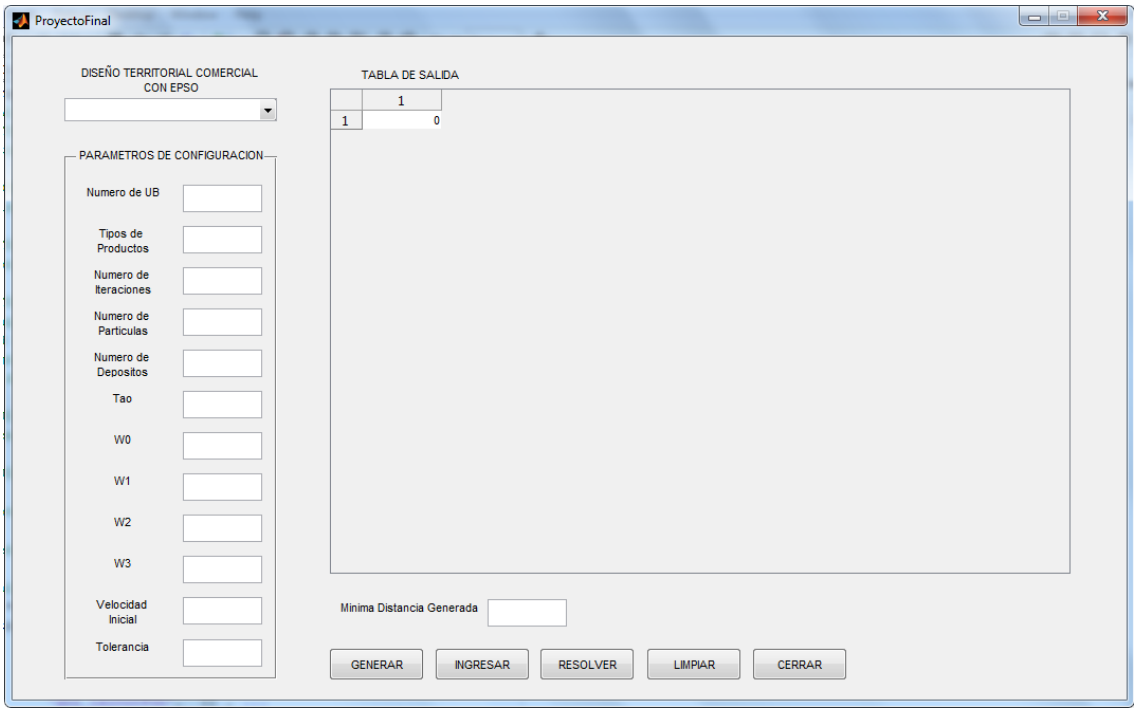
Una vez ubicada la carpeta “TDP-EPSO” en la extensión deseada es necesario abrir el archivo Excel que se encuentre en esta carpeta, el cual contiene por defecto los datos del ejemplo 1 del presente trabajo, en donde es necesario que en la hoja uno se ingresen las distancias dadas entre depósitos y UB. En la hoja dos todos los escenarios de demanda y capacidad. Si el usuario desea aumentar los datos, es posible con solo agregar más filas en el mismo orden en el archivo Excel.

3. Abrir Software:

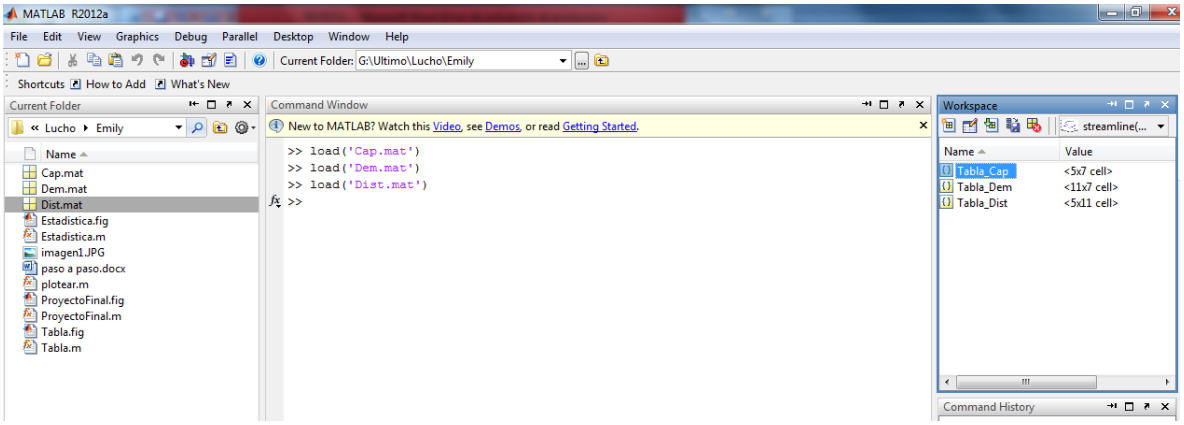
Una vez ingresados los datos nos dirigimos a abrir la aplicación dando doble click en el archivo “ProyectoFinal.m” donde se abrirá la siguiente ventana.



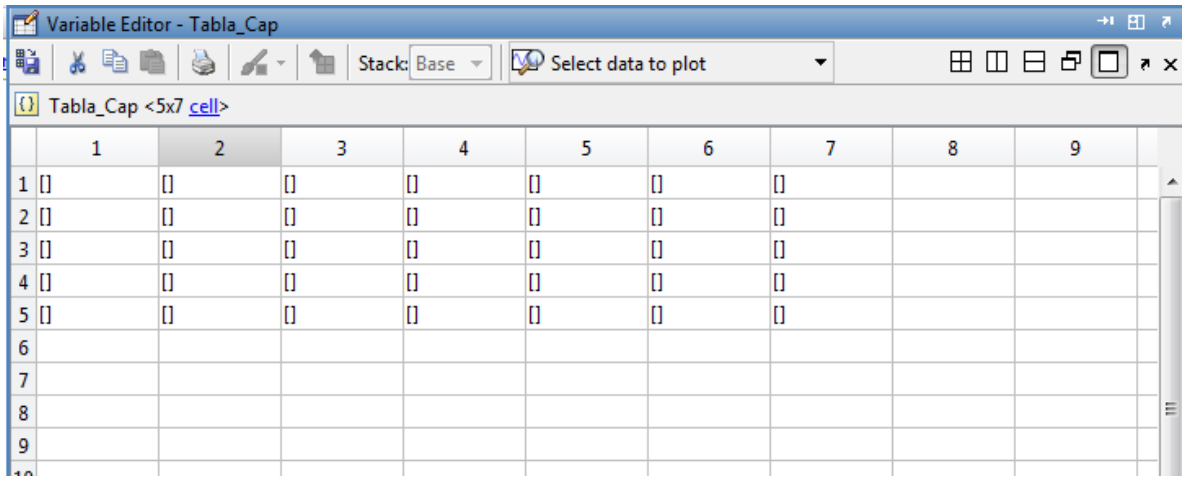
Luego de eso, dando click a botón de triangulo ver RUN para que corra el modelo. Seguidamente aparece una nueva ventana en el que se procede a meter los datos correspondientes.

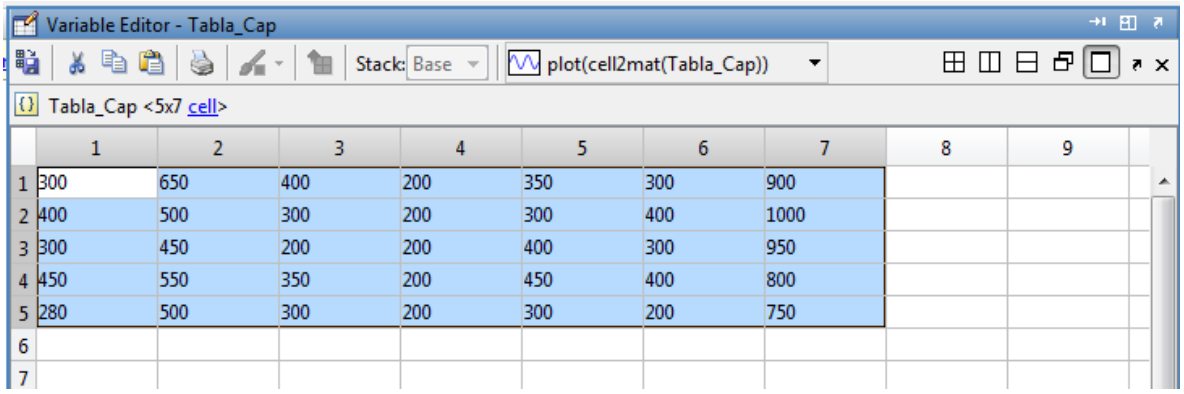


Al ingresar los datos damos click en el botón “GENERAR”. Luego abrimos la ventana principal de matlab y en el lado izquierdo de la pantalla le damos click en “Cap.mat” , “Dem.mat” y “Dist.mat” para que carguen. Luego en el lado derecho de Workspace damos click en “Tabla_Cap” para agregar los datos de capacidad.



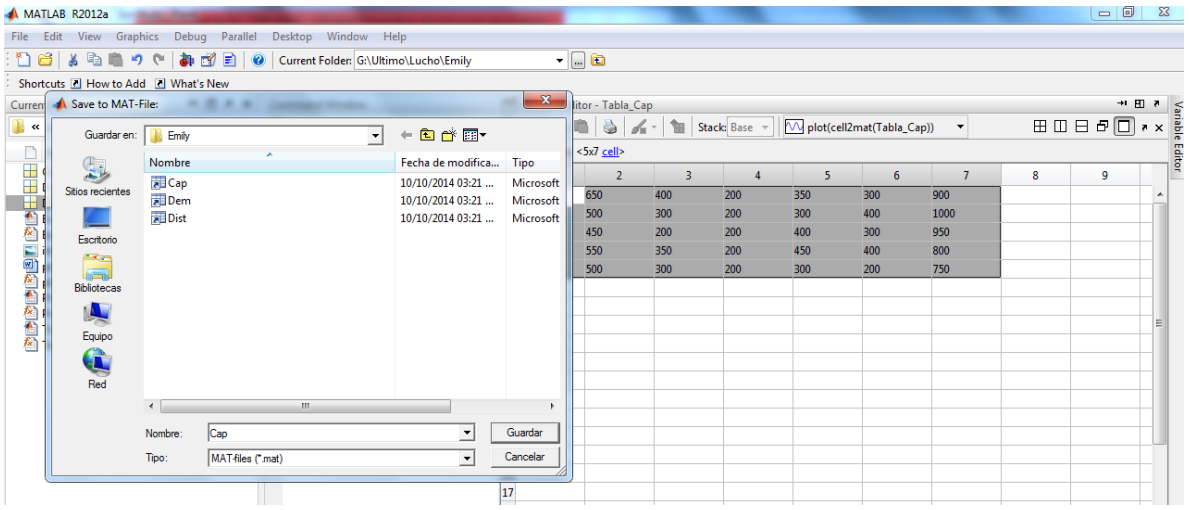
Aparece una tabla con unos corchetes, en ese espacio es donde agregamos los datos, para ello vamos a las tablas de Excel en donde se encuentran los datos, copiamos y luego pegamos en el cuadro de Variable Editor.



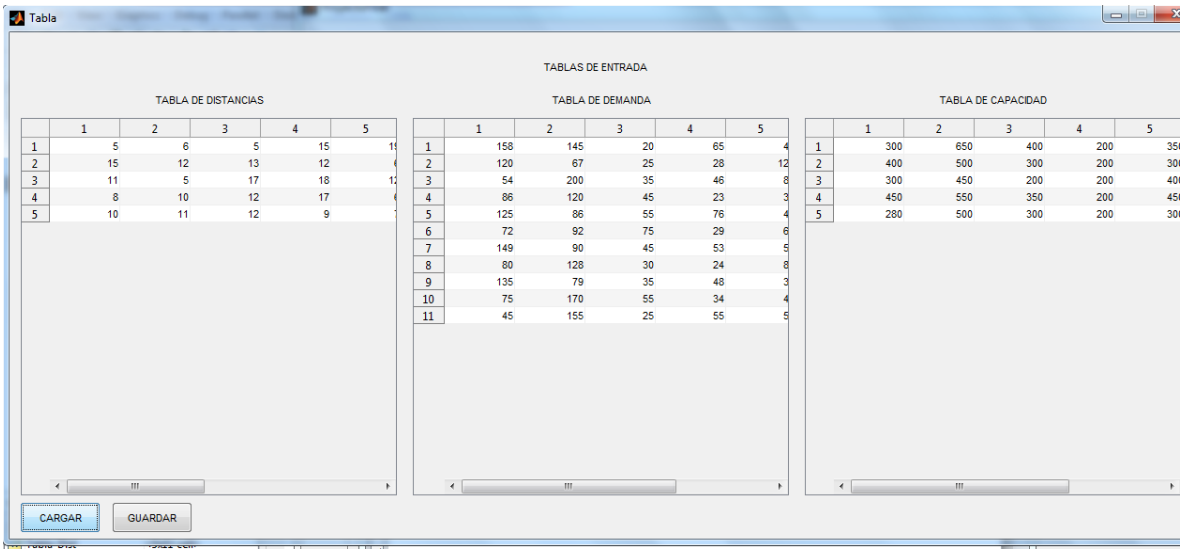


	1	2	3	4	5	6	7	8	9
1	300	650	400	200	350	300	900		
2	400	500	300	200	300	400	1000		
3	300	450	200	200	400	300	950		
4	450	550	350	200	450	400	800		
5	280	500	300	200	300	200	750		
6									
7									

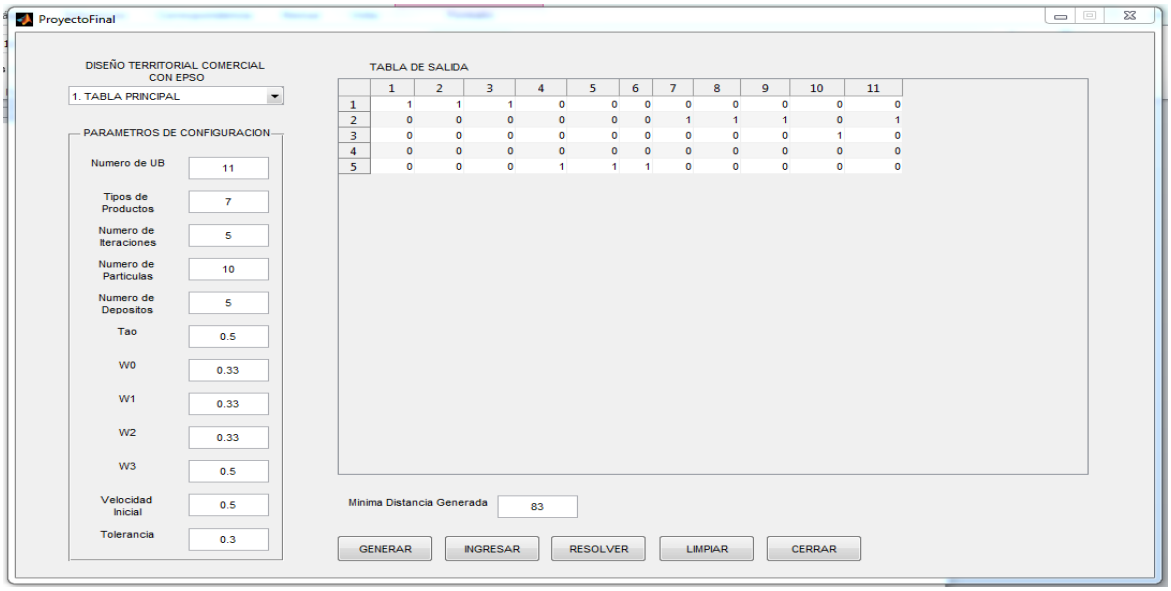
Damos click en el botón “Save” y seleccionamos “Cap” y guardamos. Enseguida pregunta si deseamos reemplazarlo y seleccionamos Si. De ese mismo modo cargamos los datos de la demanda y la distancia.



Nos devolvemos a la ventana “ProyectoFinal” en matlab y damos click en el botón “INGRESAR”. Aparece una nueva ventana con tres tablas. Damos click en la opción “CARGAR” y aparecen todos los datos que teníamos en el archivo de Excel. Luego damos click en “GUARDAR”



Luego en la ventana principal de “ProyectoFinal” le damos click en el botón “RESOLVER” y finalmente aparece la asignación y la distancia mínima.



Anexo B. Implementación en MATLAB del Algoritmo Optimización Evolutiva de Enjambre de Partículas (EPSO) Propuesto.

```
function varargout = ProyectoFinal(varargin)
% PROYECTOFINAL MATLAB code for ProyectoFinal.fig
%       PROYECTOFINAL, by itself, creates a new PROYECTOFINAL or raises
the existing
%      singleton*.
%
%      H = PROYECTOFINAL returns the handle to a new PROYECTOFINAL or the
handle to
%      the existing singleton*.
%
%       PROYECTOFINAL('CALLBACK',hObject,eventData,handles,...) calls the
local
%       function named CALLBACK in PROYECTOFINAL.M with the given input
arguments.
%
%       PROYECTOFINAL('Property','Value',...) creates a new PROYECTOFINAL
or raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%       applied to the GUI before ProyectoFinal_OpeningFcn gets called.
An
%         unrecognized property name or invalid value makes property
application
%         stop.   All inputs are passed to ProyectoFinal_OpeningFcn via
varargin.
%
%       *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help ProyectoFinal

% Last Modified by GUIDE v2.5 05-Oct-2014 19:10:38

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @ProyectoFinal_OpeningFcn, ...
                   'gui_OutputFcn',  @ProyectoFinal_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before ProyectoFinal is made visible.
function ProyectoFinal_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ProyectoFinal (see VARARGIN)

% Choose default command line output for ProyectoFinal
handles.output = hObject;
clc;
% Update handles structure
guidata(hObject, handles);

% UIWAIT makes ProyectoFinal wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
function varargout = ProyectoFinal_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;



function N_Callback(hObject, eventdata, handles)
% hObject    handle to N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of N as text
%           str2double(get(hObject,'String')) returns contents of N as a
double


% --- Executes during object creation, after setting all properties.
function N_CreateFcn(hObject, eventdata, handles)
% hObject    handle to N (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function M_Callback(hObject, eventdata, handles)
% hObject    handle to M (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of M as text
%           str2double(get(hObject,'String')) returns contents of M as a
double


% --- Executes during object creation, after setting all properties.
function M_CreateFcn(hObject, eventdata, handles)
% hObject    handle to M (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function N_ITER_Callback(hObject, eventdata, handles)
% hObject    handle to N_ITER (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of N_ITER as text
%         str2double(get(hObject,'String')) returns contents of N_ITER as
a double


% --- Executes during object creation, after setting all properties.
function N_ITER_CreateFcn(hObject, eventdata, handles)
% hObject    handle to N_ITER (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function N_PART_Callback(hObject, eventdata, handles)
% hObject    handle to N_PART (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of N_PART as text
%         str2double(get(hObject,'String')) returns contents of N_PART as
a double


% --- Executes during object creation, after setting all properties.
function N_PART_CreateFcn(hObject, eventdata, handles)
% hObject    handle to N_PART (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function N_DEP_Callback(hObject, eventdata, handles)
% hObject    handle to N_DEP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of N_DEP as text
%        str2double(get(hObject,'String')) returns contents of N_DEP as a
double


% --- Executes during object creation, after setting all properties.
function N_DEP_CreateFcn(hObject, eventdata, handles)
% hObject    handle to N_DEP (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function TAO_Callback(hObject, eventdata, handles)
% hObject    handle to TAO (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of TAO as text
%         str2double(get(hObject,'String')) returns contents of TAO as a
double


% --- Executes during object creation, after setting all properties.
function TAO_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TAO (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function W0_Callback(hObject, eventdata, handles)
% hObject    handle to W0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of W0 as text
%          str2double(get(hObject,'String')) returns contents of W0 as a
double


% --- Executes during object creation, after setting all properties.
function W0_CreateFcn(hObject, eventdata, handles)
% hObject    handle to W0 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function W1_Callback(hObject, eventdata, handles)
% hObject    handle to W1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of W1 as text
%          str2double(get(hObject,'String')) returns contents of W1 as a
double


% --- Executes during object creation, after setting all properties.
function W1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to W1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function W2_Callback(hObject, eventdata, handles)
% hObject    handle to W2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of W2 as text
%          str2double(get(hObject,'String')) returns contents of W2 as a
double


% --- Executes during object creation, after setting all properties.
function W2_CreateFcn(hObject, eventdata, handles)
% hObject    handle to W2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function W3_Callback(hObject, eventdata, handles)
% hObject    handle to W3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of W3 as text
%          str2double(get(hObject,'String')) returns contents of W3 as a
double


% --- Executes during object creation, after setting all properties.
function W3_CreateFcn(hObject, eventdata, handles)
% hObject    handle to W3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function V_INICIAL_Callback(hObject, eventdata, handles)
% hObject    handle to V_INICIAL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of V_INICIAL as text
%         str2double(get(hObject,'String')) returns contents of V_INICIAL
as a double


% --- Executes during object creation, after setting all properties.
function V_INICIAL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to V_INICIAL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



% --- Executes on button press in RESOLV.
function RESOLV_Callback(hObject, eventdata, handles)

%--------------------------Boton Resolver-------------------------
global n m n_iter n_part n_dep tao;
global w0 w1 w2 w3;
global salida k sbg;
global dist sum v bg dem cap cap_rest tol;

%--------------------------Iteracion-------------------------
cont_t=1; %Contador del numero de iteraciones inicializado en 1
for t=1:n_iter %Ciclo del numero de iteraciones

%--------------------------Ordenar depositos-------------------------
cont=0; %Contador de depositos
orden_dep=[1:n_dep]*0;
while cont < n_dep %Condicion del ciclo
for i=1:n_dep
        p_dep(i)=.5;
R=rand(1,1); %Random entre 0 y 1
if  R <= p_dep(i) && orden_dep(i)==0 %Condicion para odenar depositos
cont=cont+1; %Aumenta contador
            orden_dep(i)=cont;  %Iguala  el  vector  ordenar  depositos  al
valor del contador
end
end
end

cont_q=1;
for q=1:n_part %Ciclo del numero de particulas

if t==1 %Condicion para el mejor b(q)
b(q)=sum; %b(q) se hace igual que la suma de todos los datos
end

%--------------------------Asignacion UB-------------------------
UB_asig=[1:n]*0;
UB_no=[1:n]*0;
for i=1:n_dep %Lee fila por fila
RR=rand(1,1); %Random entre 0 y 1
if RR<=.5 %Condicion para random menor que 0.5
for j=1:n %Lee columna por columna
```

```
UB_no(j)=0;
if UB_asig(j)==0
for l=1:m %Lee m
if dem(j,l)>(cap_rest(orden_dep(i),l)*(1+tol) ) %Condicion de asignacion
                        UB_no(j)=1;
end
end
if UB_no(j)==0
salida(orden_dep(i),j,q,t)=1;
cap_rest(orden_dep(i),l)=cap_rest(orden_dep(i),l)-dem(j,l);
                    UB_asig(j)=1;
end
end
end
end
end

for j=1:n %Lee columna por columna
if UB_asig(j)==0
for i=1:n_dep %Lee fila por fila
            UB_no(j)=0;
if UB_asig(j)==0
for l=1:m %Lee m
if dem(j,l)>(cap_rest(i,l)*(1+tol) )%Condicion de asignacion
                        UB_no(j)=1;
end
end
if UB_no(j)==0
salida(i,j,q,t)=1;
for l=1:m %Lee m
                        cap_rest(i,l)=cap_rest(i,l)-dem(j,l);
end
                    UB_asig(j)=1;
end
end
end
end
end



%-------------------------Calcular distancia-------------------------
d_total=0;
for i=1:n_dep
for j=1:n
d_total=d_total+salida(i,j,q,t)*dist(i,j); %Determina la distancia total
end
for l=1:m
        cap_rest(i,l)=cap(i,l);
end
end

%-------------------------Actualizar valores-------------------------
```

```
if d_total<bg
bg=d_total;
for i=1:n_dep
for j=1:n
sbg(i,j)=salida(i,j,q,t); %Almacena salida en en la matriz sbg
end
end
end

if d_total<b(q)
b(q)=d_total;
for i=1:n_dep
for j=1:n
sb(i,j,q)=salida(i,j,q,t); %Almacena salida en en la matriz sb
end
end
end

%-------------------------Actualizar parametros-------------------------
-
s=randn(1,4);

w0=w0*(1+tao*s(1,1)); %sobreescribe en W0
w1=w1*(1+tao*s(1,2)); %sobreescribe en W1
w2=w2*(1+tao*s(1,3)); %sobreescribe en W2
w3=w3*(1+tao*s(1,4)); %sobreescribe en W3

for j=1:n
    RRR=rand(1,1);
if RRR<=w3
        sbg_alt(i,j)=sbg(i,j); %Almacena la matriz sbg en sbg_alt
else
if sbg(i,j)==1
            sbg_alt(i,j)=0;
else
            sbg_alt(i,j)=1;
end
end
end

for i=1:n_dep
for l=1:m
        cap_rest(i,l)=cap(i,l);
end
end

if q<n_part
for i=1:n_dep
for j=1:n
                v(i,j,q+1)=v(i,j,q)*w0+w1*(salida(i,j,q,t)-sb(i,j,q))+...
w2*(salida(i,j,q,t)-sbg_alt(i,j)); %Guarda en la matriz velocidad con q+1
k(i,j)=1/(1+exp(-v(i,j,q+1))); %Guarda en la matriz K con q+1
end
```

```
end
else
for i=1:n_dep
for j=1:n
           v(i,j,1)=v(i,j,q)*w0+w1*(salida(i,j,q,t)-sb(i,j,q))+...
w2*(salida(i,j,q,t)-sbg_alt(i,j)); %Guarda en la matriz velocidad
k(i,j)=1/(1+exp(-v(i,j,1))); %Guarda en la matriz K
end
end
end

%--------------------------Aumento de contadores-------------------------
-
    cont_q = cont_q + n_dep + 3; %Aumenta contador de particulas
end
    cont_t = cont_t + n + 5; %Aumenta contador de iteraciones
end
set(handles.tab, 'Value', 1);
set(handles.VAR2,'data',num2cell(sbg(:,:))); %Tabla final
set(handles.SUMTOT, 'String', bg);
display('Los valores de b(q) son: ');%Mensaje
disp(b(:)) %Visuliza en el Workspace el valor de b(q)

guidata(hObject,handles)

% --- Executes on button press in LIMP.
function LIMP_Callback(hObject, eventdata, handles)

%--------------------------Boton limpiar-------------------------
global n n_dep;
clc;

%--------------------------Limpia variable por variable-----------------
-
set(handles.N,'String', ' ');
set(handles.M,'String', ' ');
set(handles.N_DEP,'String', ' ');
set(handles.N_ITER,'String', ' ');
set(handles.N_PART,'String', ' ');
set(handles.W0,'String', ' ');
set(handles.W1,'String', ' ');
set(handles.W2,'String', ' ');
set(handles.W3,'String', ' ');
set(handles.TAO,'String', ' ');
set(handles.V_INICIAL,'String', ' ');
set(handles.tab, 'String', ' ');
set(handles.TOL, 'String', ' ');
set(handles.SUMTOT, 'String', ' ');

%--------------------------Limpia tablas-------------------------
handles.tabla2=cell(n_dep,n);
handles.tabla2(:,:)={''};
set(handles.VAR2,'Data',handles.tabla2);
```

```
Tabla;

clear all;


% --- Executes when entered data in editable cell(s) in VAR1.
function VAR1_CellEditCallback(hObject, eventdata, handles)
%eventdata.NewData

% hObject    handle to VAR1 (see GCBO)
% eventdata  structure with the following fields (see UITABLE)
%   Indices: row and column indices of the cell(s) edited
%   PreviousData: previous data for the cell(s) edited
%   EditData: string(s) entered by the user
%    NewData: EditData or its converted form set on the Data property.
Empty if Data was not changed
%    Error: error string when failed to convert EditData to appropriate
value for Data
% handles    structure with handles and user data (see GUIDATA)
Data=set(hObject,'Data');


% --- Executes during object creation, after setting all properties.
function VAR1_CreateFcn(hObject, eventdata, handles)
% hObject    handle to VAR1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called


% --- Executes on button press in GENTAB.
function GENTAB_Callback(hObject, eventdata, handles)

%-------------------------Boton Generar-------------------------
global n m n_iter n_part n_dep tao;
global w0 w1 w2 w3 v_inicial tol;

%-------------------------Toma lo valores de cada caja----------------
--
n=str2double(get(handles.N,'String'));
m=str2double(get(handles.M,'String'));
n_iter=str2double(get(handles.N_ITER,'String'));
n_part=str2double(get(handles.N_PART,'String'));
n_dep=str2double(get(handles.N_DEP,'String'));
tao=str2double(get(handles.TAO,'String'));
w0=str2double(get(handles.W0,'String'));
w1=str2double(get(handles.W1,'String'));
w2=str2double(get(handles.W2,'String'));
w3=str2double(get(handles.W3,'String'));
v_inicial=str2double(get(handles.V_INICIAL,'String'));
tol=str2double(get(handles.TOL,'String'));
```

```
%-------------------------Genera pop-up menu---------------------
cont1=1;
for i=1:n_part*n_iter
if i==1
    l=i;
y{i}=[num2str(round(i)) '. TABLA PRINCIPAL'];
end
if l<=n_part
y{i+1}=[num2str(round(i+1)) '. Fila ' num2str(round(cont1)) ', Columna '
num2str(round(l))];
        l=l+1;
elseif l>n_part
        l=i-n_part*cont1;
        cont1=cont1+1;
y{i+1}=[num2str(round(i+1)) '. Fila ' num2str(round(cont1)) ', Columna '
num2str(round(l))];
        l=l+1;
end
end
set(handles.tab, 'String', y(:));

%-------------------------Genera tamaño de TABLA DE SALIDA--------------
-------

handles.tabla2=cell(n_dep,n);
handles.tabla2(:,:)={''};
set(handles.VAR2,'Data',handles.tabla2);

Tabla_Dist=cell(n_dep, n);
%Tabla_Dist(:,:)={''};
Tabla_Dem=cell(n,m);
%Tabla_Dem(:,:)={''};
Tabla_Cap=cell(n_dep,m);
%Tabla_Cap(:,:)={''};
save('Dist.mat','Tabla_Dist');
save('Dem.mat','Tabla_Dem');
save('Cap.mat','Tabla_Cap');

guidata(hObject,handles)


% --- Executes on button press in CLOSE.
function CLOSE_Callback(hObject, eventdata, handles)

%-------------------------Boton cerrar-------------------------
close all;
clear all;
clc;

% --- Executes on button press in ESTA.
function ESTA_Callback(hObject, eventdata, handles)
```

```
%-------------------------Boton estadistica-------------------------
Estadistica;


% --- Executes on selection change in tab.
function tab_Callback(hObject, eventdata, handles)

%-------------------------POP-UP MENU-------------------------
global n_part salida sbg;

val = get(hObject,'Value'); %Selecctor del pop-up menu

if val==1
set(handles.VAR2,'data',num2cell(sbg(:,:))); %Tabla final vector sbg
elseif val>1
    cont2=1;
for i=1:val-1
if i==1
            l=0;
end
if l<n_part
            l=l+1;
elseif l>=n_part
            l=l+1;
            l=i-n_part*cont2;
            cont2=cont2+1;
end
end
set(handles.VAR2,'data',num2cell(salida(:,:,l,cont2))); %Tabla de salida
vector salida
end
guidata(hObject,handles)

% Hints: contents = cellstr(get(hObject,'String')) returns tab contents
as cell array
%        contents{get(hObject,'Value')} returns selected item from tab


% --- Executes during object creation, after setting all properties.
function tab_CreateFcn(hObject, eventdata, handles)
% hObject    handle to tab (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: popupmenu controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
```

```
% --- Executes on button press in ING.
function ING_Callback(hObject, eventdata, handles)

%--------------------------Boton ingresar--------------------------
Tabla;



function SUMTOT_Callback(hObject, eventdata, handles)
% hObject    handle to SUMTOT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of SUMTOT as text
%        str2double(get(hObject,'String')) returns contents of SUMTOT as
a double


% --- Executes during object creation, after setting all properties.
function SUMTOT_CreateFcn(hObject, eventdata, handles)
% hObject    handle to SUMTOT (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end



function TOL_Callback(hObject, eventdata, handles)
% hObject    handle to TOL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of TOL as text
%        str2double(get(hObject,'String')) returns contents of TOL as a
double


% --- Executes during object creation, after setting all properties.
function TOL_CreateFcn(hObject, eventdata, handles)
% hObject    handle to TOL (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles      empty - handles not created until after all CreateFcns
called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if         ispc         &&         isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
set(hObject,'BackgroundColor','white');
end
```

Anexo C. Artículo “Modelo para la solución al problema de diseño de territorios comerciales, mediante la metaheurística de optimización evolutiva de enjambre de partículas (EPSO)”

MODELO PARA LA SOLUCIÓN AL PROBLEMA DE DISEÑO DE TERRITORIOS COMERCIALES, MEDIANTE LA META-HEURÍSTICA DE OPTIMIZACIÓN EVOLUTIVA DE ENJAMBRE DE PARTÍCULAS (EPSO)

MODEL FOR THE SOLUTION TO THE PROBLEM OF COMMERCIAL TERRITORY DESIGN THROUGH THE META-HEURISTIC EVOLUTIONARY OPTIMIZATION OF PARTICLE SWARM (EPSO)

SILVIA JULIANA ARIZA GARCÍA
Universidad Industrial de Santander, Colombia, silvia.ariza@correo.uis.edu.co
EMILY YOSELIN CARVAJAL CALDERÓN
Universidad Industrial de Santander, Colombia, Emily.carvajal@correo.uis.edu.co
HENRY LAMOS
Ph.D. en Matemáticas, Profesor Universidad Industrial de Santander, Colombia, hlamos@uis.edu.co

RESUMEN: En este artículo se presenta solución al problema de diseño territorial (TDPs), mediante la elaboración de un framework y la construcción de un toolbox en Matlab®, implementando la meta-heurística EPSO “Optimización por enjambre de partículas evolutivas”. El TDPs consiste en determinar una división de un conjunto de unidades ubicadas en un territorio que cumple con los criterios múltiples como la compacidad, la conectividad y el equilibrio en términos de clientes y la demanda del producto, constituyendo un soporte importante que no debe ser ignorado por los responsables de las actividades inherentes a la distribución comercial ya que se convierte en una decisión táctica para la empresa.
PALABRAS CLAVES: EPSO, Particle Swarm Optimization Evolution, TDP, Diseño Territorio, Ubicación, NP-hard, Clientes, Demanda.
ABSTRACT: This work of investigation is to develop a solution to the problem of territorial design (TDPs), through the development of a framework and building a toolbox in Matlab, implementing the meta-heuristic EPSO "Evolutionary Swarm Optimization particles." The TDPs is to determine a division of a set of units located in a territory that meets multiple criteria such as compactness, connectivity and balance in terms of customer and product demand, constituting an important support that should not be ignored by those responsible for the activities related to the commercial distribution as it becomes a tactical decision for the company.
KEY TERMS: EPSO, Particle Swarm Optimization Evolution, TDP, Territory Design, Find out, NP-hard, Customers, Demand.

1. INTRODUCCIÓN

El diseño de territorios permite a las empresas dedicadas a las ventas de bienes y servicios mejorar la competitividad con el objetivo de optimizar la rentabilidad. En la actualidad las empresas implementan estrategias innovadoras para mejorar su nivel de servicio al cliente. Una actividad importante a considerar es la distribución de territorios comerciales, la cual permite a la organización ajustarse adecuadamente a las necesidades de los clientes, mejorando su cobertura y ayudando a la compañía a mantener su fuerza de ventas, tiempos de viaje y costos bajo control.
El problema de diseños de territorios consiste en determinar una agrupación de manzanas, códigos postales o clientes individuales que se denominan territorios o unidades básicas (UBs), en un área geográfica determinada, en un número fijo de territorios de manera que se cumplan una serie de requerimientos de planificación impuestos por la empresa.

2. DEFINICIÓN DEL PROBLEMA

El problema de diseño de territorios se basa en un conjunto de unidades básica (UBs) - manzanas en este caso- y cercanía entre manzanas. Cada manzana posee unos nodos j (C1j; C2j), los cuales representan los clientes y el volumen en ventas. Hace parte fundamental del problema el uso de una distancia Euclidiana, dij, esta se calcula para cada UBs i y j. Las UBs deben ser agrupadas en p territorios de manera tal que cada nodo pertenezca sólo a un territorio. Esta empresa desea dividir las Unidades Básicas de la mejor manera, teniendo en cuenta criterios tales como económicos y geográficos, logrando así una distribución equitativa en las fuerzas de ventas.

Una restricción significativa es la de conexidad, es decir, entre las UBs debe existir una ruta entre ellas y estas deben pertenecer al mismo territorio. De igual forma, se debe lograr la compacidad entre las UBs del mismo territorio, es decir, se deben encontrar tan cerca una de la otra como sea posible.
Para esto en necesario lograr minimizar la medida de dispersión, maximizando la compacidad entre las unidades básicas. En este trabajo estudia dos medidas, una basada en el objetivo del problema de p-centro (pCP) y la otra basada en el objetivo del problema p-mediana (pMP). Esto conduce a dos modelos diferentes que se describen a continuación.

MODELOS DE OPTIMIZACIÓN

El Modelo que se estudia en este trabajo fue introducido por R.Z. RÍOS-MERCADO, M.A. SALAZAR-AGUILAR y M. CABRERA-RÍOS en el artículo de investigación "New Models for Commercial Territory Design" Published online: 7 January 2011. Springer Science Bussiness Media, LLC 2011.
Para presentar el modelo, se define primero al conjunto N^i, para todo i en V como el conjunto de todas las UBs adyacentes a la UB i, es decir,

$$N^i = \{j \in V : (i,j) \in E \vee (j,i) \in E\}$$

Las variables de decisión se definen como:

$$x_{ij} = \begin{cases} 1 \text{ si la UB } i \text{ es asignada al territorio con centro en } j \\ 0 \text{ en otro caso} \end{cases}$$

Note que $x_{ii} = 1$ implica que la UB i es un centro territorial.

El modelo matemático MTDP (TDP basado en p-mediana) se define a continuación.

(MTDP)

$$\text{minimizar} z = \sum_{j \in V} \sum_{i \in V} d_{ij}\, x_{ij} \qquad (1)$$

$$\text{sujeta a} \sum_{i \in V} x_{ii} = p \qquad (2)$$

$$\sum_{i \in V} x_{ij} = 1 \quad j \in V \qquad (3)$$

$$\sum_{j \in V} w_j^a\, x_{ij} \geq (1 - \tau^a)\mu^a x_{ii} \quad i \in V, a \in A \qquad (4)$$

$$\sum_{j \in V} w_j^a\, x_{ij} \leq (1 - \tau^a)\mu^a x_{ii} \quad i \in V, a \in A \qquad (5)$$

$$\sum_{j \in \cup_{v \in S} \frac{N^v}{S}} x_{ij} - \sum_{j \in S} x_{ij} \geq 1 - |S| \quad i \in V, S \subset \frac{V}{N^i \cup \{i\}} \qquad (6)$$

$$x_{ij} \in \{0,1\} \quad i, j \in V \qquad (7)$$

El objetivo (1) representa una medida de dispersión basada en el objetivo del *p*MP. En este sentido, minimizar dispersión es equivalente a maximizar compacidad. La restricción (2) garantiza la creación de exactamente p territorios. Las restricciones (3) representan la asignación exclusiva de las UBs.

Las restricciones (4)-(5) representan el balance con respecto a cada actividad y establecen que el tamaño de cada territorio debe estar dentro de un rango de variación (determinado por τ^a) con respecto al tamaño promedio. La conexidad de los territorios está dada por las restricciones (6). Estas últimas son similares a las restricciones de eliminación de subrutas (subtours) en el problema del agente viajero.

La cantidad de estas restricciones es un número exponencial por lo cual escribirlas explícitamente resulta prácticamente imposible. El método de solución propuesto genera de manera iterativa las restricciones de conexidad necesarias para encontrar una solución óptima del problema. Este modelo puede ser visto como el problema *p*MP con múltiples restricciones de capacidad y restricciones adicionales (4)-(6).

Cuando la medida de dispersión utilizada es el objetivo del *p*CP, la función objetivo (1) es reemplazada por la función (8). El

modelo resultante es llamado CTDP y fue introducido en[1].

$z = max_{i,j \in V}\{d_{ij}x_{ij}\}$ (8)

Estos modelos pertenecen a la clase NP-hard. Pruebas de complejidad para problemas similares en el contexto de distritos políticos pueden encontrarse en Altman[2].

Estos modelos pertenecen a la clase NP-duro. Cabe señalar que aunque en teoría, las restricciones de conexidad pudieran ser escritas explícitamente, esto no tendría ningún sentido práctico debido a su número exponencial.

Denominemos como R_MTDP al modelo relajado que se obtiene de relajar las restricciones (6) del MTDP. De manera similar definimos el modelo relajado R_CTDP como el modelo resultante al eliminar (6) en CTDP.

Se introduce nuevas formulaciones matemáticas del problema utilizando optimización cuadrática entera (IQP). El número de variables se redujo de n^2 a $2np$. En el modelo cuadrático, hacemos uso de los mismos parámetros utilizados en el modelo lineal. Se define un conjunto adicional $Q = \{1,2,\dots,p\}$ de índices de territorios y un conjunto de variables binarias y_{ip} para identificar los centros de los territorios y z_{jq} para representar la asignación de UBs a territorios. Las variables de decisión para el modelo IQP se definen así:

$$Z_{jq} = \begin{cases} 1 & si\ la\ UB\ j\ es\ asignada\ al\ territorio\ q \\ 0 & en\ otro\ caso \end{cases}$$

$$Y_{iq} = \begin{cases} 1 & si\ la\ UB\ j\ es\ asignada\ al\ territorio\ q \\ 0 & en\ otro\ caso \end{cases}$$

De acuerdo con las definiciones anteriores, la equivalencia existente entre las variables utilizadas en el modelo lineal y las utilizadas en el cuadrático está dada por:

$X_{ij} = \sum_{q \in Q} Z_{jq} Y_{iq}$ (9)

El modelo QMTDP (*quadratic median-based territory design problema*) usa una medida de dispersión equivalente a la utilizada en el modelo MTDP. A continuación se muestra la formulación del modelo QMTDP.

(QMTDP)

$$\text{minimizar} z = \sum_{q \in Q}\sum_{j \in V}\sum_{i \in V} d_{ij}\, Z_{jq}\, Y_{iq} \qquad (10)$$

$$\text{sujeta a} \sum_{i \in V} Y_{iq} = 1 \qquad q \in Q \qquad (11)$$

$$\sum_{q \in Q} Z_{jq} = 1 \qquad j \in V \qquad (12)$$

$$Z_{jq} \geq Y_{jq} \qquad q \in Q, j \in V \qquad (13)$$

$$\sum_{j \in V} w_j^a\, Z_{jq} \geq (1 - \tau^a)\mu^a \qquad q \in Q, a \in A \qquad (14)$$

$$\sum_{j \in V} w_j^a\, Z_{jq} \leq (1 + \tau^a)\mu^a \qquad q \in Q, a \in A \qquad (15)$$

$$\sum_{q\in Q}\sum_{j\in U_{v\in S}N^v/S} Z_{jq}Y_{iq} - \sum_{q\in Q}\sum_{j\in S} Z_{jq}\,Y_{iq} \geq 1 - |S|$$

$$i \in V, S \subset \frac{V}{N^i \cup \{i\}} \quad (16)$$

$$Z_{jq} \in \{0,1\} \qquad q \in Q, j \in V \qquad (17)$$

$$Y_{iq} \in \{0,1\} \qquad q \in Q, j \in V \qquad (18)$$

Las restricciones (11) son usadas para garantizar la asignación de un centro por cada territorio. La asignación exclusiva está dada por las restricciones (12). El balance territorial se establece con las restricciones (14)-(15). Las restricciones (13) indican que una UB j no puede ser centro del territorio q si j no pertenece al territorio q. El último conjunto de restricciones (16) garantizan la conexidad de los territorios. Nuevamente tenemos un número exponencial de estas restricciones.

Bajo esta formulación cuadrática, una medida de dispersión basada en el objetivo *pCP*está dada por (19). Entonces, el QCTDP (*quadratic center-based territory design problema)* es el modelo resultante al reemplazar la función objetivo (10) por la medida de dispersión (19).

$$\min \; z = max_{i,j\in V}\left\{\sum_{q\in Q} Z_{jq}Y_{iq}\right\} \qquad (19)$$

Cabe recalcar que estas formulaciones IQP son nuevas en la literatura de diseño de territorios. QMTDP es difícil de resolver debido a que posee un objetivo cuadrático y un conjunto de restricciones cuadráticas (restricciones de conexidad). Adicionalmente, no es posible escribirlas explícitamente debido a su número exponencial. Si las restricciones de conexidad son relajadas, el modelo puede resolverse utilizando cualquier optimizador para problemas MINLP.

Similar a la definición de los modelos relajados R_MTDP y R_CTDP, definimos R_QMTDP como la relajación de QMTDP cuando las restricciones (16) son removidas del modelo. Claramente, una solución para R_QMTDP brinda una cota inferior para QMTDP. Hay algunos casos especiales para los cuales el modelo puede ser reforzado, por ejemplo, cuando no hay soluciones factibles que contengan territorios formados por una sola UB. Es decir, cuando cada solución factible tiene territorios con al menos dos unidades básicas asociadas a él, la siguiente es una desigualdad válida para R_QMTDP.

$$\sum_{i\in N} Z_{iq} \geq Z_{jq} \qquad q \in Q, j \in V \qquad (20)$$

Estas desigualdades evitan la creación de subconjuntos no conexos S tales que $|S| = 1$. Existen un número polinomial de estos subconjuntos, así que estas desigualdades pueden incorporarse fácilmente al modelo. Note que, para las formulaciones MILP las desigualdades válidas equivalente están dadas por:

$$\sum_{i\in N} X_{il} \geq X_{ij} \qquad i \in V, j \in \frac{V}{(\{i\} \cup N^i)} \quad (21)$$

En contraste con las desigualdades dadas en (20) que sólo son válidas cuando permanece la condición de territorios formados por más de una unidad básica, las restricciones (21) son válidas para

cualquier instancia. Definamos entonces a R1_QMTDP como la relajación conformada por R_QMTDP más las restricciones de adicionales (20). De manera similar se puede definir modelos relajados para el modelo QCTDP. No referimos a éstos como R_QCTDP y R1_QCTDP, respectivamente. Igualmente, para los modelos MTDP y CTDP, se obtienen nuevos modelos relajados agregando (21) en los modelos relajados R_MTDP y R_CTDP. Los denominamos como R1_MTDP y R1_CTDP, respectivamente.

3. OPTIMIZACIÓN EVOLUTIVA DE ENJAMBRE DE PARTÍCULAS PARA EL DISEÑO TERRITORIAL COMERCIAL

La motivación de desarrollar la presente herramienta para la solución de problemas de diseño territorial comercial deriva del hecho que en competencia con otros algoritmos metaheurísticos, este ha logrado mejores resultados en muchos casos, posicionando el método como una opción para resolver problemas de optimización complejos.
El problema que se estudia se basa en el trabajo de R.Z. RÍOS-MERCADO, M.A. SALAZAR-AGUILAR y M. CABRERA-RÍOS . En el trabajo se aborda un problema de toma de decisiones que surge en el campo de diseño territorial como una aplicación de una empresa distribuidora de bebidas embotelladas, en el cual el problema consiste en determinar una agrupación de unidades básicas, dentro de una área geográfica objetivo, en un número fijo de territorios de tal manera que se cumplan una serie de requerimientos de planificación.

5.1.2 Algoritmo EPSO

En el EPSO cada partícula representa una solución, la cual evolucionara en un espacio de búsqueda para lograr encontrar un punto óptimo dentro de este mismo, cada una de estas partículas tiene asociadas unas velocidades y unos para metros que la llevan siempre hacia la mejor solución.

Es necesario definir la trayectoria, velocidad y movimiento los cuales se pueden definir en términos de las probabilidades de cambio, para que una variable cambie de un estado al otro, es por esto que en el algoritmo binario una partícula se mueve en cada una de sus coordenadas en un espacio restringido de [0 1] dependiendo de su velocidad, donde cada V representa la probabilidad de que una de las variables de decisión de la partícula tome determinado valor. En otras palabras, si para una partícula la velocidad de una de sus variables de decisión es V=0,20 entonces hay una probabilidad del veinte por ciento de que esta variabsle tome el valor de 1, es decir, se asigna la UB al depósito y un ochenta por ciento de probabilidad de tomar el valor de 0, no se asigna.
La velocidad en el algoritmo EPSO no está representada como una probabilidad. La velocidad se rige bajo la siguiente ecuación.

$$V_i^{t+1} = W_{i0}^* v_i^t + W_{i1}^* \left(S_i^t - b_g\right) + W_{i2}^* \left(S_i^t - b_g^*\right)$$

Es por esto que se utiliza la función sigmoide para convertir la expresión de la velocidad en una probabilidad.

$$K_{\left(v_i^k\right)} = \frac{1}{1 + e^{-\left(v_i^t\right)}}$$

Y por último la regla de movimiento de la partícula de una iteración k a una iteración k+1 está dada por la siguiente ecuación:

$$S_i^{t+1} = \begin{cases} 1 \; si \; rand() \leq K_{(v_i^t)} \\ 0 \; si \; rand() > K_{(v_i^t)} \end{cases}$$

El algoritmo utilizado para resolver el TDP, se presenta a continuación. En primer lugar se inicializa los parámetros para luego generar los enjambres y se le asigna los pesos de los depósitos para cada unidad básica; de esta manera se genera las partículas para luego su posterior asignación. Consecutivamente se calcula la distancia total y se actualizan los pesos, la mejor posición, la velocidad y el peso de la velocidad. Se continúa con la siguiente iteración

Sean

$\boldsymbol{n}$: *Número de UB*

$\boldsymbol{m}$: *Cantidades de tipos de productos*

$\boldsymbol{n\_dep}$: *Número de depósitos*

$\boldsymbol{n_{iter}}$: *Número de iteraciones*

$\boldsymbol{n_{part}}$: *Número de partículas por iteración*

$\boldsymbol{tao}$: *Grados de libertad del aprendizaje*

$\boldsymbol{tol}$: *Tolerancia relativa con respecto a la cantidad de producto a manejar*

$\boldsymbol{w_0}$: *Peso del término inercia*

$\boldsymbol{w_1}$: *Peso del término de memoria*

$\boldsymbol{w_2}$: *Peso del término de cooperación*

$\boldsymbol{w_3}$: *Parámetro estratégico (peso) asociado con la partícula i.*

$\boldsymbol{d_{total}}$: *Suma de las distancias*

$$\boldsymbol{S_{ijqt}} = \begin{cases} 1 \; Si \; la \; UB \; es \; asinada \; al \; depósito \; i \\ 0 \; de \; lo \; contrario \end{cases}$$

$\boldsymbol{bg}$: *Distancia de la mejor posición*

$\boldsymbol{cap_{rest(j)}}$: *Capacidad restante durante la asignación de UB*

$\boldsymbol{cdem(j)}$: *Demanda estimada por UB*

$\boldsymbol{k(i,j)}$: *Peso de la velocidad adaptada por la función sigmoide*

$\boldsymbol{sbg(i,j)}$: *Posición de la mejor solución global generada por el enjambre*

$\boldsymbol{vinicial(i,j)}$: *Velocidad inicial de las partículas*

Tabla 1. Framework EPSO aplicado al Diseño Territorial Comercial (EPSO-TDP)

EPSO-TDP
Inicializar parámetros
Elegir *n_dep* depositos y las *n* UB
Generación de enjambres
Para *t* =1 hasta *n_iter*
Asignar *p_dep* para asignacion de UB
Generación de partículas
Para *q*= 1 hasta *n_part*
Asignación de UB:
Si k < rand[0,1]
Verificar capacidades *s(i,j,q,t)=1*
Fin si
Calcular distancia total, *dtotal*
Actualizar *sb* y *sbg*
Actualizar *w0, w1, w2, w3*
Generar *sbg**
Actualizar *v, k*
Fin para
Fin para

Para el desarrollo de este algoritmo se construye en Matlab un Toolbox, el cual consiste en encontrar la distancia mínima generada y la asignación final de las unidades básicas en los depósitos.

Para ordenar los depósitos es necesario darle un peso de importancia a cada depósito, en este trabajo el peso de 0.5.

Luego por medio de un random asignamos de manera aleatoria, entonces si el random es menor o igual al peso de 0.5 y si ese depósito es igual a cero (es decir que no se ha asignado) precedemos a asignarlo y guardarlo en orden_dep(i). Finalmente se realiza lo mismo para todos los depósitos.

Para la asignación de las UB, si random es menor o igual a 0.5 y si la UB no ha sido asignada entonces tenemos en cuenta la demanda por medio de la capacidad y la tolerancia dada. Luego de eso se asigna si cumple con el criterio. Finalmente se realiza lo mismo para todas las UB. Recordando que cada UB debe ser asignada a un solo depósito.

4. RESULTADOS COMPUTACIONALES

El algoritmo EPSO para resolver el problema de Diseño Territorial Comercial fue programado en Matlab versión R2012a y ejecutado en un equipo con procesador Intel Core i3 y 4GB de RAM instalada. Los parámetros asociados al EPSO son:

Tabla 2. Parámetros del EPSO establecidos

Parámetros	**Valor**	
Número de UB	n	11
Cantidad de tipos de productos	m	7
Número de iteraciones	n_iter	5
Número de partículas por iteración	n_part	10
Número de depósitos	n_dep	5
Tao	tao	0,5
Tolerancia	tol	0,5

Ejemplo: Este ejemplo considera un sistema de 11 UB y 5 depósitos. Las distancias dadas entre cada depósito y la UB se presentan en la tabla 3. Cada situación de capacidad y demanda se muestran en las tablas 4 y 5 respectivamente.

Tabla 3. Distancias entre depósitos y UBs

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**CAP**
1	5	6	5	15	19	7	11	13	10	13	10	3100
2	15	12	13	12	6	17	3	6	11	16	5	3100
3	115	5	17	18	12	15	10	5	7	14	6	2800
4	8	10	12	17	6	10	9	14	12	18	13	3200
5	10	11	12	9	7	12	12	7	9	14	9	2530
DEM	942	602	752	568	889	579	656	576	797	731	535	

Tabla 4. Capacidad de productos en depósitos

DEP	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**TOTAL**
1	300	650	400	200	350	300	900	3100
2	400	500	300	200	300	400	1000	3100
3	300	450	200	200	400	300	950	2800
4	450	550	350	200	450	400	800	3200
5	280	500	300	200	300	200	750	2530

Tabla5. Demanda de productos por cada cliente								
Clientes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**TOTAL**
1	158	145	20	65	45	59	450	942
2	120	67	25	28	123	39	200	602
3	54	200	35	46	89	98	230	752
4	86	120	45	23	35	59	200	568
5	125	86	55	76	45	72	430	889
6	72	92	75	29	67	94	150	579
7	149	90	45	53	52	67	200	656
8	80	128	30	24	85	49	180	576
9	135	79	35	48	36	104	360	797
10	75	170	55	34	47	100	250	731
11	45	155	25	55	53	52	150	535

Como se hizo mención anteriormente, este ejemplo se corre con el algoritmo EPSO. En la tabla 6. Se observan los resultados obtenidos por algoritmo, la asignación de las unidades básicas a cada depósito se realiza teniendo en cuenta los resultados binarios, cuando el resultado es 1 la UB es asignada al depósito, de igual forma cabe resaltar que para cada UB solo se puede asignar a un depósito.

Tabla 6. Resultados obtenidos por el algoritmo EPSO											
bg*	1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	1	1	0	1	0	1	0
2	0	0	0	1	0	0	0	0	0	0	0
3	0	0	1	0	0	0	1	0	1	0	0
4	1	0	0	0	0	0	0	0	0	0	1
5	0	1	0	0	0	0	0	0	0	0	0

La asignación de las unidades básicas, teniendo en cuenta los parámetros se realiza de la siguiente manera:
Deposito 1: 5-6-8-10
Deposito 2: 4
Deposito 3: 3-7-9
Deposito 4: 1-11
Deposito 5: 2
Los experimentos numéricos efectuados para evaluar el desempeño del EPSO, se llevaron a cabo aplicando el modelo en un conjunto de instancias desarrolladas para el Diseño de Territorios Comerciales. Se trata de 3 instancias en la cual están conformadas por los siguientes elementos: la matriz de distancia entre los depósitos y las unidades básicas; la capacidad de los 7 tipos de productos en los depósitos; la demanda de los 7 tipos de productos por cada uno de los clientes.
La primera instancia es de 11 unidades básicas y 5 depósitos, la segunda instancia es de 44 unidades básicas y 15 depósitos. Finalmente la última instancia en de 88 unidades básicas y 20 depósitos. Los parámetros para estas 3 instancias están descritos en la tabla 7. De igual forma se detalla los resultados

encontrados de las distancias mínimas para diferentes tolerancias.

Tabla 7. Parámetros para EPSO con resultados para 3 instancias.

UB		11	44	88
Depósitos		5	15	20
Iteraciones		5	5	5
Partículas		10	10	10
Productos		7	7	7
Tao		0.5	0.5	0.5
W0		0.33	0.33	0.33
W1		0.33	0.33	0.33
W2		0.33	0.33	0.33
W3		0.5	0.5	0.5
Velocidad inicial		0.5	0.5	0.5
		Distancia Min		
Tolerancia	**0.3**	92	421	899
	0.5	95	411	903
	0.9	83	399	903

En los resultados obtenidos para las diferentes instancias se pudo observar que la tolerancia relativa con respecto a la cantidad de productos a manejar es un elemento significativo en el resultado. Sin embargo no se puede generalizar su aumento o disminución, puesto que se refleja resultados diferentes.

Es necesario tener en cuenta que, si el número de depósitos es el ideal para esa cantidad de unidades básicas, es decir, que si la oferta es mayor que la demanda, por lo que en algunas instancias ciertos depósitos no son asignados. Puede ser porque algún depósito se encuentra muy lejos de los demás o que con menos depósitos es suficiente para la asignación de todas las unidades básicas

Teniendo en cuenta las mismas instancias, se experimenta con 10 iteraciones y 20 partículas, los resultados y los parámetros están descritos en la tabla 8.

Tabla 8. Resultados con 10 iteraciones y 20 partículas

UB		11	44	88
Depósitos		5	15	20
Iteraciones		10	10	10
Partículas		20	20	20
Productos		7	7	7
Tao		0.5	0.5	0.5
W0		0.33	0.33	0.33
W1		0.33	0.33	0.33
W2		0.33	0.33	0.33
W3		0.5	0.5	0.5
Velocidad inicial		0.5	0.5	0.5
		Distancia Min		
Tolerancia	**0.3**	92	393	890
	0.5	93	411	898
	0.9	83	413	873

En las 3 instancias se encuentran mejores resultados en comparación con la tabla 7. Para 88 unidades básicas y 20 depósitos existe una mejor solución de 873; esto quiero decir que al aumentar el espacio de búsqueda las partículas se direccionan a una solución óptima local.

Para un mayor número de iteraciones y partículas se presentó un inconveniente con 88 unidades básicas y 20 depósitos. La memoria del computador no es suficiente para desarrollar el modelo con 100 iteraciones y 200 partículas. Esto refleja una limitante a la hora de aumentar el espacio de búsqueda. Sin embargo para 44 unidades básicas y 15 depósitos se encontró una mejor solución de 377 de

distancia. Los resultados están descritos en la tabla 9.

Tabla 9. Resultados con 100 iteraciones y 200 partículas				
UB		11	44	88
Depósitos		5	15	20
Iteraciones		100	100	100
Partículas		200	200	200
Productos		7	7	7
Tao		0.5	0.5	0.5
W0		0.33	0.33	0.33
W1		0.33	0.33	0.33
W2		0.33	0.33	0.33
W3		0.5	0.5	0.5
Velocidad inicial		0.5	0.5	0.5
		Distancia Min		
Tolerancia	**0.3**	92	382	-
	0.5	93	386	-
	0.9	83	377	-

Teniendo en cuenta el tiempo de respuesta, Matlab demora alrededor de 15 a 20 segundos para arrojar los resultados con 88 UB y 20 depósitos. Para el resto de pruebas numéricas los resultados son de inmediato.

Finalmente se realiza pruebas numéricas con una velocidad de 0.2 como lo muestra la tabla 10. En esta ocasión no refleja cambio alguno al disminuir la velocidad inicial. Sigue encontrando la misma distancia mínima generada con 10 iteraciones y 20 partículas.

Tabla 10. Cambios en Velocidad				
UB		11	44	88
Depósitos		5	15	20
Iteraciones		10	10	10
Partículas		20	20	20
Productos		7	7	7
Tao		0.5	0.5	0.5
W0		0.33	0.33	0.33
W1		0.33	0.33	0.33
W2		0.33	0.33	0.33
W3		0.5	0.5	0.5
Velocidad inicial		0.2	0.2	0.2
		Distancia Min		
Tolerancia	**0.3**	92	393	890
	0.5	93	411	898
	0.9	83	413	873

En la tabla 7, Se puede observar la mínima distancia generada con las diferentes tolerancias. Concluyendo así, que a mayor tolerancia se logra obtener una mínima distancia, lo cual mejora la compacidad entre las Unidades Básicas.

Tabla 7. Tolerancia y Distancia Mínima		
(n;n_dep)	TOLERANCIA	DISTANCIA MÍNIMA
(11,5)	0	86
(11,5)	0.1	86
(11,5)	0.2	86
(11,5)	0.3	83
(11,5)	0.4	83
(11,5)	0.5	83
(11,5)	0.6	83
(11,5)	0.7	83
(11,5)	0.8	83
(11,5)	0.9	83
(11,5)	1	83

5. CONCLUSIONES

En el presente trabajo se resolvió el problema de Diseños de Territorios Comerciales con la metaheurística EPSO, utilizando un método de decodificación para la asignación de UB a los depósitos. Adicionalmente se efectuaron experimentos numéricos para evaluar el desempeño de EPSO sobre 3 instancias de unidades básicas y depósitos conformado de esta manera (11,5) (44,15) y (88,20).

Es necesario tener en cuenta el número de depósitos para las unidades básicas, ya que si es muy grande la cantidad de depósitos puede que estos no se asignen. Puede darse por el nivel de servicio o por la lejanía de algún depósito con respecto a las demás.

Se presentó inconvenientes con respecto a la memoria del computador para resolver en Matlab® con instancias de 88 unidades básicas y 20 depósitos al aumentar las iteraciones y partículas. Este modelo solo resuelve para instancias pequeñas y con pequeñas iteraciones.

Al aumentar el número de iteraciones, el espacio de búsqueda aumenta y se puede encontrar mejores resultados con respecto a la distancia mínima.

El tiempo de respuesta para pequeñas instancias es casi de inmediato aunque para instancias mayores demora alrededor de 15 a 18 segundos.

6. RECOMENDACIONES Y TRABAJOS FUTUROS

Para futuras investigaciones enfocadas a encontrar método de solución para el Diseño Territorial Comercial se recomienda tener en cuenta estudios sobre la convergencia de las velocidades de las partículas empleadas en las heurísticas y el estudio analítico sobre el comportamiento de sus demás parámetros.

En el campo de las TDPs se está trabajando en la implementación de una vecindad basada en intercambio de unidades básicas entre territorios vecinos para tener una mayor flexibilidad en la fase de postprocesamiento. Otra de las áreas de oportunidad es evaluar la sensitividad del algoritmo con respecto al parámetro de ponderación de la función miope.

Futuros trabajos estarán enfocados en la aplicabilidad de esta metaheurística, logrando demostrar el impacto en el sector, logrando escalabilidad de la herramienta generada y su utilidad para la optimización de la planeación logística como soporte a la toma de decisiones.

REFERENCIAS

[1]RÍOS-MERCADO, R.Z. Y FERNANDEZ, E. A reactive GRASP for a commercial territory design problem with multiple balancing requirements. Computers & Operations Research, 36(3), (2009); p. 755–776.

[2]ALTMAN, M. Districting Principles and Democratic Representation. Disertación doctoral, California Institute of Technology, Pasadena, EUA, (1998).

[3]RÍOS-MERCADO, R.Z; SALAZAR-AGUILAR, Maria Angelica y CABRERA-RÍOS, M.. “New Models for Commercial Territory Design” Published online: 7 January 2011. Springer Science Bussiness Media, LLC 2011.