"OPTIMIZACIÓN DE ENJAMBRE DE PARTÍCULAS (PSO) APLICADA AL PROBLEMA DE LA P-MEDIANA."

CESAR DARIO ALVAREZ CRUZ

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICO-MECANICAS

ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES

BUCARAMANGA

2013

"OPTIMIZACION DE ENJAMBRE DE PARTICULAS (PSO) APLICADA AL PROBLEMA DE LA P-MEDIANA."

CESAR DARIO ALVAREZ CRUZ

TRABAJO DE GRADO PARA OPTAR AL TITULO DE

INGENIERO INDUSTRIAL

DIRECTOR

HENRY LAMOS DIAZ

Ph.D en Fisica-Matematica

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERIAS FISICO-MECANICAS

ESCUELA DE ESTUDIOS INDUSTRIALES Y EMPRESARIALES

BUCARAMANGA

2013

AGRADECIMIENTOS

- Al grupo de investigación OPALO por que bajo su nombre se realizó este proyecto de investigación.

- Al profesor, Dr. Henry Lamos, por su confianza, por su tiempo y por su apoyo incondicional a lo largo de este trayecto.

- A mi familia y amigos que me motivaron en esta nueva etapa.

Tabla de Contenido

Lista de Figuras

Lista de Tablas

Lista de Apéndices

RESUMEN

TITULO
"Optimización de Enjambre de Partículas (PSO) aplicada al problema de la P-mediana."[1]

AUTOR
Cesar Dario Alvarez Cruz[2]

PALABRAS CLAVES
Optimización de Enjambre de Partículas Discretas (DPSO), metaheuristica, Problema de la P-mediana (PMP).

El problema de la P-mediana consiste en ubicar un conjunto de facilidades entre un grupo de cliente distribuidos en un área, de tal manera que las suma de las distancias entre cada cliente y la facilidad más cercana sea minimizada. El estudio de este problema ha despertado un gran interés en el área de la Investigación de Operaciones debido a su aplicación en campos como las telecomunicaciones, y la logística de transporte y distribución de la industria. Sin embargo, el problema planteado desde la teoría de Optimización Combinatoria, ha sido difícil de resolver debido al creciente tamaño de problemas prácticos que se tienen que abordar actualmente y a la característica de tiempo exponencial para resolver óptimamente el problema con los algoritmos disponibles en el momento.

En este estudio, se propone y describe una representación de una solución y un método de decodificación para aplicar la metaheuristica Enjambre de Partículas Discreto (DPSO) al problema de la P-mediana estándar (PMP). Aunque este método no garantiza la prueba de optimalidad en la solución resultante, puede ser utilizado como un método aproximado para obtener cotas superiores de calidad para problemas prácticos de gran tamaño en un tiempo computacional razonable.

El algoritmo propuesto es probado utilizando problemas de la librería OR-Library, y el código es implementado en el software MATLAB. Los resultados de los experimentos numéricos muestran desviacionesmínimas respecto a la soluciónóptima de cada problema, incluso en los casos másdifíciles que es cuando la proporción de medianas y vértices es alta.

[1] Proyecto de Grado
[2] Facultad de Ingenierías Fisicomecanicas . Escuela de Estudios Industriales y Empresariales.
Director: Henry Lamos Diaz

ABSTRACT

TITLE
"Particle Swarm Optimization Applied to the P-median Problem."[1]

AUTHORS
Cesar Dario Alvarez Cruz[2]

KEYWORDS
Discrete Particle Swarm Optimization (DPSO), metaheuristics, P-median Problem (PMP).

The P-median Problem consists of locating a set of facilities among a set of customers distributed over a given area in such a way that the total sum of distances between each customer and its nearest facility is minimized.This problem has aroused great interest in the field of Operations Research since it can be applied to problems related to telecommunications and industrial transportation and distribution. However, the way by which it can be described through Combinatorial Optimization theory, makes it difficult to solve given the current size of practical problems and the available methods that specifically solve the combinatorial interpretation of the problem.

In this study, we propose and describe a representation of solutions and codification method for applying the metaheuristics Discrete Particle Swarm Optimization (DPSO) to the P-median Problem (PMP). Although this method doesn't have the capability of proving optimality in the yielding solution, it can be used as an approximate method to get good lower bounds for practical problems of great size in a reasonable amount of time.

The algorithm proposed is tested on problems of the OR-Library, and implemented in MATLAB software. The results show minimum deviations to the optimal solution even in the most difficult cases which occurs when the proportion of medians and vertices is high.

[1] Proyecto de Grado
[2] Facultad de Ingenierías Fisicomecanicas . Escuela de Estudios Industriales y Empresariales.
Director: Henry Lamos Diaz

INTRODUCCIÓN

El estudio de los problemas del espacio o localización como determinantes de la actividad económica, es relativamente reciente en la economía. Los economistas clásicos y neoclásicos se preocuparon en su mayor parte por el problema del tiempo. No obstante, al alcanzar la economía industrial un alto grado de desarrollo, se comenzó a desarrollar la teoría de la localización industrial de manera constante y creciente. Gran parte de este auge se debió a que los bienes de capital y la tecnología de punta relativamente accesibles, la seguridad de mercados crecientes y la maximización de utilidades se convirtieron en elementos primordiales en las decisiones industriales. La localización de plantas empezó hacer un importante elemento de competencia entre las firmas.[1]

Los primeros estudios sobre la teoría del espacio se iniciaron en Alemania durante el siglo XIX. En esa época, un grupo de economistas alemanes empezaron las investigaciones sobre las relaciones espaciales en la actividad económica. Alfred Weber fue el primero en dar a los problemas de la localización industrial un tratamiento sistemático, el modelo consiste en buscar la localización más conveniente de una rama industrial específica. Weber asigno mayor importancia al mercado, a las materias primas y a la oferta de mano de obra. Estos tres factores estaban expresados en los costos de transporte, costos de mano de obra y fuerzas constitutivas aglomeradoras. Weber incluyo en su formulación los costos de las materias primas y combustible en los costos de transporte en términos de pesos y distancias; la combinación que suministraba la menor medida tonelada-kilo indicaba la ubicación más ventajosa. Los costos de mano de obra y las fuerzas de aglomeración eran modificados por decisiones políticas, por lo tanto, el factor fundamental de localización es la distancia: la distancia de la planta de producción a los recursos y al mercado.[2] Weber representa su teoría en un triángulo en el cual dos vértices corresponden a las materias primas e insumos

[1] DASKIN, Mark; SNYDER, Lawrence y BERGER, Rosemary.Facility Location in Supply Chain Design.p.3.
[2] CASAS GONZALES, Antonio. La Economía del Espacio y los Problemas de los Países en Vías de Desarrollo. p. 107-108

respectivamente, y el tercer vértice es el lugar del mercado. El objetivo del modelo (llamado más tarde *minsum*) era encontrar la mínima suma de distancias de la instalación a los vértices, y fue este el objetivo que domino la poca literatura de localización hasta mediados del siglo XIX.

La teoría de la localización tuvo poco desarrollo en los siguientes años debido a la ausencia de una teoría sólida y al nivel de complejidad en el cálculo de soluciones de problemas con aplicaciones reales. A mediados del siglo XX, sin embargo, la teoría de localización se convirtió en un área de gran interés debido a la confluencia de diferentes factores. Primero, se desarrolló la teoría económica del espacio. Esta teoría estudia los factores que han determinado la distribución geográfica de las actividades económicas y trata de proponer alternativas para el desarrollo de la estructura espacial. El crecimiento irregular de las ciudades e industrias así como el cambio social y económico de este periodo forzó a las empresas a ajustarse a las nuevas tendencias para ser más competitivos. Además, la llegada y el rápido desarrollo de los computadores permitieron realizar fácilmente la gran cantidad de cálculos que aparecen en los problemas de localización.[3]

Aunque existen algunos trabajos teóricos (Isard, 1962; Predohl, 1928; Cooper, 1963) e intentos de resolución de aplicaciones prácticas (Mansfield y Wein, 1958; Miehle, 1958; Valinsky, 1955) posteriores al trabajo de Weber, no es hasta mediados de los 60, cuando Hakimi (1964, 1965) abordo los problemas de localización mediante técnicas de investigación operativa. Hakimi retomo el objetivo *minisum* y formulo el problema en una red conectada por nodos, además definió la mediana como el nodo en el cual se minimiza las sumas de las distancias mínimas desde los clientes (nodos) hasta las instalaciones (medianas). Posteriormente, generalizo el problema para más de una instalación y definió el

[3] CANOS DAROS, M.J; MARTINEZ ROMERO, M. y MOCHOLI ARCE, M. Búsqueda de un Objetivo en Problemas de Localización. XIV Jornadas de ASEPUMA y II Encuentro Internacional, 2006. p. 2.

conjunto de p medianas tal que la suma de las distancias de los clientes a la instalación más cercana sea mínima, de esta manera acuñó el término P-mediana (P-median Problem o PMP por sus siglas en inglés) el cual sigue siendo utilizado en la literatura. Así mismo, demostró que el problema pertenece a una clase de problemas complejos denominados NP-hard .

Un aspecto importante en el desarrollo de nuevos algoritmos para el Problema de la P - mediana es la elaboración de algoritmos a través del modelado basado en la inteligencia biológica y natural. Los algoritmos inteligentes incluyen redes neuronales, la computación evolutiva, inteligencia de partículas, sistemas inmunes artificiales y sistemas difusos. Bajo el paradigma de Inteligencia de Partículas (SI), se encuentran los procedimientos de comportamiento social de organismos tales como Colonias de Hormigas (AC) y Enjambre de Particulas (PSO) como métodos para abordar la solución a problemas optimización combinatoria con altos requerimientos computacionales.

Dado el nivel de complejidad del PMP y su importancia como herramienta para la toma de decisiones empresariales, el presente trabajo propone una codificación binaria para aplicar la metaheuristica Quantum Particle Swarm Optimization en resolver el problema de la P-mediana.

Cumplimiento de Objetivos	
Objetivo Específicos	**Numerales Relacionados**
Recopilar, seleccionar y organizar la literatura existente concerniente al método metaheuristico Optimización Discreta de Enjambre de Partículas (DPSO o DiscreteParticleSwarmOptimization por sus siglas en ingles) y su aplicabilidad en la programación no lineal entera.	2
Desarrollar un algoritmo basado en la metaheuristica Optimización Discreta de Enjambre de Partículas para resolver el problema de la P – mediana.	3
Implementar en MATLAB los algoritmos basados en la Optimización Discreta de Enjambre de Partículas para la solución del problema de la P-mediana	3, Apendice B
Realizar experimentos numéricos que permitan medir la eficiencia del algoritmo desarrollado para la solución del problema de la P mediana.	4

1. Objetivos

1.1. Objetivo General

Elaborar un modelo de optimización basado en la metaheuristica Optimización de Enjambre de Partículas (PSO) para resolver el Problema de la P mediana (PMP).

1.2. Objetivos Específicos

- Recopilar, seleccionar y organizar la literatura existente concerniente al método metaheuristico Optimización Discreta de Enjambre de Partículas (DPSO o DiscreteParticleSwarmOptimization por sus siglas en ingles) y su aplicabilidad en la programación no lineal entera.

- Desarrollar un algoritmo basado en la metaheuristica Optimización Discreta de Enjambre de Partículas para resolver el problema de la P – mediana.

- Implementar en MATLAB los algoritmos basados en la Optimización Discreta de Enjambre de Partículas para la solución del problema de la P-mediana

- Realizar experimentos numéricos que permitan medir la eficiencia del algoritmo desarrollado para la solución del problema de la P mediana.

2. Marco Teórico

2.1. Optimización Matemática

Para estudiar los problemas de localización, se requiere modelar o construir una representación aproximada de los procesos y actividades que afectan la toma de decisiones respecto a la localización de instalaciones en un sistema de distribución.

Los modelos, en general, buscan capturar y abstraer la esencia de un evento u objeto del cual se indaga algo. Los modelos en Investigación de Operaciones sirven para mostrar relaciones que permiten describir problemas de la vida real, y así facilitar su análisis. Debido a que el conocimiento adquirido está basado en una representación, las conclusiones que se derivan del análisis del modelo corresponden al modelo utilizado y no al problema directamente. En este sentido, se espera que los modelos objetos de estudios tengan validez, y por esta razón se prefieren métodos robustos basados en la matemática para la modelación de problemas.[4]

Los modelos de optimización (también llamados Programas Matemáticos) buscan encontrar la mejor solución entre un espacio de soluciones. Estos modelos están constituidos básicamente por tres partes: *Las variables de decisión*, representan un conjunto de decisiones cuantificables las cuales se relacionan para derivar diferentes soluciones del problema; *las restricciones*, limitan los valores que pueden tomar las variables de decisión; y *la función objetivo*, mide el desempeño de la soluciones evaluadas de acuerdo a uno o varios criterios (Maximizar y/o Minimizar).

[4] HILLIER, Frederick S. y LIEBERMAN, Gerald J..Introduction to Operations Research.ed.7.Mc-Graw Hill.New York, 2001. p. 10-11.

2.2. Programación Entera y Optimización Combinatoria

Un problema general de optimización matemática se puede definir de la siguiente manera

$$\max f\ x\ ,\quad x \in S \subseteq \mathbb{R}^n \qquad (1)$$

Donde $\mathbb{R}^n$ es el conjunto de todos los vectores reales de n dimensiones y f es una función definida sobre S. El conjunto S es llamado conjunto *restringido* y f es llamada la función objetivo. Cada elemento $x \in S$ es llamado una solución factible de (1). Si existe un $x^o \in S$ tal que

$$\infty > f\ x^o\ \geq f\ x\ ,\quad \forall x \in S$$

entonces x^o es llamada la solución óptima de (1). El objetivo de la Optimización Matemática es definir si existe una solución óptima y encontrar esa solución o conjunto de soluciones óptimas.

Un Problema de Programación Entera (IP) es un problema de optimización Matemática en el cual

$$S \subseteq \mathbb{Z}^n \subset \mathbb{R}^n$$

donde $\mathbb{Z}^n$ es un conjunto de vectores de n dimensiones que solo toman valores enteros. En varios modelos, los valores de las variables enteras se usan para representar relaciones lógicas, y por lo tanto se restringen sus valores al conjunto de números $\{0,1\}$. Así surge un modelo especial de la programación entera llamada Programación Binaria Entera (BIP) en la cual $x \in \mathrm{B}^n$, donde B^n es el conjunto de todos los vectores binarios de n dimensiones.

Aunque no existe una definición general acerca de lo que es un problema de optimización combinatoria, varios de estos problemas pueden ser representados mediante BIP cuando se tratan con conjuntos y colecciones de conjuntos finitos. La siguiente es una definición genérica de un problema de optimización combinatoria.

Sea $N = \{1,...n\}$ un conjunto finito de elementos y sea $c = (c_1,...,c_n)$ un vector de orden n. Si F es un subconjunto del conjunto N, se define $c(F) = \sum_{j \in F} c_j$ .Si se supone una colección de particiones $\mathscr{F}$ de N, el problema de optimización combinatoria consiste en

$$\max(c(F) : F \in \mathscr{F})$$

es decir, encontrar el conjunto de vértices contenido en la partición $\mathscr{F}$ que maximice la función objetivo.

Es importante tener en cuenta que el objetivo de Optimización Matemática es desarrollar una teoría que lleve a la construcción de algoritmos que puedan tratar con instancias grandes de un problema dado y puedan ser implementados en computadores digitales de alta velocidad. Existen algoritmos que pueden ser utilizados a mano para resolver instancias pequeñas de un problema en un tiempo determinado. No obstante, ciertos aspectos de esos algoritmos se vuelven de vital importancia cuando las instancias del problema son grandes, como.

- ¿Está garantizada la terminación del Algoritmo?
- Si está garantizada, ¿existe una cota superior del número de computaciones?
- ¿La experiencia computacional es promisoria con respecto a la velocidad del algoritmo?
- ¿Se requiere un nivel de almacenamiento computacional razonable?

- ¿Si el algoritmo no termina en un tiempo determinado, se generan soluciones factibles con el algoritmo?

En el Apéndice D se muestra una revisión sencilla de los conceptos más importantes de complejidad computacional.

2.3. Problemas de Localización (generalidades)

Los problemas de localización, en su forma más general, se pueden describir de la siguiente manera. Un conjunto de clientes distribuidos espacialmente en un área geográfica demandan un cierto producto o servicio. La demanda de los clientes debe ser cubierta por una o varias instalaciones, y las instalaciones pueden operar en un marco de cooperación o competencia dependiendo del bien o servicio que sea requerido por el cliente. El proceso de decisión establece donde se deben ubicar las instalaciones en el territorio, tomando en cuenta los requerimientos de los clientes y las restricciones geográficas.[5]

Dentro de los problemas de localización se puede identificar tres elementos esenciales. Las *instalaciones*, que denotan un conjunto de objetos que serán localizados para proporcionar un servicio o producto. Las *localizaciones*, que se refieren al conjunto de posibles puntos para situar instalaciones. Los *clientes*, que son los usuarios de las instalaciones que demandan ciertos servicios o productos. El papel que desempeñan estos elementos varía entre los problemas de localización, y sirven para tipificar cada uno de los problemas.

2.3.1. Instalación o Facilidad

El termino instalación es utilizado en los problemas de localización para definir un objeto cuya posición espacial es optimizada teniendo en cuenta la forma como

[5]SCAPARRA, Maria Paola y SCUTELLA, Maria Grazia.Facilities, Locations, Customers: Building Blocks of Location Models. A survey. Universidad de Pisa, 2001. p. 2.

interactúa con objetos pre-existentes del problema. Algunos ejemplos de instalaciones son: almacenes, plantas productivas, escuelas, hospitales, centros comerciales, edificios públicos, etc. Las propiedades principales que caracterizan a las instalaciones son su: número, tipo y costo. Cuando el número de instalaciones está fijado *a priori*, se pueden distinguir entre: problemas de instalación simple y problemas multi-instalaciones.

Las instalaciones también se pueden clasificar de acuerdo al tipo. Esta propiedad especifica características tales como la capacidad, servicio y consideraciones sobre su estructura. Los modelos se pueden diferenciar de acuerdo a servicio sencillo y multi-servicios, basándose en la capacidad de la instalación para proveer uno o varios tipos de servicios o productos. Un ejemplo de un proveedor de servicio sencillo es un cadena de venta de comida rápida, y un ejemplo de proveedor multi-servicio son los hospitales que ofrecen múltiples servicios y productos de salud. Algunos problemas de localización admiten instalaciones donde se considera la capacidad de las instalaciones ilimitada, mientras que otros buscan la mejor ubicación con una producción o capacidad limitada.

Otra propiedad importante que permite diferenciar los problemas de localización según las instalaciones, es el costo. Existen dos tipos de costo: fijo y variable. El costo fijo está relacionado al costo de emplazamiento de las instalaciones y el costo variable es una función que se relaciona directamente con la distancia entre el cliente y la instalación.

2.3.2. Localización

El segundo elemento esencial en los modelos de localización, es el lugar físico en donde se va ubicar la instalación. El conjunto de ubicaciones es comúnmente

llamado espacio solución y se puede representar de manera continua, discreta o sobre redes.

- *Espacio Discreto*: En modelos de espacio discreto se especifica un conjunto finito de posibles lugares para ubicar las instalaciones. Los criterios de selección de los posibles lugares se hacen de acuerdo a factores geográficos y económicos. Ya que la solución óptima es alcanzada partiendo de un conjunto de candidatos, estos modelos son llamados modelos de selección.

- *Espacio Continuo*: Algunos modelos de localización definen los posibles lugares de ubicación según coordenadas en el espacio euclidiano (El caso más típico es en un espacio de dos dimensiones.) Ya que no se presume sitios candidatos para la ubicación, en cambio estos son generados por el algoritmo de búsqueda, estos modelos son llamados modelos de generación de sitios.

- *Representación en Redes*: Varias aplicaciones consideran problemas de localización en los que se tiene que operar utilizando una cierta infraestructura de red (red de carretera, red vial, red ferroviaria, red de telecomunicaciones, oleoductos ,etc.), comúnmente llamada grafo. El espacio de soluciones sobre redes también puede ser clasificado en: continua y discreta. En el primer caso, los posibles lugares de ubicación yacen tanto en los vértices como en cualquier punto sobre los arcos del grafo, mientras que en el segundo caso únicamente los vértices son candidatos para ubicar las instalaciones.

2.3.3. Demanda o Cliente

Los problemas de localización surgen de la necesidad de localizar centros para la satisfacción óptima de la demanda de un conjunto de clientes. Así pues, el tercer elemento esencial en los problemas de localización es la demanda o cliente, el cual se usa para denotar objetos que requieren accesibilidad a un servicio o demandan un producto. Cuando se analizan los problemas de localización, se debe interactuar con clientes, por lo tanto es necesario conocer su *distribución*, *demanda*, y *comportamiento.*

- Por *distribución* se asume que los clientes se distribuyen uniformemente en el espacio o en los vértices de una red.
- En el caso de la *demanda*, a cada cliente se le asigna un valor que expresa la cantidad de servicio o producto que requiere. La demanda no siempre es conocida de antemano y los análisis se hacen con demanda estocástica.
- La última característica del cliente es su *comportamiento*. En algunos casos, el cliente es libre de escoger desde cual instalación desea ser servido; el cliente siempre va a la instalación más cercana o utiliza otros criterios y preferencias para escoger la instalación.

2.3.4. Características del PMP estándar

La formulación estándar del PMP tiene las siguientes características:

- El problema está definido en una red o grafo, que consta de n nodos y r arcos.
- El costo fijo de emplazamiento de las instalaciones no se toma en cuenta pues se asume que es igual en cada uno de los vértices.
- Existe una restricción respecto al número máximo de instalaciones.
- La ubicación de las instalaciones se da únicamente en los vértices de la red o grafo.

- Las instalaciones no tienen restricciones de capacidad.
- Los clientes requieren una cantidad fija de servicios o productos y por comodidad escogen ser servidos por la instalación más cercana.

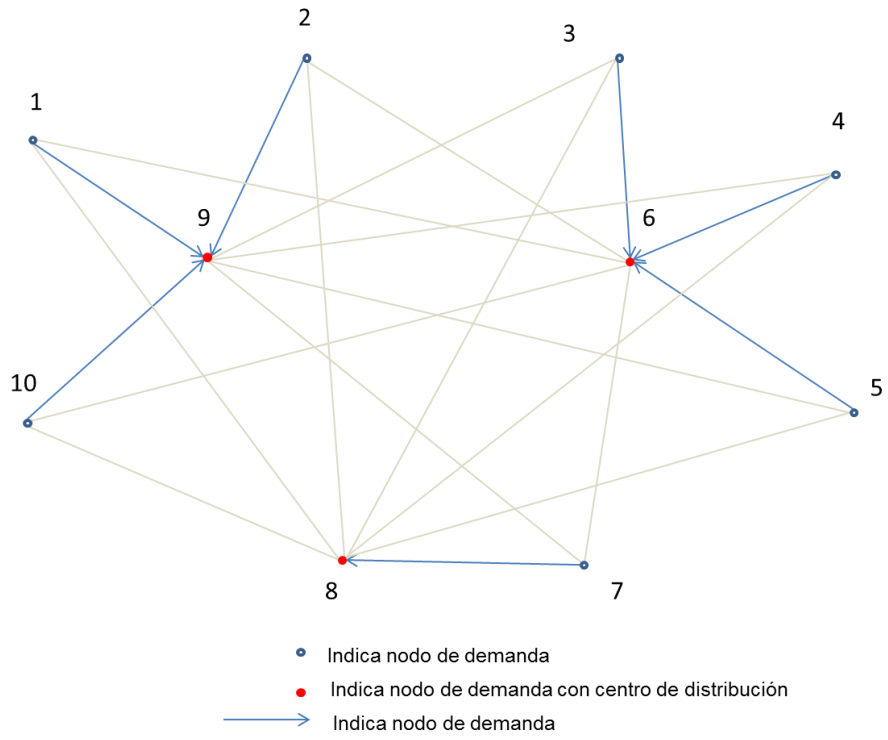
2.4. Descripción del Problema de la P-mediana

El problema de la P-mediana (P-median Problem o PMP por sus siglas en inglés) considera una situación como la siguiente. En un sistema de comunicación de redes N, como las redes de teléfono, existe usualmente un número de centros de distribución $S_1, S_2, \ldots, S_p$. Todo el tráfico (mensajes o información) que fluye dentro de la red debe pasar por un centro de distribución S para ser procesado y enviado a su respectivo destino. Un modelo para describir esta situación se puede hacer mediante un grafo G con pesos (números no negativos) asignados a sus vértices y aristas. El peso d_{ij} asignado a la arista ij de G representa la longitud del elemento. El peso h_i asignado al vértice v_i de G representa el número de cables (líneas) para procesar la información entre el vértice v_i y un centro de distribución dado. El problema consiste en encontrar la localización exacta de los centros de distribución $S_1, S_2, \ldots, S_p$ de tal manera que la cantidad de cable utilizado para manejar el flujo de información sea mínimo

Tanto los arcos como los vértices del grafo son posibles lugares para ubicar los centros de distribución, sin embargo, el Teorema de Hakimi (1963) demuestra la existencia de un conjunto formado únicamente por vértices que generan un resultado tan bueno como un conjunto de puntos situado en cualquier parte del grafo. En el anexo A esta contenida la demostración para $p = 1$ y $p > 1$.

A continuación se muestra una representación grafica de una solución del PMP. Los puntos de localización con coordenadas en un plano de dos dimensiones se muestran con colores azul y rojo. Para hacer ilustrativa la figura, se asume que cada par de puntos pueden ser conectados por medio de un cable homogéneo en línea recta, es decir, la distancia mínima para conectar dos puntos equivale a la distancia entre dos puntos en la métrica euclidiana. Así mismo, se asume que todos los puntos de localización requieren únicamente de un cable para transferir o recibir la información del centro de distribución más cercano, por lo tanto, la cantidad de cable requerido para conectar dos puntos dado que uno de los puntos es centro de distribución, es equivalente a la distancia euclidiana entre los dos puntos independientemente de cuál de los dos es el centro de distribución. El número de centros de distribución es 3 y el número de posibles puntos de localización es 10. En la figura se muestra la localización de los tres centros (puntos rojos); las líneas azules representan la asignación mínima entre los posibles centros de distribución (líneas grises) para cada vértice. En la figura aparece la configuración que utiliza la menor cantidad de cable posible cuando los puntos de localización de los centros de distribución son $(6, 8, 9)$.

Figura 1. Representacion Grafica de una solucion del PMP



Fuente: Autor

2.5. Formulación matemática del Problema de la P-mediana

Se denota por L el conjunto de instalaciones, m el número de instalaciones, el conjunto U representa n usuarios, y la matriz $D = d_{ij}$ de orden $n \times m$ representa las distancias (o costos incurridos) del nodo i (usuario) al nodo j (la instalación localizada) que se acarrea para satisfacer la demanda del usuario, para todo $j \in L, i \in U$. La función objetivo es la suma de las distancias y el criterio es minimizar, esto es

$$\min_{J} \sum_{i \in U} \min_{j \in J} d_{ij}$$

donde $J \subseteq L$ y $\left|J\right| = p$.

Además de la formulación combinatoria, el PMP se puede formular como un problema de programación entera (ReVelle y Swain, 1970) [25]. Se definen los dos siguientes conjuntos de variables: el primer conjunto de variable se usan para las instalaciones, mientras que el segundo hace referencia a los clientes: sean las variables de decisión: 1) $y_j = 1$, si una instalación es abierta en $j \in L$, y 0, de otra manera; 2) $x_{ij} = 1$, si el cliente i es servido por la instalación ubicada en $j \in L$, y 0, de otra manera. El problema de programación lineal entera (o ILP por sus siglas en inglés) se formula de la siguiente manera

$$\min \sum_i \sum_j a_i d_{ij} x_{ij}$$

Sujeto a:

$$\sum_j x_{ij} = 1, \forall i, \text{(2)}$$

$$x_{ij} \leq y_j, \forall i, j, \text{(3)}$$

$$\sum_j y_j = p, \text{(4)}$$

$$x_{ij}, y_j \in \ 0,1 \ \ \forall i, j, \text{(5)}$$

Las restricciones (2) aseguran que todos los usuarios están asociados a una y solo una instalación. La restricciones (3) se usan para prevenir que algún usuario sea servido desde algún punto que no sea instalación. La restricción (4) asegura que p vértices se seleccionen como medianas. Finalmente, el conjunto de restricciones (5) especifica que las variables de decisión son binarias.

Cuando el problema se define sobre un grafo G y el conjunto de nodos de demanda es igual al conjunto de nodos de posibles instalaciones ($L = U$), la mediana se define de la siguiente forma. Se considera una matriz simétrica de distancias D de G de orden $n \times n$, y la distancia entre cada par de vértices v_i, v_j está dada por d_{ij} .Para el caso del grafo con vértices de mismo peso, la mediana v_k será el vértice para el cual la suma de elementos correspondientes a cada fila en D es minimizada, esto es, si

$$d_j = \sum_{i=1}^{n} d_{ij}, \quad (j = 1,2,...,n)$$

Entonces v_k es la mediana del grafo si y solo si

$$d_k = \min(d_1, d_2, ..., d_n)$$

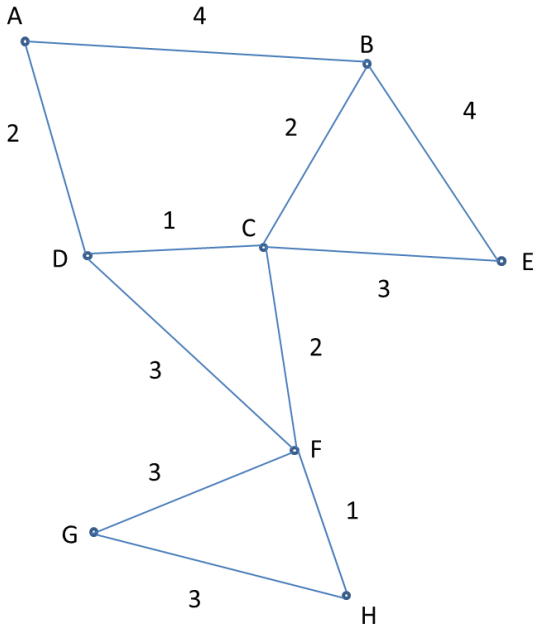
Para el caso en que los vértices tienen pesos diferentes, es necesario redefinir la matriz de distancias. Se considera una matriz diagonal de orden $n \times n$ con los pesos correspondientes a cada vértice ubicados en la diagonal. La matriz de distancias ponderada R de G esta definida de la siguiente manera

$$R = HD = \left[r_{ij} \right] = \left[h_i d_{ij} \right]$$

Donde r_{ij} representa la distancia ponderada asociada al vértice v_i si v_j es su único punto de abastecimiento.

A continuación se ilustra un ejemplo sobre una red de 8 vértices. La primera figura es el modelo grafico de una red de centros urbanos o puntos de demanda. Cada nodo representa la localización de un grupo de consumidores y esta rotulado por medio de letras mayúsculas. Las líneas representan los posibles caminos directos entre los puntos de demanda y tienen asociado un número que representa la distancia entre los puntos conectados. La tabla debajo del grafo contiene la cantidad de personas en cada centro urbano que representa el peso o demanda en cada vértice.

Figura 2. Red de centros urbanos sobre la cual se debe ubicar una instalación



Vertice	Poblacion (miles)
A	3
B	4
C	2
D	4
E	1
F	2
G	4
H	1

Fuente: Autor

La matriz de distancia D de orden 8×8 (Figura 3) contiene la mínima distancia d_{ij} entre todos los pares de vértices del grafo anterior. La distancia mínima entre dos vértices (shortest path problem) es la mínima suma de los pesos asignados de un conjunto de aristas tal que todos los elementos del conjunto conectan dos puntos del grafo, por ejemplo, los vértices A y E pueden conectarse a través de diferentes trayectos (conjuntos de aristas) como: $A-B-E$ (distancia = 4 + 4 = 8), $A-B-C-E$ (distancia = 4 + 2 + 3 = 9) y $A-D-C-E$ (distancia = 2 + 1 + 3 = 6). Sin embargo, solo el trayecto $A-D-C-E$ contiene la mínima distancia entre los nodos A y E. Puede existir el caso en que más de un trayecto entre dos puntos tengan la distancia mínima, no obstante la distancia mínima es un valor único. Las filas de la matriz representan los centros urbanos y las columnas las posibles medianas.

Figura 3. Matriz Simétrica de Distancias Mínimas para el grafo de la Figura 3

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{bmatrix} 0 & 4 & 3 & 2 & 6 & 5 & 8 & 6 \\ 4 & 0 & 2 & 3 & 4 & 4 & 7 & 5 \\ 3 & 2 & 0 & 1 & 3 & 2 & 5 & 3 \\ 2 & 3 & 1 & 0 & 4 & 3 & 6 & 4 \\ 6 & 4 & 3 & 4 & 0 & 5 & 8 & 6 \\ 5 & 4 & 2 & 3 & 5 & 0 & 3 & 1 \\ 8 & 7 & 5 & 6 & 8 & 3 & 0 & 3 \\ 6 & 5 & 3 & 4 & 6 & 1 & 3 & 0 \end{bmatrix} \end{array} = [d_{ij}]$$

Fuente: Autor

Debido a que las distancias son ponderadas de acuerdo a la cantidad de demanda, se multiplica cada fila de la matriz por el correspondiente valor de la población para obtener la matriz de distancias ponderadas. Para obtener este resultado, se forma una matriz diagonal H con la tabla de poblaciones y se multiplica por la matriz D como se muestra en la siguiente figura.

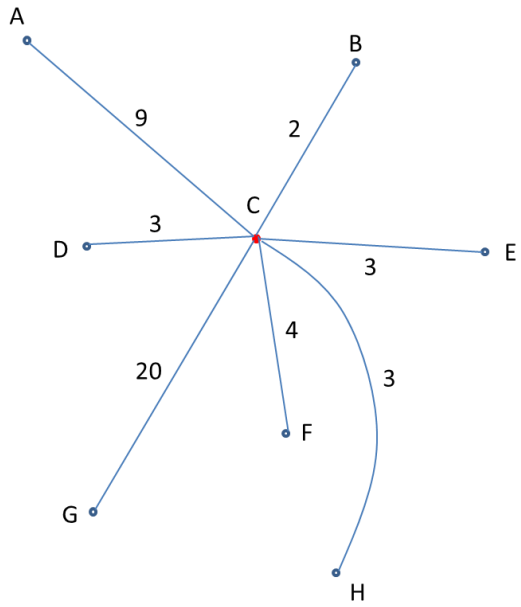
Figura 4. Matriz Ponderada de Distancias Mínimas para el grafo de la Figura 3

$$\mathrm{R} = \begin{bmatrix} 3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} 0 & 4 & 3 & 2 & 6 & 5 & 8 & 6 \\ 4 & 0 & 2 & 3 & 4 & 4 & 7 & 5 \\ 3 & 2 & 0 & 1 & 3 & 2 & 5 & 3 \\ 2 & 3 & 1 & 0 & 4 & 3 & 6 & 4 \\ 6 & 4 & 3 & 4 & 0 & 5 & 8 & 6 \\ 5 & 4 & 2 & 3 & 5 & 0 & 3 & 1 \\ 8 & 7 & 5 & 6 & 8 & 3 & 0 & 3 \\ 6 & 5 & 3 & 4 & 6 & 1 & 3 & 0 \end{bmatrix} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \\ \sum \end{array} \begin{array}{c} \begin{array}{cccccccc} A & B & C & D & E & F & G & H \end{array} \\ \begin{bmatrix} 0 & 12 & 9 & 6 & 18 & 15 & 24 & 18 \\ 16 & 0 & 8 & 12 & 16 & 16 & 28 & 20 \\ 6 & 4 & 0 & 2 & 6 & 4 & 10 & 6 \\ 8 & 12 & 4 & 0 & 16 & 12 & 24 & 16 \\ 6 & 4 & 3 & 4 & 0 & 5 & 8 & 6 \\ 10 & 8 & 4 & 6 & 10 & 0 & 6 & 2 \\ 32 & 28 & 20 & 24 & 32 & 12 & 0 & 12 \\ 6 & 5 & 3 & 4 & 6 & 1 & 3 & 0 \\ 84 & 73 & 51 & 58 & 104 & 65 & 103 & 80 \end{bmatrix} \end{array}$$

Fuente: Autor

Para resolver el problema de una mediana se debe encontrar el vértice en el cual se minimiza la suma de distancias mínimas a los demás vértices. Por ejemplo, si se escoge el vértice A como solución actual, la suma de las distancias mínimas desde A a los demás vértices es equivalente a la suma de los elementos de la columna A (0 + 16 + 6 + 8 + 6 + 10 + 32 + 6 = 84). Para encontrar la solución óptima se debe encontrar la columna en la cual sea mínima esta suma. Debajo de la matriz R se muestra la suma de los elementos de las columnas y se encuentra que la solución óptima es ubicar la mediana en el vértice C. La siguiente figura muestra una representación gráfica de la asignación de vértices en la solución óptima $\{C\}$.

Figura 5. Grafica de asignación de clientes de la solución óptima para $p=1$



$$f(\{C\}) = 9+8+0+4+3+4+20+3 = 51$$

Fuente: Autor

Los resultados obtenidos para $p=1$ se pueden generalizar para $p>1$ de la siguiente manera. Se denota por V_p a un subconjunto de exactamente p vértices de G. Para un grafo de n vértices existen $\binom{n}{p}$ posibles subconjuntos del mismo tamaño, y se denota a cada uno de estos subconjuntos por ${V_p}^m \left[m = 1,2,\ldots,\binom{n}{p} \right]$.

Para cada subconjunto se construye una matriz ${R_p}^m$ agrupando todas la columnas de R correspondientes a los vértices en ${V_p}^m$. ${R_p}^m$ es de orden $n \times p$ y contiene el conjunto de instalaciones ${V_p}^m$ y las distancias ponderadas a cada nodo de

demanda. Si se asume que las instalaciones no tienen restricciones en la capacidad de abastecimiento, cada destino v_i es servido por la instalación v_k de V_p^m para el cual r_{ik} es la distancia ponderada mínima. Esto es,

$$r_{ik} \leq r_{ij} \quad \forall\, v_k, v_j \in V_p^m$$

La suma total de distancias mínimas ponderadas r_m para el conjunto de instalaciones V_p^m es la suma de los mínimos elementos de cada fila de la matriz R_p^m:

$$r_m = \sum_{i=1}^{i=n} r_{ik}$$

donde k se refiere a la instalación para la cual r_{ij} es minimizada para un nodo de demanda dado. El conjunto de medianas del grafo G es definido como un conjunto de vértices V_p^{m*} tal que

$$r_{m*} = \min\left(r_1, r_2, ..., r_{\binom{n}{p}}\right)$$

es decir,

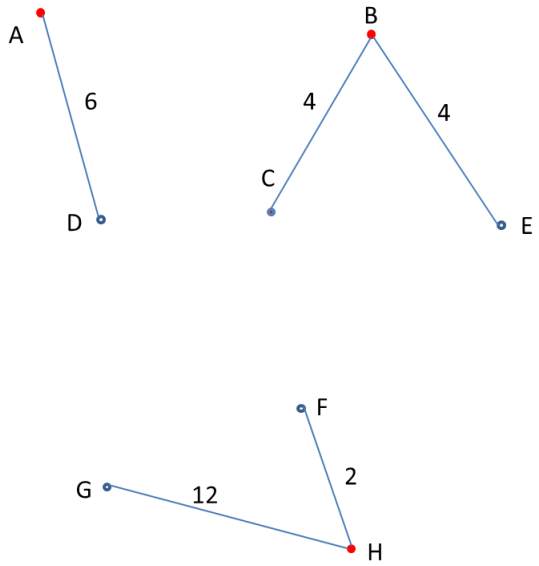
$$r_{m*} \leq r_m \quad \left(m = 1,2,..., \binom{n}{p}\right)$$

A continuación se muestra un ejemplo con $p = 3$ sobre el grafo y la matriz de distancias ponderadas del ejemplo utilizado en $p = 1$. Para resolver instancias

donde $p > 1$ la evaluación de una solución es un poco diferente. Por ejemplo, se escoge el conjunto de vértices $V_3 = \{A, B, H\}$ como solución actual. Para evaluar esta solución se requiere únicamente información (distancias) de las columnas correspondientes a los vértices de V_3, ya que ningún cliente (fila) puede ser asignado a algún nodo que no pertenezca al conjunto de medianas (conjunto de restricciones (3) del modelo LP). Se forma una sub-matriz R_3 con la columnas de correspondientes a V_3 y se rotula el mínimo elemento de cada fila de la sub-matriz. Este proceso de rotulamiento asegura que cada cliente sea asignado a una sola instalación: la instalación más cercana (conjunto de restricciones (1) del modelo LP). Finalmente se suman los elementos rotulados y se obtiene la evaluación de esa solución como se muestra en la siguiente figura.

Figura 6. Submatriz de distancias ponderadas y grafico de asignación para $V_3 = \{A, B, H\}$

$$R_3 = \begin{array}{c|ccc} & A & B & H \\ A & 0 & 12 & 18 \\ B & 16 & 0 & 20 \\ C & 6 & 4 & 6 \\ D & 8 & 12 & 16 \\ E & 6 & 4 & 6 \\ F & 10 & 8 & 2 \\ G & 32 & 28 & 12 \\ H & 6 & 5 & 0 \end{array}$$



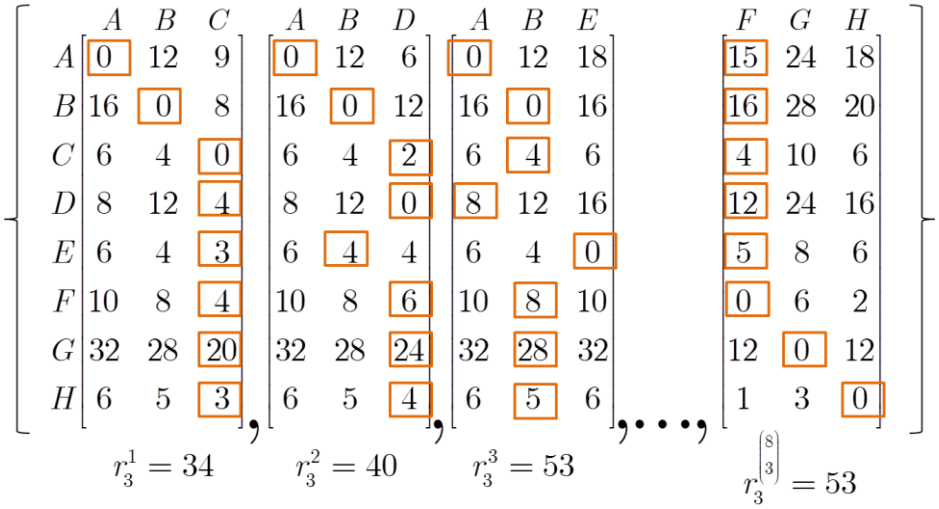
$$r_3 = f(V_3) = 0 + 0 + 4 + 8 + 4 + 2 + 12 + 0 = 30$$

Fuente: Autor

El problema de la P-mediana consiste en encontrar un método eficiente para llegar a la solución óptima y dar prueba verídica de ese hecho. Se entiende por solución optima al conjunto de vértices cuya evaluación es menor que cualquier otro conjunto de vértices en la colección de conjuntos o espacio de soluciones. Debido a que el tamaño del ejemplo es pequeño, un algoritmo efectivo para resolver el problema (incluso mejor que algunas heurísticas sofisticadas) y dar prueba de la optimalidad (contrario a cualquier metodo heuristico) es Enumeración Exhaustiva, que pertenece a la clase de Métodos Exactos. Como su nombre lo sugiere, esta técnica consiste en evaluar cada una de las $\binom{n}{p}$ soluciones y decidir cuál o cuáles suministran el menor valor. A continuación se muestra una ilustración compacta del espacio de soluciones del problema, donde cada solución está representada

por la sub-matriz de distancias ponderadas R_3^m. Arriba de cada sub-matriz se encuentra el conjunto de puntos que conforman las instalaciones de esa solución, y en la parte de abajo se encuentra la evaluación de la función objetivo. Cabe notar, que para hacer una revisión exhaustiva del espacio de soluciones, se requiere evaluar $\binom{8}{3} = 56$ sub-matrices de distancias.

Figura 7. Representación gráfica del espacio de soluciones para $n = 8$ y $p = 3$

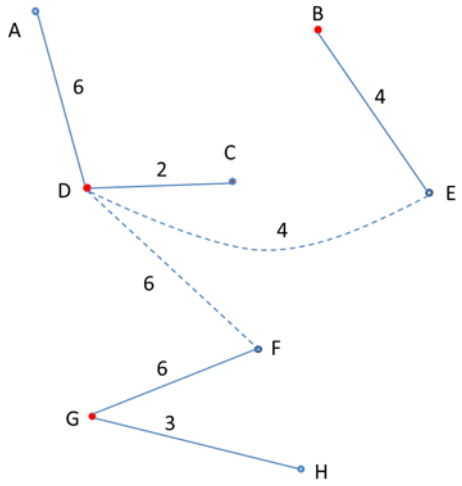


Fuente: Autor

La siguiente figura ilustra la sub-matriz de distancias y el grafico de asignación de la solución óptima.

Figura 8. Submatriz de distancias ponderadas y grafico de asignación para la solución óptima

$$R_3^* = \begin{array}{c|ccc} & B & D & G \\ A & 12 & \boxed{6} & 24 \\ B & \boxed{0} & 12 & 28 \\ C & 4 & \boxed{2} & 10 \\ D & 12 & \boxed{0} & 24 \\ E & \boxed{4} & \boxed{4} & 8 \\ F & 8 & \boxed{6} & \boxed{6} \\ G & 28 & 24 & \boxed{0} \\ H & 5 & 4 & \boxed{3} \end{array}$$



$$r_3^* = f(V_3) = 6+0+2+0+4+6+0+3 = 21$$

Fuente: Autor

La solución óptima $V_3^* = \{B, D, G\}$ tiene diferentes formas para hacer la asignación de la demanda, esto se debe a que un grupo de instalaciones puede ser equidistante a un cliente en particular, por lo tanto un intercambio de asignación no afecta la función objetivo. En la figura anterior se muestra como el cliente E y F pueden ser asignados a dos instalaciones distintas sin afectar la evaluación de la solución.

2.6. Algoritmos Exactos

Los algoritmos exactos buscan encontrar la solución óptima del problema y dar prueba verídica de ese hecho, sin embargo, la aplicación de estos algoritmos en instancias grandes es inviable debido a la cantidad exorbitante de tiempo de ejecución que requieren.

Existen diferentes métodos exactos además de la enumeración exhaustiva, como la búsqueda Branch-and-Bound (B&B), relajación lineal y métodos basados en relación lagrangeana.

El metodo Branch-and-Bound consiste en dividir un problema en subproblemas (branching) y suministrar cotas inferiores para cada uno de estos subproblemas (bounding). Posteriormente, bajo un conjunto de criterios dado se decide en cual subproblema se va intensificar la búsqueda y cuáles pueden ser descartados de este proceso (fathoming).

El-Shaieb (1973) [4] presenta dos formas de calcular cotas inferiores para ser usados en un algoritmo branch-and-bound; la primera de ellas requiere un mayor número de iteraciones y menor tiempo computacional por iteración para alcanzar la solución óptima, por tanto es eficiente cuando el número de las instalaciones es pequeño, por el contrario, la segunda requiere menos iteraciones, mayor tiempo computacional por iteración, y es más eficiente cuando el número de instalaciones es grande. Ambos métodos son utilizados en un arbol Branch-and-Bound que comienza con un conjunto vacío de asignación $S:D$, conformado por los conjuntos S (de instalaciones) y D (de nodos de demanda o puntos de destino). En cada iteración un nodo es añadido a cualquiera de los conjuntos dependiendo de cuál reporta la mejor cota inferior, este proceso es repetido hasta que S es igual p o D igual a $n-p$.

Jarvinen *et al* (1973) [12] propone un método para hallar cotas inferiores en un algoritmo Branch-and-Bound. Inicialmente todos los n nodos contienen medianas y en cada iteración se remueven medianas hasta $n-p$. Para calcular la mejor cota inferior en cada iteración, se dividen los nodos en dos grupos: r nodos ($0 \leq r \leq n-p$) pertenecen al grupo de nodos de demanda definitivos, y $n-r$ nodos son posibles medianas. Para cada nodo que pertenece a r se selecciona la menor distancia entre este nodo y una posible mediana (un elemento del grupo de posibles medianas). Para cada nodo que pertenece a $n-r$ se selecciona la

segunda menor distancia entre este nodo y un nodo estrictamente diferente dentro del grupo $n-r$. Posteriormente se suman todos los valores de los nodos de los dos grupos. En cada iteración se evalúan todos los posibles conjuntos que se pueden obtener de incrementar en un nodo de demanda el grupo r y se escoge el que representa un menor incremento.

Los primeros trabajos que hicieron uso de la relajación de programación lineal fueron desarrollados por Revelle y Swain (1970) y Garfinkel *et al* (1974). En estos algoritmos las restricciones de integralidad positiva se cambian por restricciones de no negatividad ($x_{ij} \geq 0, \forall i, j$). Aunque en la mayoría de casos los resultados son 0 o 1, existe la posibilidad de que las variables tomen valores fraccionarios y por ende la solución encontrada sea no factible. Otro problema con este tipo de formulación es el tamaño excesivo del problema de programación lineal correspondiente.

Los métodos lagrangeanos suministran cotas inferiores a partir de las cuales se puede obtener soluciones factibles del problema.Los métodos lagrangeanos se basan en la formulación del problema por medio de programación lineal entera (ILP) descrita anteriormente. La idea de este método es que el problema original, el cual contiene un conjunto de restricciones que se consideran “complicadas”, puede ser tratado como un nuevo problema que es más fácil de resolver. Por medio de la dualizacion de dicho conjunto de restricciones, se crea un problema dual cuya información será útil para encontrar la solución óptima del problema original. Este procedimiento se llama Relajación Lagrangeana y consiste en eliminar las restricciones que hacen intratable el problema y añadirlas en la función objetivo con unos pesos asignados llamados Multiplicadores de Lagrange. El problema primal y dual no mantienen principio de dualidad fuerte en programación discreta, esto implica que el valor optimo del problema dual no es necesariamente el valor optimo del problema primal. Luego, en lugar de

$$Z_D = Z_{IP}$$

(donde Z_D es el valor optimo del problema dual y Z_{IP} es el valor optimo del problema primal), solo se mantiene el principio de dualidad débil, el cual establece que

$$Z_D \leq Z_{IP}$$

Como no en todas las instancias el valor óptimo del problema dual y primal coincide se busca encontrar las cotas inferiores más grandes de manera que el intervalo de dualidad se cierre, y así alcanzar buenas aproximaciones del valor óptimo primal.

Usualmente las restricciones que aseguran que cada nodo de demanda está asignado a una y sola una instalación se relaja en la función objetivo, pues esta relajación produce buenas cotas inferiores. La solución encontrada puede no ser factible pues la restricciones mencionadas pueden ser violada, sin embargo, factibilidad puede ser alcanzada asignando los nodos a la instalación más cercana.

En los metodos lagrangeanos los siguientes pasos se repiten en cada iteración. 1) Se definen los valores de los multiplicadores u_i,(en la primera iteración toman valor cero), 2)se resuelve el modelo lagrangeano; se encuentran los valores de x_{ij} y y_i para un u dado y utilizando un algoritmo se vuelve factible esta solución dada, así mismo se actualiza el valor de la función objetivo después de ejecutar este algoritmo (este último paso puede ser implementado ocasionalmente en lugar de todas las iteraciones), finalmente,3)se ajustan los valores de los multiplicadores usando subgradiente de optimización. El valor más grande del problema dual (sobre todas las iteraciones) representa un límite inferior del problema de optimización original. Narula *et al* (1976) [22] proponen utilizar los métodos

lagrangeanos como subrutina de *Bounding* en el algoritmo exacto Branch-and-Bound.

Galvao y Raggi (1989) [7] desarrollaron un método de tres fases para resolver el problema Uncapacitated Facility Location Problem (UFLP) el cual tiene contiene como casos especiales al P-median Problem y Simple Plant Location Problem. La primer fase es un algoritmo primal-dual que consiste en generar cotas superiores por medio de un procedimiento sobre la formula primal (vertex substituiton) y cotas inferiores por medio de un procedimiento sobre la formulación dual (dual ascent procedure). Si el intervalo de las cotas no se cierra, se procede al paso dos en el cual se aplica subgradiente de optimización al problema dual lagrangeano. Si en el segundo paso tampoco se cierra el intervalo con la solución dual (lagrangeana) se procede a la tercer fase y se aplica un algoritmo Branch-and-Bound. El algoritmo resuelve óptimamente problemas de hasta 200 nodos en tiempos de computación razonable.

2.7. Métodos Heurísticos

Para la mayoría de problemas de optimización no existe un algoritmo con complejidad polinomial que encuentre la solución óptima a dicho problema. Además, la cardinalidad del espacio de búsqueda de estos problemas suele ser muy grande, lo cual hace inviable el uso de algoritmos exactos ya que la cantidad de tiempo que necesitaría para encontrar una solución es inaceptable. Debido a estos dos motivos, se necesita utilizar algoritmos aproximados o heurísticos que permitan obtener una solución de calidad en un tiempo razonable. El término heurística proviene del vocablo griego *heuriskein*, que puede traducirse como encontrar, descubrir o hallar.

La heurística constructiva Greedy, tambien conocida como Método de Adición de Vértices, fue propuesta inicialmente por Kuhen y Hamburger en un problema similar llamado Simple Plant Location Problem (1963) [17] y consiste en adicionar recursivamente vértices al conjunto de medianas. Inicialmente, se resuelve el problema para $p = 1$, y luego se añaden las instalaciones que representan los menores incrementos en la función objetivo hasta que el tamaño deseado de p sea alcanzado.

Para problemas de programación no lineal que poseen estructuras complejas, un método heurístico común es ejecutar un procedimiento de mejora local (Busqueda Local o First Improvement). Tal procedimiento comienza con una solución de prueba inicial y, en cada iteración, busca en la vecindad de la solución de prueba para tratar de encontrar una mejor solución que la actual. Este proceso continúa hasta que no se pueda encontrar una solución mejorada en la vecindad de la solución de prueba actual.

El primer algoritmo heurístico de Mejora Local para el problema de la P – mediana fue propuesto por Maranzana (1964) [18].La técnica de Partición de Vértices, como es conocida en la literatura, consiste en hacer una partición de los nodos en p grupos (asignando cada nodo a su instalación más cercana) y posteriormente se determina el “centro de gravedad” en cada grupo para ajustar la posición de las instalaciones. Dicho procedimiento se itera hasta que las instalaciones dejan de cambiar de posición.

Uno de los métodos más citados en la literatura, debido a su robustez comparado con otros algoritmos de Búsqueda Local, es la Substitución de Vértices (Vertex Substitution) de Teitz y Bartz (1964) [29]. Este método será estudiado con más detalle en la siguiente sección.

2.8. Metaheurística

El término metaheurística o meta-heurística fue acuñado por F. Glover en el año 1986. Con este término, pretendía definir un “procedimiento maestro de alto nivel que guía y modifica otras heurísticas para explorar soluciones más allá de la simple optimalidad local”.

Las metaheurísticas son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son efectivos. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los procedimientos estadísticos

Según el teorema NFL (No Free LucnhTheorem), los métodos generales de búsqueda, entre los que se encuentran las metaheurísticas, se comportan exactamente igual cuando se promedian sobre todas las funciones objetivo posibles, de tal forma que si un algoritmo A es más eficiente que un algoritmo B en un conjunto de problemas, debe existir otro conjunto de problemas de igual tamaño para los que el algoritmo B sea más eficiente que el A. Esta aseveración establece que, en media, ninguna metaheurística (algoritmos genéticos, búsqueda dispersa, búsqueda tabú, etc.) es mejor que la búsqueda completamente aleatoria. Una segunda característica que presentan las metaheurísticas es que existen pocas pruebas sobre su convergencia hacia un óptimo global; es decir, que a priori no se puede asegurar ni que la metaheurística converja ni la calidad de la solución obtenida.[6]

[6]GUTJAHR, Walter J. Stocasthic Search in Metaheuristics. En: Handbook of Metaheuristics[On line]. ed. 2.GENDREAU, Michel y y POTVIN, Jean-Yves. p. 587-589. Google Books, 2013 [Citado el 17 de Marzo de 2013: 18:00:00]. Disponible en: http://books.google.com/

A pesar de estos aparentes problemas, la realidad es que el comportamiento experimental de la mayoría de las metaheurísticas es extraordinario, convirtiéndose para muchos problemas difíciles de resolver en la única alternativa factible para encontrar una solución de calidad en un tiempo razonable.

Las metaheuristicas se clasifican en dos grupos: técnicas que se basan en la trayectoria y técnicas que se basan en poblaciones.

2.8.1. Metaheuristicas basadas en la trayectoria

Las técnicas que se basan en la trayectoria, guían su búsqueda partiendo del concepto de vecindad, intentando mejorar el desempeño de la solución incumbente y explorando nuevas posibilidades para no quedar atrapado en óptimos locales. Un ejemplo de estas técnicas aplicado a la P-mediana fue propuesto por Choyoshi y Galvao (2000)[3]. El algoritmo combina la heurística robusta de Teitz y Bartz con Recocido Simulado para aceptar soluciones que puedan llevar la búsqueda a vecindarios más promisorios.

Las técnicas de esta clase que mejor se desempeñan en resolver el PMP son Variable Neighborhooh Search (Mladenovic, 1997) , Heuristic Concentration (Rosing y Revelle, 1997), Busqueda Tabu(Rolland *et al*,1996) GRASP-PathRelinking(Resende y Werneck, 2003).

En VariableNeighborhoohSearch (1997) [10] se define una estructura de vecindario variable a través del método $\lambda - \mathrm{interchange}$. El VNS procede inicialmente con un método descendente (Variable Neighborhood Descent) que efectúa la búsqueda de un mínimo local a través de varios vecindarios (para el PMP los autores utilizan la heurística Fast Interchange propuesta por Whitaker en 1964), y posteriormente diversifica la búsqueda explorando vecindarios distantes a la solución incumbente. El cambio de estructura de vecindarios y el método de

búsqueda descendente requieren la implementación de una métrica para definir la distancia entre dos soluciones. Así pues, sea $X = \{x \mid x = \{m_1, m_2, ..., m_p\}, m_i \in L, |L| = m\}$ el espacio de soluciones del problema, se dice que la distancia entre dos puntos x_1 y x_2 $(x_1, x_2 \in X)$ es igual a k si y solo si las dos soluciones difieren en k instalaciones. Dado que X es un conjunto conformado por subconjuntos, la función (simétrica) de distancias ρ se puede definir como

$$\rho(x_1, x_2) = |x_1 \setminus x_2| = |x_2 \setminus x_1|, \forall x_1, x_2 \in X$$

donde ρ es una función métrica discreta sobre X y $|x_1 \setminus x_2|$ representa la distancia entre los puntos x_1 y x_2 que pertenecen al espacio métrico de soluciones.

La construcción de las estructuras de vecindario es inducida por la métrica ρ, es decir, el vecindario N_k de una solución x, es el conjunto de soluciones que resulta del intercambio de k nodos en el conjunto actual de instalaciones. El VNS básico es un método descendente aleatorio del mejor primero (First Improvement). Se puede convertir fácilmente en un método descendente-ascendente modificando la regla de decisión de movimiento; se asigna $x \leftarrow x''$ con una probabilidad, incluso si la solución en juicio es peor que la incumbente. De igual manera se puede transformar en un método Best Improvement; se hace un movimiento al mejor vecindario k entre el conjunto de vecindarios $k_{\max}$. Otras variaciones del VNS básico incluyen encontrar la mejor solución x' (paso 2a) entre ℓ soluciones aleatorias del k^{th} vecindario, y/o introducir $k_{\min}$ y k_{step}, dos parámetros que controlan el proceso de cambio de estructura de vecindario. El algoritmo fue utilizado para resolver instancias de hasta 500 nodos y 25 medianas

y en todos los casos encontró la mejor solución conocida en tiempos de computación razonable .

La técnica Heuristic Concentration (1997) [27] busca aprovechar características de las mejores soluciones encontradas por medio de un método de Búsqueda Local y así encontrar mejores aproximaciones a la solución óptima. En la primera fase de HC, Vertex Substitution se utiliza para generar soluciones óptimas variadas partiendo de diferentes soluciones iniciales, las mejores corridas serán utilizadas para formar un Conjunto de Concentración. En la fase dos de HC, el grupo de nodos potenciales para ubicar instalaciones se limita al Conjunto de Concentración definido en la primera fase, y se resuelve el subproblema utilizando métodos exactos. Los resultados de 90 instancias con redes de hasta 300 nodos de demanda y 50 instalaciones muestran que HC alcanzo la solución óptima el 78.9% de las veces. Un aspecto importante tener en cuenta es que el algoritmo no tiene buen desempeño para instancias pequeñas del problema, si se eliminan los resultados para $p = 5$ de la tabla de resultados, la tasa de éxitos del algoritmo aumenta a 87.7%.

La metaheuristica Busqueda Tabu (1996) [26] es utilizada como marco para crear un algoritmo eficiente para resolver el PMP. El esquema de movimiento de 1-intercambio de nodos por iteración crea un vecindario definido por el conjunto de instalaciones que resulte de añadir o remover una instalación de este conjunto. Una lista tabú de corta memoria y una función de larga memoria que penaliza los movimientos más frecuentes, son implementadas para evitar óptimos locales y diversificar la búsqueda en distintos vecindarios. Así mismo, un criterio de aspiración es utilizado para asegurar que los movimientos restringidos o tabú no impiden alcanzar soluciones de calidad. Una característica importante de la búsqueda tabú, oscilación estratégica, permite alcanzar otros vecindarios y evitar óptimos locales a través de soluciones no factibles. Este algoritmo es comparado con Heuristic Concentration en 100 instancias (con tamaños desde 13 hasta 500

nodos), y los resultados muestran mejoras significativas tanto en el tiempo de computación como en la calidad de la solución encontrada.

La técnica GRASP-PathRelinking (2003) [24] es un hibrido de dos técnicas efectivas para resolver problemas combinatorios. La técnica GRASP (Greedy Randomized Adaptive Search Procedure) es un método de múltiples corridas que consiste en construir multiples soluciones con la técnica Randomization Greedy (a diferencia de la técnica clásica Greedy, en RGreedy se escoge aleatoriamente entre un grupo de los mejores candidatos, es decir los que menos incrementan la función objetivo), y mejorar la calidad de estas soluciones por medio de un método de búsqueda local. Los métodos tradicionales toman la mejor solución de entre todas las corridas como solución óptima, no obstante esta heurística aplica un método de intensificación para mejorar la calidad de los resultados. Se crea un conjunto con las mejores soluciones encontradas llamado Conjunto Elite, en cada iteración la solución encontrada por medio de la búsqueda local es combinada con una de las soluciones del Conjunto Elite por medio de una técnica llamada Path-Relinking. En la fase de Post-optimizacion del algoritmo, las soluciones elites son combinadas, y las resultantes son añadidas a un nuevo Conjunto Elite de soluciones, dicho procedimiento se itera hasta que no exista evidencia de mejoras en la función objetivo. Los resultados de experimentos con problemas de OR-library y TSP-library entre otras, muestran que el algoritmo encontró solucionas con a lo más 1% de diferencia con respecto a las mejores cotas inferiores encontradas.

2.8.2. Metaheuristicas basadas en Poblaciones

Las técnicas basadas en poblaciones parten de un conjunto de soluciones llamado población inicial y sobre ese conjunto se aplican diversos operadores con el objetivo de mejorar el desempeño de las soluciones y diversificar los espacios de búsqueda. Los métodos más utilizados dentro de esta categoría fueron inspirados en la naturaleza, tales como Optimización de Enjambre de Partículas (PSO),

Algoritmos Genéticos (GA), Estrategias de Evolución (ES), Redes Neuronales (NN) y Optimizacion de Colonia de Hormigas (ACO); sin embargo, existen técnicas que operan de manera similar y no fueron inspirados en la naturaleza como Busqueda Dispersa (SS).

Cierto grupo de algoritmos inspirados en la naturaleza, se basan específicamente en la Teoría de la Evolución Biológica, en la cual se entiende evolución como un proceso de optimización donde el objetivo es mejorar la capacidad de un organismo (o sistema) para sobrevivir en entornos dinámicamente cambiantes y competitivos. Este proceso se adaptación de los individuos se hace a través de la reproducción, en el cual se espera que los padres transmitan las mejores características a sus hijos. Cuando los padres transmiten malas características a sus hijos, estos últimos son menos aptos para sobrevivir y tienden a desaparecer. [7]Un ejemplo de este proceso en la vida real, son ciertas especies de aves en la cual los polluelos que nacen primero se alimentan mejor y con el tiempo se deshacen sus hermanos más débiles botándolos del nido. Así mismo, cuando los hijos heredan genes nocivos de sus padres, como enfermedades o deformaciones, en muchos casos estos individuos no son compatibles con la vida y desaparecen. Las técnicas que se basan en estos procesos evolutivos se llaman métodos de Computación Evolutiva (EC).

Debido a que varios algoritmos pueden ser descritos de esta manera, al considerar el diseño de un nuevo algoritmo de EC para la solución de un problema particular de optimización, se deben tener en cuenta las siguientes características principales:

- Un proceso de *evolución* que permite definir los cambios en la población a cada generación o de manera continua

[7] ENGELBRETCHT, Andries P..Computational Intelligence.An Introduction.ed.2.WILEY.Inglaterra, 2005. p. 127-128

- Una definición de *vecindad* que permite conocer el modo como los individuos intercambian información.
- Una medida de la *factibilidad* de la solución obtenida, lo cual permite determinar cuál es buena, óptima o inadecuada.
- Un mecanismo de *intensificación*, que realiza mejoras sobre un individuo sin tener en cuenta la información suministrada por otros individuos, permitiendo la intensificación de la búsqueda sobre algunas regiones del espacio.
- Un mecanismo de *diversificación*, el cual permite evitar convergencia hacia puntos óptimos locales.

Los Algoritmos Genéticos (GA) son un grupo de técnicas que modelan la evolución genética, en la cual los individuos son representados usando genotipos. Cada individuo es un grupo de *genes* o *alelos,* llamado *cromosoma*, el cual contiene información respecto a una solución del problema. Los operadores principales de los Algoritmos Genéticos son la Selección de Individuos (sobrevivencia del más apto) y la combinación de cromosomas a través de operadores genéticos, como la Recombinación (crossover) y la Mutación (mutation).[8]

Hosagge y Goodchild (1986) [11] fueron los primeros en aplicar los algoritmos genéticos al PMP. La representación de cada cromosoma es una cadena binaria de n celdas que tienen 1 donde se sitúa una instalación. La cardinalidad del espacio genotípico es de 2^n , siendo muchas de estas soluciones no factibles. El algoritmo fue aplicado a 100 instancias aleatorias de 20 vértices y 3 medianas, y entre el 70% y 90% de los casos se encontró la solución óptima.

[8] Ibid,.p 143-156.

Erkun (2003) [5] aplica un algoritmo genético eficiente basado en una más compacta, en la cual cada cromosoma tiene p celdas, y cada celda tiene un número entero que representa el índice del vértice en el cual se sitúa una instalación. El algoritmo fue utilizado para resolver instancias que varían entre 100 y 1000 nodos, y en 85% de los casos alcanzo una solución con diferencias de al menos 0.1% con la solución optima real.

2.9. Inteligencia de Enjambres (IS)

De acuerdo a Marco Dorigo y Mauro Birattari, de la Université Livre de Bruxelles, Bélgica, se llama Inteligencia de Enjambres a “la disciplina que trata con sistemas naturales y artificiales compuestos de muchos individuos que se coordinan utilizando control descentralizado y auto-organización. En particular, la disciplina se enfoca en las conductas colectivas que resultan de las interacciones locales de los individuos entre sí y con el ambiente".[9]

Los estudios de animales e insectos sociales han dado como resultado distintos modelos de inteligencia de enjambre. Los sistemas biológicos de enjambre que han inspirado modelos computacionales incluyen hormigas, termitas, abejas, arañas, bancos de peces y aves. En estos enjambres, los individuos son relativamente simples en estructura, pero su comportamiento colectivo es generalmente complejo. El comportamiento complejo de un enjambre es el resultado del patrón de interacciones de los individuos a medida que pasa el tiempo. Así mismo, no es una propiedad de un solo individuo, y no suele ser fácil de predecir o deducir a partir de los patrones básicos de comportamiento de los individuos. Este proceso se conoce como "*emergencia"* y consiste en generar

[9]DORIGO, Marco y BIRATTARI, Mauro. Swarm Intelligence[On line]. Scholarpedia.A peer-reviewed open-access encyclopedia, 2013 [Citado el 15 de Marzo de 2013; 22:00:00]. Disponible en: http://www.scholarpedia.org/article/Swarm_intelligence

estructuras nuevas y coherentes, patrones y propiedades (o comportamientos) sin ningún sistema de control coordinado.

El elemento más importante en la inteligencia de enjambre, y facilitador del comportamiento emergente, es la interacción o cooperación. La interacción entre individuos ayuda a refinar el conocimiento que tiene cada individuo acerca de su entorno y se puede dar de dos formas: directa, por medio de contacto físico, o por medio de visual, entradas de audio perceptiva, o químicos; o indirecta, a través de cambios locales del medio ambiente. El término *estigmercia* se utiliza para referirse a la forma indirecta de comunicación entre varios individuos.[10]

A continuación se mencionan algunos ejemplos de sistemas biológicos que hacen uso de este tipo de interacción para lograr un objetivo comun.

- Las termitas construyen grandes nidos cuya complejidad está más allá del entendimiento y habilidad de una sola termita.
- Las tareas en una colonia de hormigas son dinámicamente asignadas, sin necesidad de un sistema central de coordinación.
- Los bancos de sardinas realizan maniobras colectivas en el mar para despistar a sus depredadores y evitar ser comidos. Y al igual que los bancos de aves, usan el conocimiento colectivo para guiar la búsqueda de comida en un ambiente cambiante.

El objetivo de los modelos de Inteligencia de Enjambres en Computación es modelar el comportamiento simple individual de los organismos, y las interacciones locales de estos individuos con sus vecinos y con el medio ambiente. Por ejemplo, en PSO se modelan dos simples conductas; cada individuo tiende a acercarse más a: 1) la mejor posición encontrada por algún individuo del enjambre, y 2) la mejor posición encontrada en su trayecto individual. En Optimización de Colonia de Hormigas (ACO) se modela el sistema de marcadores de pistas de feromonas, en el cual cada individuo (hormiga)

[10] ENGELBRETCHT, Op cit,. p. 285-286

selecciona probabilísticamente la dirección con la concentración más alta de feromonas.

2.10. Optimización de Enjambre de Partículas (PSO)

La optimización por Enjambre de Partículas (PSO), desarrollada por Kennedy y Eberhart (1995) [13], es un método de optimización para funciones no lineales en espacios continuos, basado en la simulación de un modelo social simple del desplazamiento de cardúmenes y bandadas.

El propósito inicial del modelo fue simular el movimiento sincrónico impredecible de una bandada de aves o un banco de peces. En este modelo cada individuo ajusta su velocidad de acuerdo a la velocidad del vecino más cercano y a una variable aleatoria, para mantener una distancia optima respecto a sus vecinos. La adición de la variable aleatoria evita que el movimiento de la bandada se vuelva uniforme y contribuye a una descripción más real de estos eventos naturales. Posteriormente se modificó el paradigma y se propuso un modelo en el cual las partículas (se asigna el nombre partícula a los agentes ya que estos no poseen masa ni volumen en el modelo) del grupo se mueven de acuerdo a la posición respecto a un puesto de comida fijo, la experiencia de vuelvo individual y la experiencia de vuelo del grupo. En este sentido, PSO combina un modelo "únicamente social", el cual sugiere que los individuos ignoran su propia experiencia y ajustan su conocimiento de acuerdo a las creencias exitosas de los individuos en la vecindad; y un modelo "únicamente cognitivo", el cual trata a los individuos como seres aislados. Cada partícula cambia de posición utilizando estos dos modelos. El paradigma puede ser utilizado como algoritmo para resolver problemas de optimización, si se sustituye el puesto de comida fijo del modelo anterior por una función *fitness* que permita evaluar el desempeño de viaje del enjambre, y si se adiciona un parámetro como número de iteración o tiempo real para terminar la corrida del algoritmo.

Un algoritmo PSO mantiene un enjambre de partículas, donde cada partícula representa una solución del problema. Estas partículas se mueven en un espacio multi-dimensional, y la posición de las partículas se ajusta de acuerdo al conocimiento social, individual y al efecto inercial. Se denota $\mathrm{x}_i(t) = (x_{i1}, x_{i2}, ..., x_{iD})$ la posición de la partícula i en un espacio de D-dimensiones en la iteración t. Esta posición se ajusta adicionando un vector de velocidad $\mathrm{v}_i(t) = (v_{i1}, v_{i2}, v_{i3}, ..., v_{iD})$, de la siguiente manera

$$\mathrm{x}_i(t+1) = \mathrm{x}_i(t) + \mathrm{v}_i(t+1)$$

El vector velocidad es el elemento que dirige el proceso de optimización, y refleja: el conocimiento que la partícula ha adquirido a través de la experiencia de vuelo, el conocimiento que ha adquirido de su entorno social, y el efecto inercial de vuelo de la partícula. El conocimiento individual es llamado el *componente cognitivo,* y es proporcional a la distancia entre la posición actual de la partícula y la mejor posición encontrada desde la primera iteración. El conocimiento adquirido del sistema es llamado *componente social* de la ecuación de velocidad y depende de la estructura de vecindario. De igual forma es proporcional a la distancia entre la partícula y la mejor posición del vecindario. El efecto inercial de vuelo se puede entender como la influencia de vuelos pasados en futuras posiciones de una partícula.

Inicialmente, dos tipos de algoritmos PSO fueron propuestos. Estos difieren en la forma como el conocimiento social es adquirido por cada una de las partículas.

En el algoritmo Global PSO, el vecindario de cada partícula es el enjambre entero. El tipo de red social que se utiliza en este algoritmo refleja una topología de tipo estrella, en la cual cada partícula tiene relación directa con todas las partículas del enjambre y así el conocimiento social que se adquiere es del enjambre entero. En este caso, el conocimiento social está representado por la partícula del enjambre

con mejor fittness, y esta denotada por y_{gbest}. Para el Global PSO, la velocidad de la partícula i se calcula de la siguiente manera

$$\mathrm{v}_{iD}(t+1) = \mathrm{v}_{iD}(t) + c_1 r_{1D}(t)[\mathrm{y}_{iD}(t) - \mathrm{x}_{iD}(t)] + c_2 r_{2D}(t)[\mathrm{y}_{gbest}(t) - \mathrm{x}_{iD}]$$

donde $\mathrm{v}_{iD}(t)$ es la velocidad inercial de la partícula. c_1 y c_2 son constantes de aceleración positiva utilizadas para escalar la contribución de cada uno de los componentes, y $r_1(t), r_2(t) \sim U(0,1)$, son números aleatorios con distribución uniforme en el rango (0,1).

La mejor posición individual asociada a cada partícula está representada por y_{iD}, y es la mejor posición encontrada desde el comienzo de las iteraciones. En problemas de minimización, la mejor posición individual en la iteración $t+1$ se calcula de la siguiente manera

$$\mathrm{y}_{iD}(t+1) = \begin{cases} \mathrm{y}_{iD}(t) & if \;\; f(\mathrm{x}_{iD}(t+1)) \geq f(\mathrm{y}_{iD}(t)) \\ \mathrm{x}_{iD}(t+1) & if \;\; f(\mathrm{x}_{iD}(t+1)) < f(\mathrm{y}_{iD}(t)) \end{cases}$$

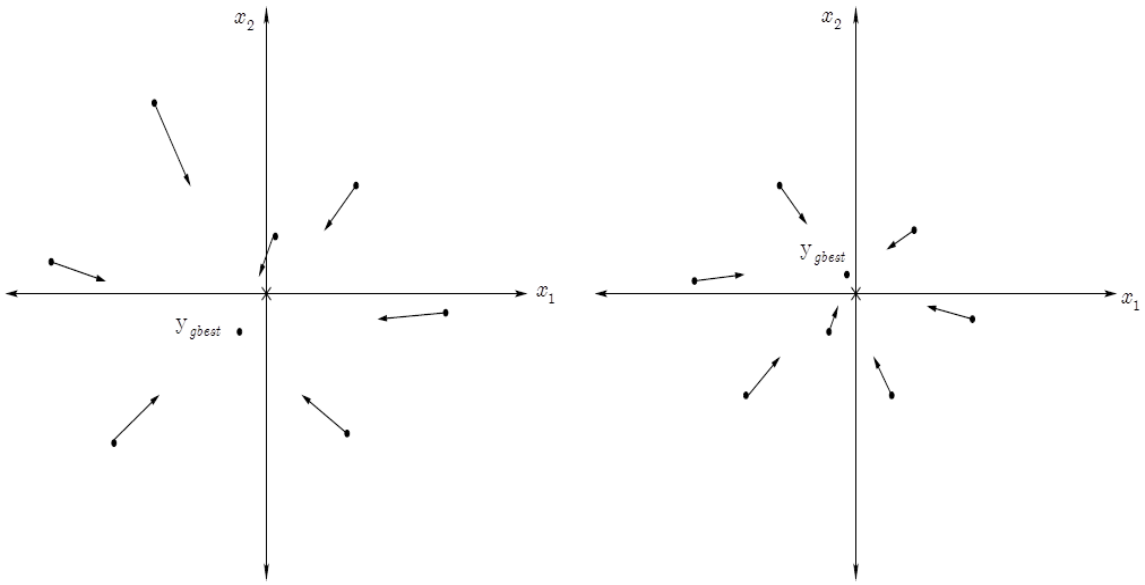
donde $f : \mathbb{R}^D \rightarrow \mathbb{R}$ es la función fitness para medir el desempeño de las partículas.

La mejor posición global y_{gbest}, en la iteración t, se define como

$$\mathrm{y}_{gbest}(t) \in \; \mathrm{y}_0(t),...,\mathrm{y}_{n_s}(t) \;\; | \, f(\mathrm{y}_{gbest}(t)) = \min \;\; f(\mathrm{y}_0(t)),...f(\mathrm{y}_{n_s}(t))$$

La Figura 10 muestra el comportamiento de un enjambre en 2 Dimensiones influenciado únicamente por la mejor posición Global. Se puede notar que en las dos iteraciones el vector de velocidad resultante fuerza las partículas a moverse más cerca de la mejor posición global que cambia en la segunda iteración.

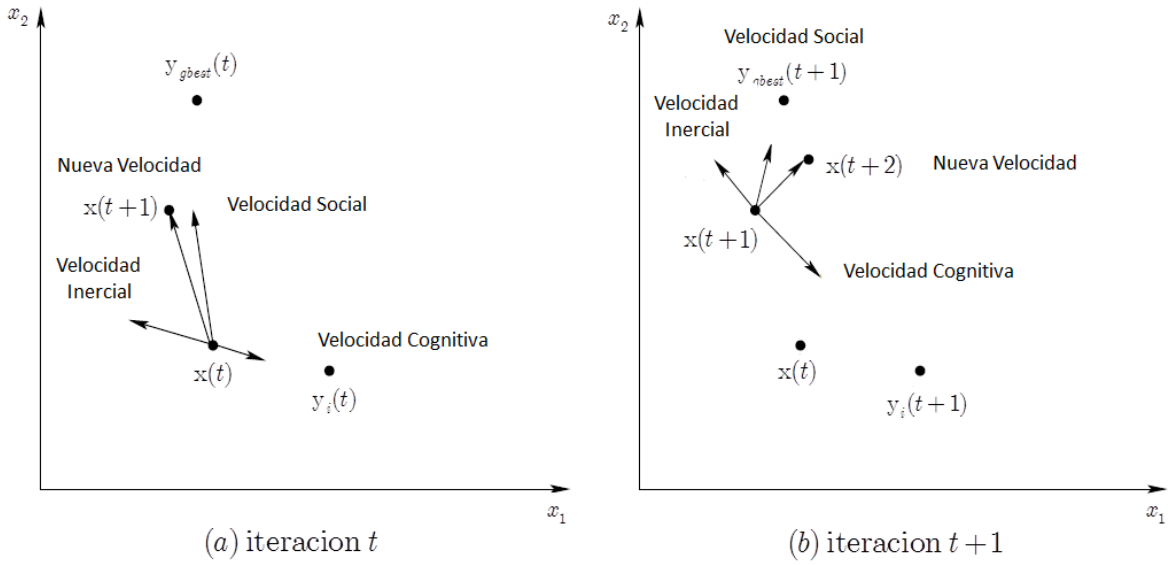
Figura 9. Ilustración de GLOBAL PSO



Fuente: ENGELBRETCHT, Op cit,. p. 295

En la Figura 11 se muestra un ejemplo del movimiento de una partícula influenciada por todos los componentes de la velocidad. Se puede observar que el vector Nueva Velocidad en las dos figuras es una combinación lineal de los diferentes componentes de la velocidad. En la Figura 11.a la Velocidad Social e Inercial son de mayor magnitud, por lo tanto la partícula se mueve más en estas dos direcciones. En la Figura 11.b la Velocidad Cognitiva tiene mayor magnitud, no obstante los otros dos componentes de velocidad tiene direcciones similares, lo que no permite un arrastre tan drástico del vector Nueva Velocidad en la dirección del vector Velocidad Cognitiva.

Figura 10. Ilustración Geométrica de Velocidad y Posición de una Partícula en un espacio 2D



Fuente: ENGELBRETCHT, Op cit,. p. 294

En el algoritmo Local PSO, los vecindarios son más péquenos, y cada partícula recibe información parcial de la condición del enjambre. El componente social en la ecuación de velocidad es proporcional a la distancia entre la partícula y la mejor posición del vecindario, y se denota por y_{iNbest}. El vector velocidad de Local-PSO difiere de Global-PSO únicamente en el cálculo del tercer término, como muestra la ecuación

$$\mathrm{v}_{iD}(t+1) = \mathrm{v}_{iD}(t) + c_1 r_{1D}(t)[\mathrm{y}_{iD}(t) - \mathrm{x}_{iD}(t)] + c_2 r_{2D}(t)[\mathrm{y}_{iNbest}(t) - \mathrm{x}_{iD}]$$

La mejor posición encontrada en el vecindario $\mathcal{N}_i$ por la partícula i, se define como

$$\mathrm{y}_{iNbest}(t+1) \in \mathcal{N}_i \mid f(\mathrm{y}_{iNbest}(t+1)) = \min\{f(\mathrm{x})\}, \ \forall \mathrm{x} \in \mathcal{N}_i$$

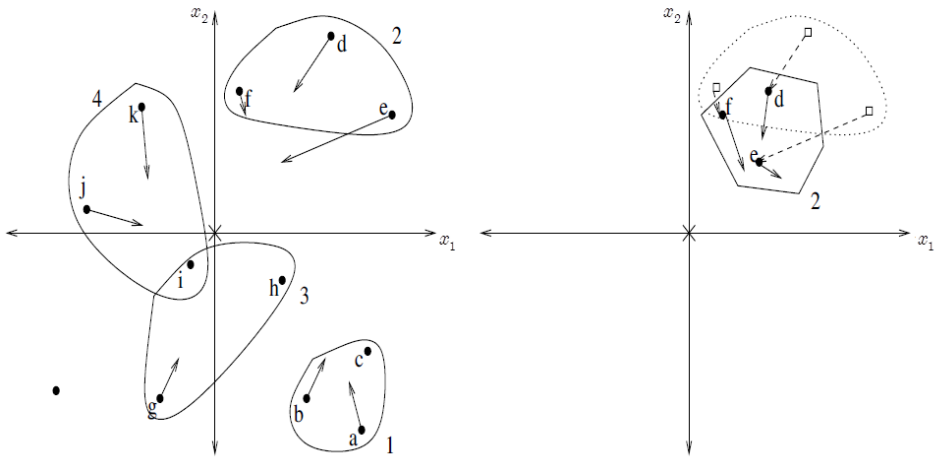
El vecindario $\mathcal{N}_i$ de la partícula i se define como el conjunto de elementos

$$\mathcal{N}_i = \mathrm{y}_{i-n_{\mathcal{N}_i}}(t), \mathrm{y}_{i-n_{\mathcal{N}_i+1}}(t), ..., \mathrm{y}_{i-1}(t), \mathrm{y}_i(t), \mathrm{y}_{i+1}(t), ..., \mathrm{y}_{i+n_{\mathcal{N}_i}}(t)$$

donde $n_{\mathcal{N}_i}$ es el tamaño del vecindario. Es importante notar que la estructura del vecindario está basada en los índices de la partícula y no en las distancias euclidianas de estas. Esta notación es conveniente, ya que el cálculo de vecindarios euclidianos requiere algoritmos cuya complejidad es de orden $O(n_s^{\,2})$, y promueve la diversificación ya que partículas en un mismo vecindario pueden tener distancias grandes.

En la Figura 12 se ilustra un ejemplo de como las partículas son influenciadas por sus vecinos inmediatos. Para mantener un grafica simple, solo ciertos vecindarios son ilustrados, y solo la dirección del vector de velocidad agregada es graficado. Así mismo, el vecindario basado en índices coincide con el vecindario euclidiano. En el vecindario 1, las partículas a y b se mueven hacia la partícula c, que es la mejor posición dentro del vecindario. En el vecindario 2, las partículas d y e tienden a moverse hacia f, esta última por ser la mejor del vecindario tiene un movimiento menor. En la siguiente iteración e es la mejor posición de ese vecindario, como se muestra en la parte b de la figura, y ahora f y d se mueven hacía e, el cual tiene un movimiento menor.

Figura 11. Ilustración de LOCAL PSO



(a) Local PSO - Enjambre Inicial (b) Local PSO - Enjambre en la siguiente iteracion

Fuente: ENGELBRETCHT, Op cit,. p. 296

En las dos versiones de PSO mencionadas, la actualización de la velocidad fuerza el enjambre a encontrar soluciones óptimas con el paso de las iteraciones. En Local PSO este proceso se da como resultado de la interconectividad (overlapping) de los vecindarios de las partículas (como se ilustra en los vecindarios 3 y 4 de la Figura 12). Sin embargo existe diferencia en la convergencia de los dos algoritmos. Debido al alto grado de interconectividad de vecindarios en Global PSO, el algoritmo converge rápidamente en óptimos locales. El menor grado de interconectividad en otros tipos de vecindarios (como anillo o árbol) permite que el algoritmo explore áreas diversas para evitar óptimos locales y en algunos casos alcanzar soluciones de mejor calidad.

2.10.1. Peso Inercial y Límite de Velocidad

Un aspecto importante que determina la eficiencia y precisión de un algoritmo de optimización es el balance entre la capacidad de *exploración* e *intensificación*.

Exploración es la habilidad para cubrir diferentes regiones del espacio de soluciones. *Intensificación* es la habilidad para refinar una solución candidata en una región promisoria del espacio de soluciones. En un buen algoritmo de optimización debe existir un balance apropiado entre estas dos habilidades, las cuales se pueden controlar con el vector velocidad.

En las primeras aplicaciones de PSO, se encontró que en muchos casos el tamaño del vector velocidad crecía más que exponencialmente y por lo tanto el enjambre divergía del espacio de soluciones. Para controlar el nivel de exploración global del enjambre y estas explosiones exponenciales, se introdujo *velocity clamping* o Límites de Velocidad. Si la velocidad de una partícula excede un valor máximo especificado, este último es asignado como velocidad de la partícula.

Se denota $V_{\max j}$ al máximo valor de velocidad permitido en la dimensión j. La velocidad de la partícula i es modificada antes de ajustar la posición de la partícula de la siguiente manera

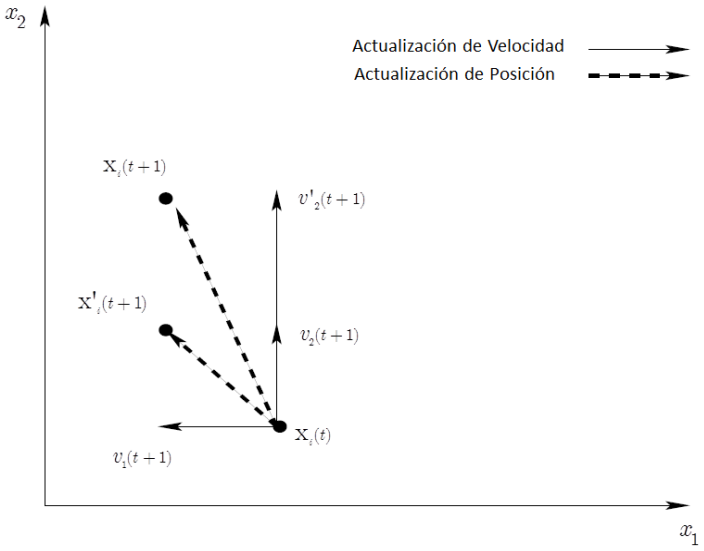
$$v_{ij}(t+1) = \begin{cases} v'_{ij}(t+1) & if \;\; v'_{ij}(t+1) < V_{\max j} \\ V_{\max j} & if \;\; v'_{ij}(t+1) \geq V_{\max j} \end{cases}$$

donde v'_{ij} es la velocidad calculada directamente de la ecuación de la velocidad.

El valor de $V_{\max j}$ es importante ya que ayuda a controlar las habilidades de *exploración* e *intensificación* del enjambre. Cuando el valor de $V_{\max j}$ es grande, facilita la exploración de distintas regiones del espacio de soluciones, sin embargo se puede omitir o "sobrevolar" soluciones de calidad. Cuando el valor de $V_{\max j}$ es pequeño, se favorece una búsqueda más exhaustiva en una región dada, no obstante se corre el riesgo de quedar atrapado en un óptimo local. Un aspecto importante a tener en cuenta es que los Límites de Velocidad se imponen en la magnitud de cada dimensión de una partícula, por lo tanto la dirección de esta

puede cambiar distorsionando los movimientos de búsqueda. En la Figura 13 se ilustra el efecto del Límite de la Velocidad. El componente 2 de la velocidad excede el límite $V_{\max j}$ y su magnitud es reducida. Esto ocasiona un cambio de dirección como se observa en la diferencia entre las dos posiciones (líneas punteadas)

Figura 12. Efecto del límite de velocidad (Velocity Clamping)



Fuente: ENGELBRETCHT, Op cit,. p. 305

Otro problema con los Límites de Velocidad ocurre cuando todas las dimensiones del vector velocidad son igual a la velocidad máxima $V_{\max j}$. Si no se establecen medidas para evitar esta situación, las partículas limitan la búsqueda a la superficie de un hiper-cubo definido por la ecuación $[\mathrm{x}_i(t)-V_{\max}, \mathrm{x}_i(t)+V_{\max}]$. Este problema se puede evitar reduciendo el valor de $V_{\max j}$ con el paso de las iteraciones. La idea de esta estrategia es permitir que el enjambre se disperse

buscando áreas promisorias en el espacio de soluciones (*exploración)* y a medida que se reduce el valor de la velocidad máxima se promueve la búsqueda local en la áreas visitas por el enjambre (*intensificación)*.

El peso inercial w fue propuesto por Shi y Eberhart (1998) [28] como un mecanismo de control de las habilidades de *exploración* e *intensificación, y* como elemento substitutivo de los Límites de Velocidad. Aunque demostró ser efectivo como mecanismo de control, no logro eliminar por completo la necesidad de los Límites de Velocidad.El peso inercial es utilizado para controlar el impacto de las velocidades previas en la velocidad actual de las partículas. Si w es grande el efecto de vuelo inercial puede ser predominante en la dirección y magnitud de la velocidad y de esta manera se incentiva la habilidad de *exploración* de distintas regiones. Si w es pequeño, la experiencia colectiva y cognitiva pueden toman mayor importancia y así las partículas difícilmente salen de las regiones que el grupo o la experiencia propia sugieren son más promisorias. La ecuación de la velocidad con este nuevo parámetro es la siguiente.

$$\mathrm{v}_{iD}(t+1) = w\,\mathrm{v}_{iD}(t) + c_1 r_{1D}(t)[\mathrm{y}_{iD}(t) - \mathrm{x}_{iD}(t)] + c_2 r_{2D}(t)[\mathrm{y}_{gbest}(t) - \mathrm{x}_{iD}]$$

2.11. Optimización Por Enjambre de Partículas en un Espacio de Soluciones Discreto (DPSO)

El algoritmo PSO fue inicialmente diseñado para tratar problemas cuyo espacio de soluciones es continuo. Para resolver problemas con espacios de soluciones discretos, es necesario redefinir la distancia entre soluciones, y bajo esta noción de distancia redefinir los operadores del PSO. Kennedy y Eberhart (1997) [14] propusieron el primer algoritmo basado en representación binaria (discreta) de las soluciones del problema. Este algoritmo puede ser utilizado para resolver problemas puramente discretos, y problemas continuos cuyas soluciones puedan ser representadas en punto-flotante. Cada partícula del enjambre es una cadena

binaria x_{id}. La trayectoria de cada partícula está definida por el vector velocidad $\mathrm{v}_{id} = \{v_{i1}, v_{i2}, ..., v_{id}\}$, donde v_{id} representa la probabilidad de que la coordenada j de la partícula i cambie el valor a su complemento. La fórmula del vector velocidad se mantiene igual que en PSO continuo

$$v_{id} = w * v_{id} + c_1(y_{id} - x_{id}) + c_2(y_{gbest} - x_{id})$$

Debido a que los operadores del vector velocidad (suma y producto) producen vectores en $\mathbb{R}$, se utiliza la función sigmoid para mapear los valores reales a un intervalo cerrado $[0,1]$. Así, el cambio de posición de cada partícula en la iteración k está definido por la siguiente regla

$$if(rand() < Sigmoid(v_{id}^k))\ then\ \ x_{id}^k = \neg(x_{id}^{k-1})$$
$$else\ \ x_{id}^k = x_{id}^{k-1}$$

Los autores recomiendan utilizar un Límite de Velocidad, ya que valores mayores a 6 producen probabilidades mayores que 0.9975 y menores que 1 por medio de la función sigmoid, lo cual llevaría al algoritmo a converger prematuramente. El algoritmo fue utilizado para resolver los 5 problemas de De Jong's Suite (1975), y encontró la solución óptima en la mayoría de los casos con algunas imprecisiones debido a la codificación binaria de la solución.

Khanesar *et al* (2007) [15] propusieron una versión más sofisticada de BPSO llamada Novel-BPSO. En esta versión, cada partícula tiene asociado 2 vectores de velocidad $\vec{V}_i^0$ y $\vec{V}_i^1$ que representan la probabilidad de que la partícula cambie sus valores a 0 y 1 respectivamente. Debido a que las probabilidades en los vectores mencionados no son complemento, la probabilidad de cambio en la dimensión j de la partícula i está dada por

$$V_{ij} = \begin{cases} V_{ij}^{1} & ,if\ \ x_{ij} = 0 \\ V_{ij}^{0} & ,if\ \ x_{ij} = 1 \end{cases}$$

Los vectores de velocidad de cada partícula son actualizados en cada iteración de acuerdo a un conjunto de reglas que favorecen el estado actual del mejor en el vecindario y el mejor personal. Por ejemplo, si la dimensión j de la mejor partícula en el enjambre es 1, entonces se aumenta el valor de velocidad V_{ij}^{1} de la partícula i en evaluación y se disminuye el valor en la dimensión especificada del otro vector de velocidad. Una vez se ha modificado los vectores de velocidad de la partícula, se utiliza la ecuación (anterior) para determinar cuál es la probabilidad de cambio de la dimensión j, y se aplica la función sigmoid para mapear ese valor a un rango $[0,1]$. Al igual que en BPSO original, se genera un número aleatorio y se evalúa la posibilidad de cambio en cada dimensión de la partícula.

El manejo de dos vectores de velocidad independientes tiene dos ventajas: 1) La nueva posición de cada partícula no depende únicamente del vector velocidad, también de la posición actual de la partícula (ecuación anterior). 2) El efecto del Límite de Velocidad y el Peso Inercial tiene el mismo efecto (es decir, valor altos de los dos parámetro favorecen exploración y valores pequeños intensificación) que en PSO continuo, contrario al algoritmo BPSO original

Yang *et al* (2004) [30] propusieron un algoritmo eficiente llamado Quantum-PSO para tratar problemas con representación binaria. Inspirado en conceptos de la teoría cuántica, en la cual un quantum bit es la mínima entidad que carga información y se encuentra en una superposición de los estados 0 y 1, se define un enjambre $Q(t)$ de vectores quantum $\mathrm{q}_i(t) = [q_{i1}, q_{i2}, ..., q_{im}]$ donde $0 \le q_{ij}(t) \le 1 (j = 1, ..., m;\ i =, ..., N)$; m es la longitud de la partícula y N es el tamaño del enjambre; $q_{ij}(t)$ representa la probabilidad del bit j de la partícula i de

ser 0 en la iteración t. Para obtener un vector binario x_{id} (representación de la solución del problema) de un vector quantum, se genera un vector de valores aleatorios en el rango $[0,1]$ y se evalúa cada bit; si el valor aleatorio es mayor que el bit correspondiente la dimensión en el vector binario es 1, de lo contrario es 0. El vector quantum se actualiza en cada iteración de acuerdo a la siguiente ecuación

$$Q(t+1) = c_1 * Q(t) + c_2 * Q_{perbest}(t) + c_3 * Q_{globest}(t)$$

donde c_1, c_2 y c_3 representan la influencia del vuelo anterior, influencia cognitiva y global respectivamente, y $c_1 + c_2 + c_3 = 1, 0 < c_1, c_2, c_3 < 1$. Los vectores $Q_{perbest}(t)$ y $Q_{globest}(t)$ se calculan de la siguiente manera.

$$Q_{globest}(t) = \alpha * p_{globest}(t) + \beta * (1 - p_{globest}(t))$$
$$Q_{perbest}(t) = \alpha * p_{perbest}(t) + \beta * (1 - p_{perbest}(t))$$

donde α y β son parámetros de control y $\alpha + \beta = 1, 0 < \alpha, \beta < 1$. El algoritmo es utilizado para resolver el Knapsack Problem y un grupo de funciones continuas complejas, y presenta mejores resultados en convergencia que BPSO y GA. En la sección (3) se muestra el pseudocodigo de este algoritmo.

Martinez-Garcia y Moreno-Perez (2008) [19] proponen un algoritmo inspirado en PSO para resolver problemas discretos llamado Jumping Frog Optimization. En JFO se eliminan el vector velocidad y el peso inercial, y la posición de las partículas o "ranas" se actualiza de acuerdo a una serie de "saltos" aleatorios en direcciones determinadas. La siguiente ecuación expresa formalmente el cambio de posición

$$\mathrm{x}_i = c_1\mathrm{x}_i \oplus c_2\mathrm{x}_{i,bestper} \oplus c_3\mathrm{x}_{i,bestneig} \oplus c_4\mathrm{x}_{globest}$$

donde c_1, c_2, c_3 y c_4 constituyen una partición del conjunto unidad, y por lo tanto se pueden interpretar como rangos de probabilidades. El resultado de esta operación consiste en seleccionar un punto de referencia para iniciar los movimientos de búsqueda. Los cuatro puntos son: un punto aleatorio, el mejor personal encontrado, el mejor del vecindario y el mejor global, cada uno con probabilidad de ocurrencia c_1, c_2, c_3 y c_4 respectivamente. Por lo tanto, en cada iteración se genera un número aleatorio y dependiendo del rango en que caiga se inicia la búsqueda en la direccion correspondiente. Para el caso de la P-mediana, los movimientos de búsqueda se pueden hacer por medio de intercambio, inserción o eliminación de vértices, y el número de movimientos en esa direccion se determina aleatoriamente de acuerdo a una distribución geométrica con media igual al producto entre p y el coeficiente correspondiente c_i. El algoritmo se puede utilizar cuando la representación de la solución son permutaciones de tamaño p o cadenas binarias de tamaño n.

Moraglio et al (2007) [21] proponen una interpretación geométrica de los operadores evolutivos para definir un algoritmo, llamado Geometric-PSO, que puede tratar cualquier tipo de representación de soluciones bajo los principios de PSO. Los operadores evolutivos de cruce se definen en términos geométricos utilizando la noción de segmento de línea y circulo, y se muestra como estos operadores pueden generar combinaciones convexas de un conjunto de soluciones para intensificar la búsqueda en el espacio de soluciones. Así mismo, muestran la importancia de implementar mutación ya que por medio de este mecanismo se pueden explorar soluciones distantes. Los autores presentan las definiciones sobre los espacios Euclidiano, Manhattan y de Hamming, y utilizan el algoritmo para resolver el problema del Sudoku.

Qin *et al* (2009) [23] toman la idea de Geometric-PSO de utilizar un espacio métrico para definir una relación (distancia) entre las soluciones del problema. Sin embargo, rechazan la idea de utilizar operadores evolutivos para actualizar los individuos y definen una nueva interpretación de movimiento y velocidad de las partículas. Debido a que la velocidad en términos generales es la diferencia de posición con respecto a un intervalo de tiempo (ya sea velocidad instantánea o velocidad media), se piensa en la velocidad de la partículas discretas como la cantidad de movimientos necesarios para llegar de una solución a otra en una iteración. Luego utilizan el mismo concepto de dirección y movimiento propuesto por Martinez-Garcia y Moreno-Perez (2008) para actualizar las partículas.

Otros autores proponen distintos métodos de codificación de la solución del problema para aplicar el PSO.

- Ai y Kachitvichyanukul (2009) [1] proponen dos métodos de representación de la solución del CVRP (Capacitated Vehicle Routing Problem). La representación de las soluciones (partículas) son vectores con valores reales que vuelan sobre un espacio continuo. Los vectores por si solos no dicen nada acerca de la solucion en particular, por lo tanto es necesario utilizar un método de decodificación para hacer la asignación de los clientes, establecer la ruta óptima para cada vehiculo y evaluar el desempeño de esa solución. En cada iteración (generación) se actualizan las partículas con los operadores tradicionales de PSO continuo, y se utiliza el método de decodificación para evaluar cada solucion.
- Chen *et al* (2005) [2] proponen una representación binaria de la solución del CVRP para aplicar Quantum-PSO. Cada partícula esta compuesta por $N * k$ celdas, donde N es el número de clientes y k el número de vehículos. Cada grupo de N celdas tiene unos en los clientes que pertenecen a ese grupo o vehículo. El orden de ruteo está definido por la indexación numérica del vector binario, y se utiliza Recocido Simulado para optimizar el orden de los clientes para cada vehículo. Debido a que las

operaciones de Q-PSO no aseguran factibilidad de las soluciones, se utiliza un conjunto de reglas simples (como romper empates aleatoriamente o aplicar nuevamente los operadores QPSO) para alcanzar factibilidad. Este conjunto de reglas se puede entender como un proceso de decodificación de la partícula, ya que se parte de un conjunto de datos y se aplica un procedimiento para obtener información que es útil con respecto al problema (una solución factible).

2.12. Método de Codificación

Cuando se aplica Metaheuristicas Evolutivas a problemas de carácter combinatorio, se busca encontrar métodos para representar las soluciones del problema de manera que se pueda aplicar operadores evolutivos de búsqueda. Los métodos de codificación son importantes, ya que en algunos casos proveen formas efectivas de alcanzar soluciones optimas y de controlar habilidades de intensificación o exploración a través de operadores que actúan sobre el conjunto de codificaciones. Por ejemplo, los operadores estándar de los algoritmos genéticos junto con un método de codificación apropiado, han probado ser útiles en alcanzar soluciones de calidad en problemas de alta complejidad. Sin embargo, encontrar formas efectivas para codificar las soluciones de problemas discretos para una metaheuristica evolutiva en particular es un asunto de gran complejidad.

Los métodos de codificación han sido tratados desde diferentes perspectivas, como: la relación entre el espacio fenotípico y el espacio genotípico que es cuando se decodifican los individuos en soluciones del problema; y la evolución de los individuos que es cuando se aplican operadores evolutivos para obtener mejores

soluciones.[11]A continuación se hace una revisión no inclusiva de ciertos aspectos importantes referente a los métodos de codificación.

2.12.1. Clasificación de las Codificaciones

Existen diferentes tipos de clasificaciones. La primera hace referencia al alfabeto o tipo de símbolo usado en la codificación de la solución.

- Codificación binaria: consiste en utilizar un alfabeto de solo 0 y 1 fue la primer forma como se implementaron los algoritmos genéticos en varios problemas de optimización. Se han planteado modificaciones del PSO original para tratar codificaciones binarias.
- Codificación números reales: consiste en utilizar un alfabeto teórico del conjunto de todos los números reales, en la práctica el computador solo puede representar un numero finito muy grande de números decimales por medio de punto flotante. Es ideal para la optimización de funciones sobre el espacio continuo, así mismo se han propuesto codificaciones con símbolos reales de problemas combinatorios cuyo espacio de soluciones es discreto.
- Codificación Enteros y Permutaciones: Además de la codificación binaria y real, también se pueden utilizar conjuntos de enteros o permutaciones de conjuntos de enteros. Este tipo de codificación ha sido ampliamente utilizado ya que genera partículas o codificaciones compactas que requieren menos memoria que las codificaciones binarias en problemas discretos como la P-mediana y CVRP entre otros.
- Estructura de datos generales: En algunos problemas con estructuras complejas, la mejor forma de capturar información de la solución es a través de n-eadas o estructuras de datos más complejas.

[11]GEN, Mitsuo y CHENG, Runwei.Genetic algorithms and Engineering Optimization [On line].WILEY & SONS.p. 1.Google Books, 2013. [Citado en 19 de marzo de 2013; 16:00:00]. Disponible en: http://books.google.com/

Los métodos de codificación también se pueden clasificar de acuerdo a las estructura de la codificación

- codificación unidimensional: Consiste en utilizar una sola cadena de símbolos para representar una solución del problema. Es ampliamente utilizado debido a su practicidad.
- codificación multidimensional: consiste en utilizar un grupo de cadenas de símbolos para representar una solución de los problemas. Es necesario y útil cuando los problemas a resolver tienen una estructura compleja.

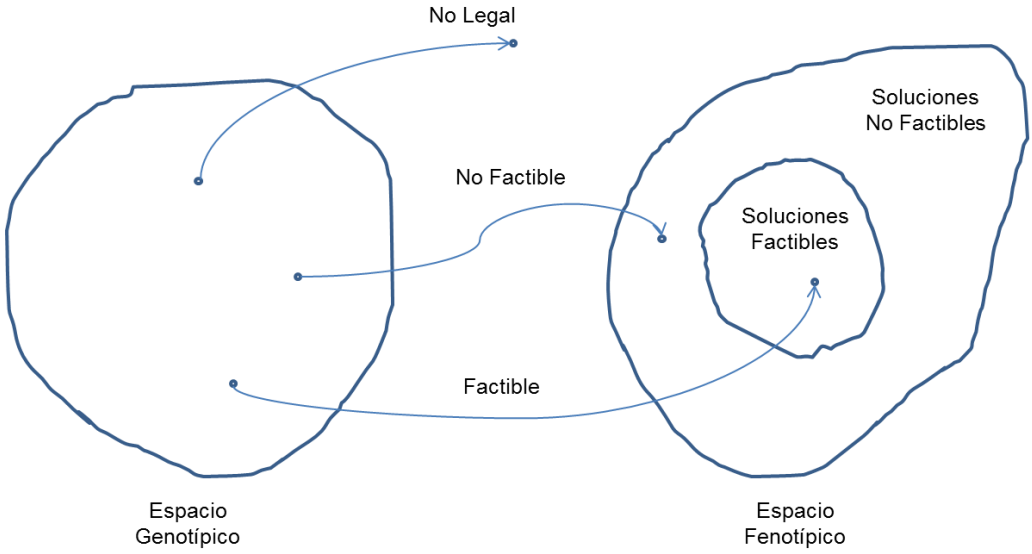
2.12.2. No legalidad y No factibilidad.

Cuando se requiere utilizar un método de codificación para aplicar una metaheuristica en particular se trabaja sobre dos conjuntos de elementos: el conjunto de codificaciones y el conjunto de soluciones, o en otras palabras, el espacio genotípico y el espacio fenotípico. Las funciones que se llevan a cabo en cada espacio dependen de la metaheuristica a aplicar por lo tanto se necesitan métodos eficientes de para mapear elementos del primer conjunto al segundo. Por ejemplo, en los algoritmos genéticos los operadores de cruce y mutación se aplican sobre el espacio genotípico mientras que las operaciones de selección y evaluación de la solución se hace en el espacio fenotípico. En el PSO continuo aplicado a funciones de optimización el espacio genotípico es equivalente al espacio fenotípico y por lo tanto los operadores evolutivos (se considera la actualización de la posición a través del vector velocidad como un operador evolutivo) y la evaluación de las soluciones se hacen sobre el mismo espacio (el conjunto de vectores en $\mathbb{R}^n$ donde n son las dimensiones de la partícula).

Uno de los inconvenientes más recurrentes con los sistemas de codificación en los problemas de optimización con restricciones y optimización combinatoria es cuando las codificaciones mapean a soluciones no factibles. En este sentido es

clave distinguir entre no factibilidad y no legalidad. No factibilidad se refiere al fenómeno de que una solución decodificada yace en la región no factible del espacio de soluciones del problema original. No legalidad se refiere al fenómeno de que no es posible decodificar un elemento del espacio genotípico ya que no tiene correspondencia en el espacio fenotípico.

Figura 13. Casos de mapeo del conjunto de codificaciones al conjunto de soluciones



Fuente: GEN. Op cit,. p. 5

El hecho de que ocurra no factibilidad en los elementos del espacio genotípico para un problema de optimización con restricciones es natural; cualquiera que sea el método de codificación utilizado, se tiene que tratar con las restricciones inherentes al problema. En muchos problemas de optimización, la región de soluciones factibles se puede representar a través de un sistema de igualdades y

desigualdades. En estos casos es posible tratar el evento de no factibilidad con métodos de penalización de las restricciones.

El hecho de que una codificación sea no legal se debe a la naturaleza de la codificación utilizada. Debido a que estos elementos no legales no se pueden mapear en soluciones, no es posible utilizar los métodos de penalización. En estos casos, es útil utilizar Métodos de Reparación para transformar las codificaciones no legales en codificaciones legales. Aunque no se han diseñado codificaciones con este problema para PSO discreto, existen codificaciones que generan cromosomas (codificaciones) no legales en ciertas implementaciones de los algoritmos genéticos. En general, los métodos de reparación son útiles para tratar el evento de no factibilidad y no legalidad de las codificaciones en problemas de optimización combinatoria.

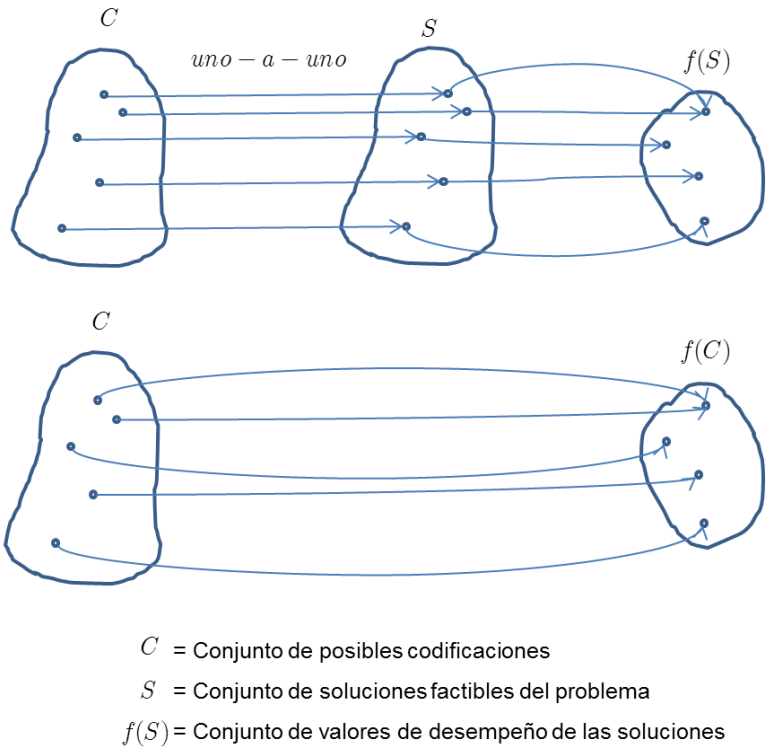
2.12.3. Propiedades de las Codificaciones

2.12.3.1. Redundancia

La relación entre los elementos del conjunto de codificaciones y el conjunto de soluciones debe ser uno-a-uno.

Cuando se diseña un método de codificación se busca establecer una relación binaria uno-a-uno entre los elementos del espacio genotípico y los elementos del espacio fenotípico, de esta manera se evita la ejecución de operaciones triviales de búsqueda o de evaluación en una corrida del algoritmo. En la siguiente figura se muestra una relación entre los conjuntos mencionados. Debido a la relación uno-a-uno entre el espacio genotípico (conjunto de codificaciones) y el espacio fenotípico (conjunto de soluciones del problema) se puede definir la evaluación de las soluciones en función directa del conjunto de codificaciones.

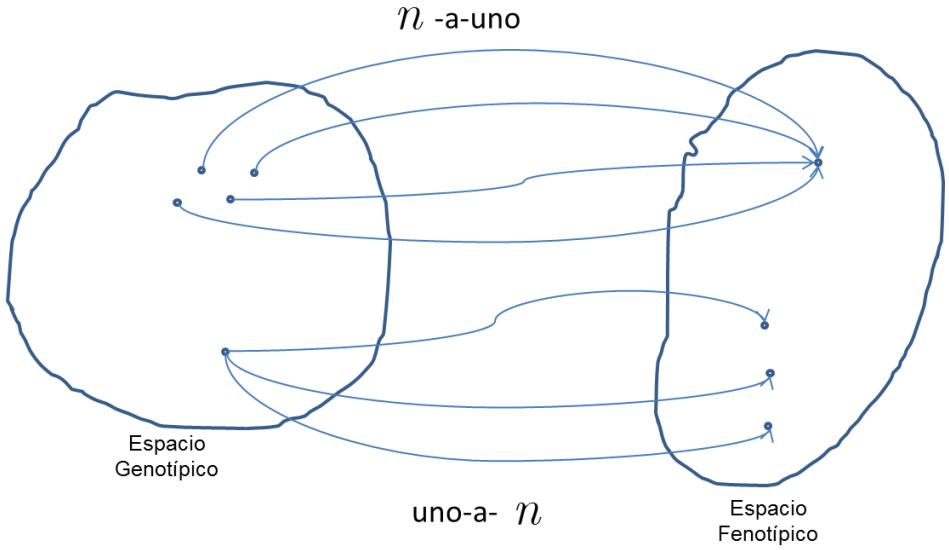
Figura 14. Caso ideal de relación entre conjunto de codificaciones y conjunto de soluciones



Fuente: Autor

Establecer la relación uno-a-uno con una codificación inclusiva y eficiente es casi imposible para muchos problemas. En la práctica ocurren dos eventos, uno menos deseable que el otro. Si el método de codificación genera una relación de n-a-uno, es decir más de una codificación mapea a una sola solución del problema (Figura 15), se puede gastar tiempo en la búsqueda de soluciones y puede hacer que el algoritmo en general tenga convergencia prematura. El caso menos deseable es cuando el método de codificación genera una relación de uno-a-n ,es decir una codificación mapea a diferentes soluciones, ya que se necesita ejecutar procedimientos sobre el espacio fenotípico para escoger una solución.

Figura 15. Casos de mapeo del conjunto de codificaciones al conjunto de soluciones



Fuente: GEN. Op cit,. p. 6

2.12.3.2. Legalidad

Cualquier permutación del sistema de codificación corresponde a una solución del problema

Esta propiedad garantiza que cada codificación corresponde a un elemento en el espacio de soluciones. Como se mencionó anteriormente, los métodos de codificación tradicionales (binaria y permutaciones) aplicado al PSO nunca producen codificaciones no legales.

2.12.3.3. Completitud

Cada solución del problema corresponde a una codificación.

Esta propiedad garantiza que todas las soluciones del problema son accesibles a través de la búsqueda en el conjunto de codificaciones.

2.12.3.4. Causalidad

Pequeñas variaciones en el espacio genotípico generan pequeñas variaciones en espacio de soluciones

Esta propiedad hace referencia a la estructura de vecindario en el espacio genotípico y fenotípico. Cuando la estructura de vecindario se conserva en los dos espacios se dice que la codificación tiene *causalidad fuerte*. Por otro lado, si pequeños movimientos o variaciones en uno de los espacios causa el efecto contrario en el otro espacio o viceversa se dice que el sistema de codificación tiene *causalidad débil.*

3. Diseño de la Solución

En base a los algoritmos de Chen *et al* (2005) y Ai y Kachitvichyanukul(2009), para resolver el problema CVRP, se propone un algoritmo que utiliza la misma idea de codificar la solución del problema para aplicar los operadores de movimiento del BPSO, y posteriormente decodificar esa partícula para evaluar el desempeño de la solución incumbente.

Las representaciones o codificación tradicionales del PMP para aplicar los operadores de los métodos metaheuristicos evolutivos, contienen información relacionada únicamente al conjunto de medianas. Por ejemplo en los algoritmos genéticos, Ant Colony Optimization y JFO, las representaciones se basan en cadenas binarias de tamaño n o cadenas de números enteros de tamaño p (Figura 17) que únicamente contienen los índices de las medianas, y en cada iteración luego de actualizar la partícula se hace la asignación de los clientes; es decir, al momento de utilizar los operadores evolutivos para obtener mejores soluciones (ya sean diversas y/o de calidad) en cada iteración, no se utiliza información de los vértices de los clientes en este proceso.

Figura 16. Representación gráfica de una solución del PMP y dos codificaciones.

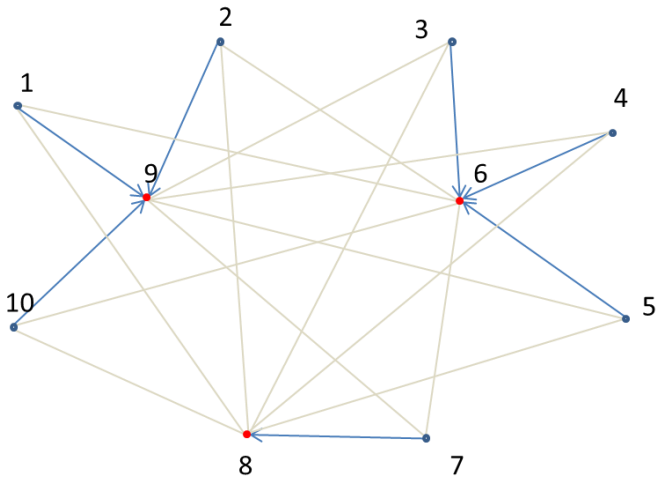
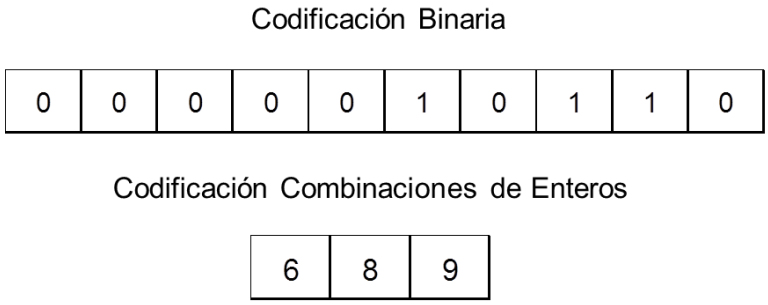


Figura 16. (Continuacion)



Fuente: Autor

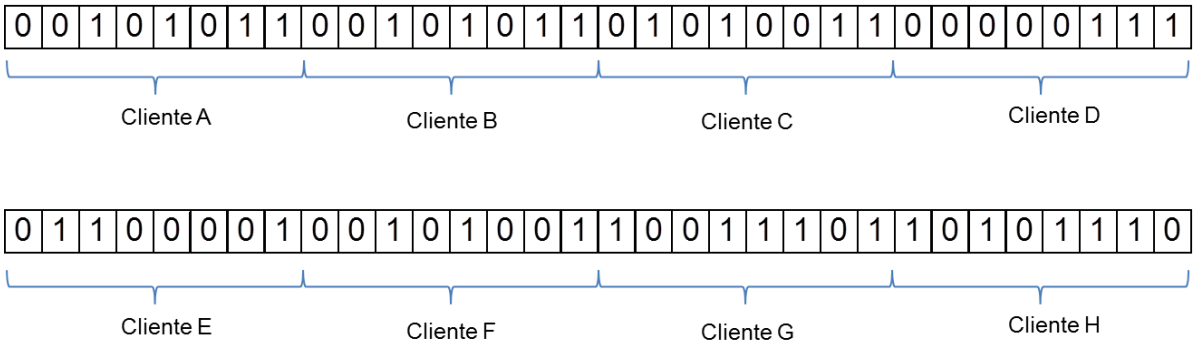
El Método de Codificación que proponemos es una nueva representación binaria de la solución del PMP que se basa en información de los vértices de demanda, y luego se aplica una serie de pasos simples junto con una Heurística Constructiva y una de Búsqueda Local para obtener una solución de calidad (solución factible y aproximada a la óptima global).

3.1. Descripción detallada del algoritmo PSO para programación binaria.

3.1.1. Codificación de la Solución

Se propone una nueva representación binaria de la solución del PMP para aplicar metaheuristicas discretas basadas en PSO. La partícula o codificación es una cadena binaria de tamaño $n * n$. Cada n celdas representa un cliente (nodo de demanda) y dentro de cada bloque de n celdas, las posiciones que tienen 1 representa los posibles candidatos (medianas) para ese cliente. A continuación se muestra un ejemplo de una partícula con $n = 8$ y $p = 3$.

Figura 17. Partícula o Codificación nueva de una solución del PMP



Fuente: Autor

En la siguiente figura se muestra la interpretación grafica de cada uno de los bloques de la partícula. Por ejemplo, según el estado actual de la codificación o partícula el cliente A(grafica con la A en color rojo) tiene 4 candidatos para mediana, los vértices C,E, G y H; así mismo, el cliente C (grafica con la C tres en color rojo) tiene 4 candidatos para medianas, los vértices B,D, G y H. Si se lee la partícula como una solución del PMP, es una solución que no cumple con las restricciones de numero de medianas (ya que si se suman todas las medianas de todos los bloques de la partícula probablemente resulta en n) y no cumple con las restricciones de reparto o asignación única (ya que cada bloque sugiere que varias medianas prestan servicio a ese cliente en particular.). En este sentido, es necesario aplicar un método que permita saber a cuál solución del problema corresponde esta partícula.

Figura 18. Grafica de la interpretación de cada bloque de la codificación binaria

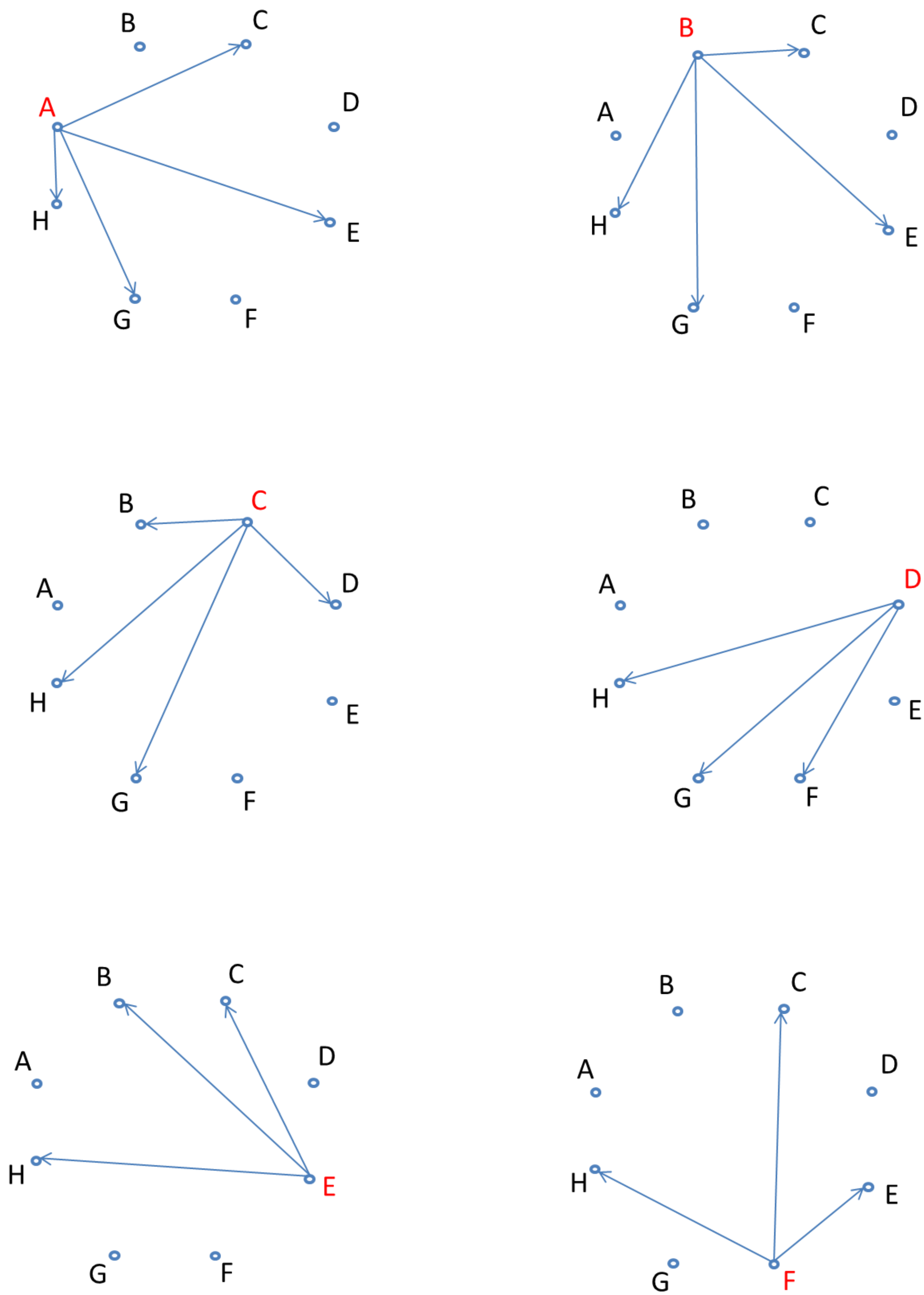
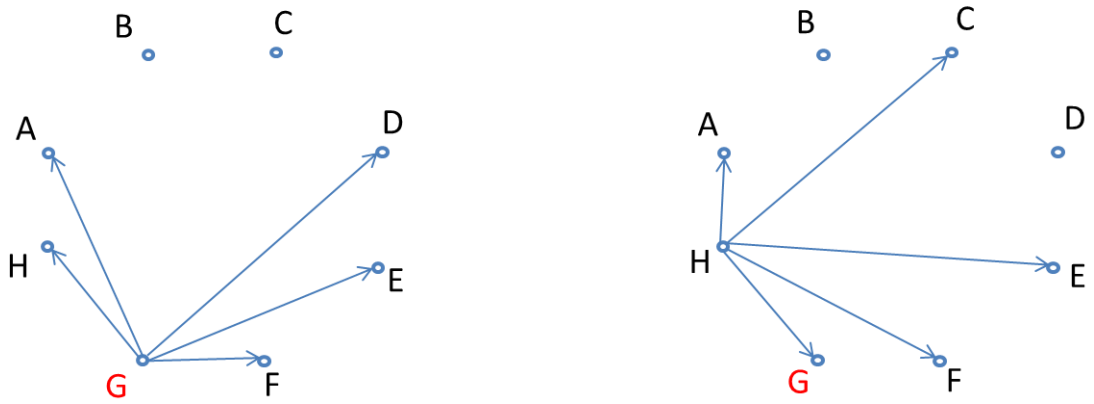


Figura 18. (Continuación)



Fuente: Autor

La idea de esta nueva representación, en contraste con las representaciones tradicionales del PMP, es utilizar información de posibles asignaciones de cada cliente para evolucionar la búsqueda bajo estas condiciones y construir soluciones factibles (método constructivo) y soluciones mejores (método búsqueda local) para encontrar soluciones optimas aproximadas de calidad.

3.1.2. Tipo de Metaheuristica

Antes de describir el método de decodificación, se va a describir el tipo de metaheuristica sobre la cual se va a implementar. Debido a que la representación es binaria, se puede utilizar los algoritmos BPSO original, Novel-BPSO y Quantum-PSO. Los dos últimos superan en desempeño al primero, por lo tanto queda descartado BPSO original [30] [15]. Novel-PSO mostro resultados promisorios en la solución de problemas con espacios discretos, sin embargo requiere mantener dos vectores de velocidad por cada partícula, situación que no es favorable desde el punto de vista de memoria teniendo en cuenta el tamaño de

las partícula de la nueva representación. Una de las ventajas de Novel-PSO es que se tiene en cuenta la memoria de vuelo en el momento de actualizar la posición, sin embargo Quantum-PSO también posee esta propiedad ya que no evalúa la probabilidad de cambio al complemento como en BPSO, en cambio evalúa la posibilidad de estadía en 0 o 1 (en cada dimensión de la partícula) por medio del único vector quantum asociado a cada partícula. Así mismo, Quantum-PSO ha sido más utilizado que Novel-PSO para implementar algoritmos basados en PSO a problemas discretos, por lo tanto se escoge Quantum-PSO para implementar el método de codificación.

A continuación se muestra los parámetros, las variables y el pseudocodigo del algoritmo metaheuristico Quantum-DPSO .

N = tamaño del enjambre

D = tamaño de la partícula

$X(t)$ = enjambre en la iteración t

$\mathrm{x}_i(t)$ = partícula i en la iteración t

$x_{ij}(t)$ = componente j de la partícula i en la iteración t

$X(t) = \{\mathrm{x}_1(\mathrm{t}),\mathrm{x}_2(\mathrm{t}),...,\mathrm{x}_N(t)\}; \mathrm{x}_i(t) = [x_{i1}(t), x_{i2}(t),..., x_{iD}(t)]$

$Q(t)$ =quantum vectores del enjambre en la iteración t

$\mathrm{q}_i(t)$ = quantum vector de la partícula i en la iteración t

$q_{ij}(t)$ = bit j de la partícula i en la iteración t

$Q(t) = \{\mathrm{q}_1(\mathrm{t}),\mathrm{q}_2(\mathrm{t}),...,\mathrm{q}_N(t)\}; \mathrm{q}_i(t) = [q_{i1}(t), q_{i2}(t),..., q_{iD}(t)]$

$Q_{gbest}(t)$ = vector quantum del mejor del enjambre en la iteración t

$\mathrm{y}_{gbest}(t)$ =mejor partícula del enjambre en la iteración t

$Q_{perbest}(t)$=vectores quantum del mejor personal de cada partícula en la iteración t

$\mathrm{y}(t)$=mejores posiciones de las partículas del enjambre en la iteración t

$Q_{i,perbest}(t)$=vector quantum del mejor personal de la partícula i

$\mathrm{y}_i(t)$= mejor posición de la partícula i en la iteración t

$$Q_{perbest}(t) = \{Q_{1,perbest}(t), Q_{2,perbest}(t), ..., Q_{N,perbest}(t)\}$$

$$c_1 + c_2 + c_3 = 1, 0 < c_1, c_2, c_3 < 1$$

$$\alpha + \beta = 1\ 0 < \alpha, \beta < 1$$

Figura 19. Algoritmo QUANTUM - DPSO

Comenzar Programa
$t = 0$
inicializar $Q(t)$ y $X(t)$
decodificar $X(t)$ y evaluar las soluciones correspondientes
guardar la mejor solución del enjambre y el mejor personal
mientras (condición de parada es falsa)
$t = t + 1$
cambiar $Q(t)$ usando Quantum-DPSO
$Q_{gbest}(t) = \alpha * \mathrm{y}_{gbest}(t) + \beta * (1 - \mathrm{y}_{gbest}(t))$
$Q_{perbest}(t) = \alpha * \mathrm{y}_i(t) + \beta * (1 - \mathrm{y}_i(t))$
$Q(t+1) = c_1 * Q(t) + c_2 * Q_{perbest}(t) + c_3 * Q_{globest}(t)$
evaluar $Q(t)$ para obtener $X(t)$
if $\mathrm{rand} > \mathrm{q}_{ij}(t) \quad \{\forall i \in N, j \in D\}$
$x_{ij}(t) = 1$
else
$x_{ij}(t) = 0$
decodificar $X(t)$ y evaluar las soluciones correspondientes
guardar la mejor solución del enjambre y el mejor personal
fin_mientras
fin_programa

Fuente: Autor

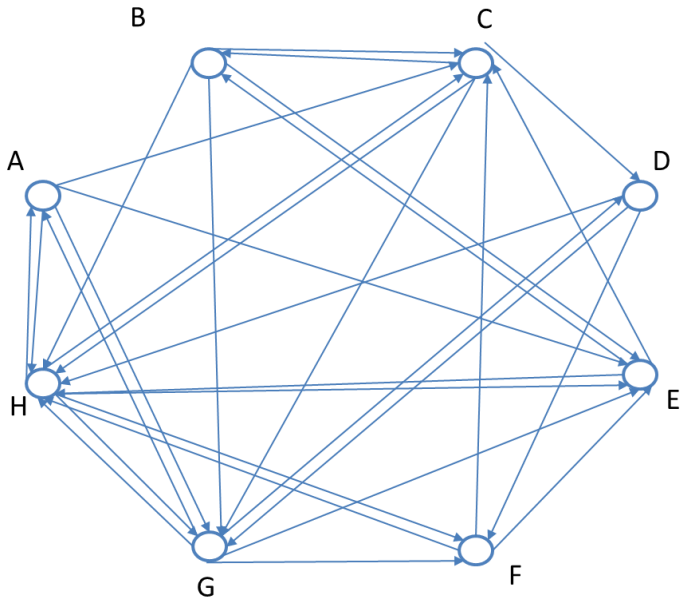
3.2. Algoritmo de Decodificación

Cuando se utiliza un método para representar una solución del problema original se llama *codificación* de la solución, y cuando se utiliza un método para convertir la codificación en una solución se llama *decodificación.*

El algoritmo de decodificación tiene tres fases principales: Tamizado de Medianas, Reparación y Mejoramiento.

1) Tamizado de Medianas: Esta parte del algoritmo consiste en obtener una aproximación del mejor conjunto de medianas de la partícula por medio de unos pasos simples. Ya que los 1 en la partícula representan aristas dirigidas, si se sobreponen todos los bloques de la Figura 19 en una misma gráfica, se obtiene un grafo donde cada vértice tiene un grado-interior diferente.

Figura 21. Interpretación de la codificación o partículas



Fuente: Autor

Para hallar el grado-interior de cada vértice se toma la partícula y se forma una matriz de $n \times n$ con cada bloque en una fila. Luego se suman los valores de las filas para todas las columnas y se obtiene un vector de valores enteros.

Figura 22. Decodificación de la Partícula.

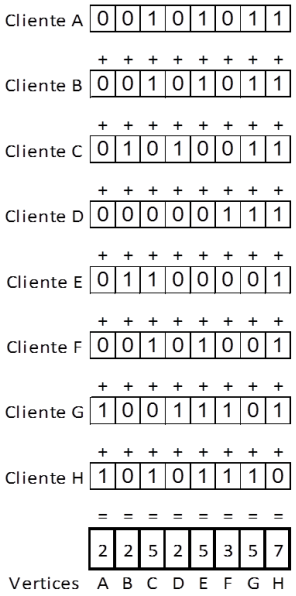


Figura a. Vector de Enteros

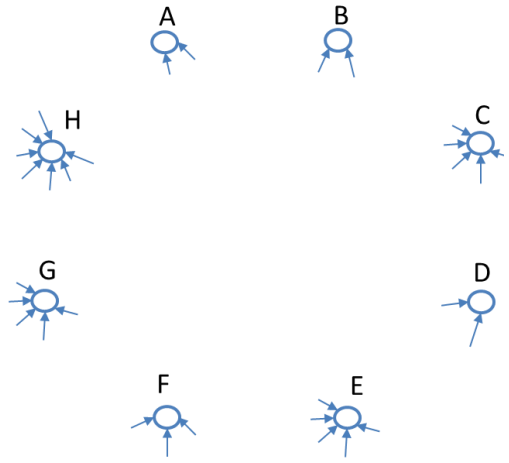


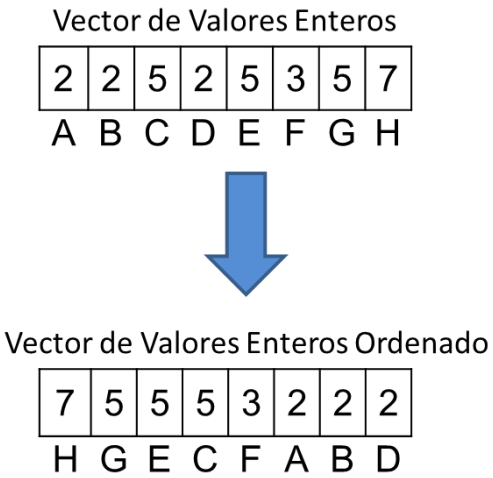
Figura b. Interpretación Grafica del Vector de Enteros

Fuente: Autor

El vector entero que resulta de esa operación representa los "votos" que recibe cada vértice (grado-interior de cada vértice) para ser mediana (Figura 22.b). Por ejemplo, el vértice H recibió el mayor número de votos de los demás vértices en ese proceso. Posteriormente se ordena el vector de valores enteros de acuerdo a los valores de "votación" de mayor a menor, ya que se

quiere obtener un conjunto inicial de medianas de acuerdo a los valores más altos.

Figura 23. Ordenar vector de valores enteros



Fuente: Autor

En este caso se tiene que escoger tres medianas. La primer celda que es el vértice H entra a ser mediana ya que su valor 7 es el más alto. Ahora quedan dos vértices por escoger y los siguientes vértices con los valores más altos son G,E y C. No se escoge arbitrariamente G y E como las otras dos medianas, ya que la idea del método de decodificación es suministrar una solución aproximada a la óptima, y puede ser el caso que el vértices C es una mejor opción que alguno de los dos vértices mencionados para ser parte del conjunto de medianas. Así mismo, el orden de estos tres elementos es dependiente de la estabilidad del algoritmo *sorting* que fue utilizado para

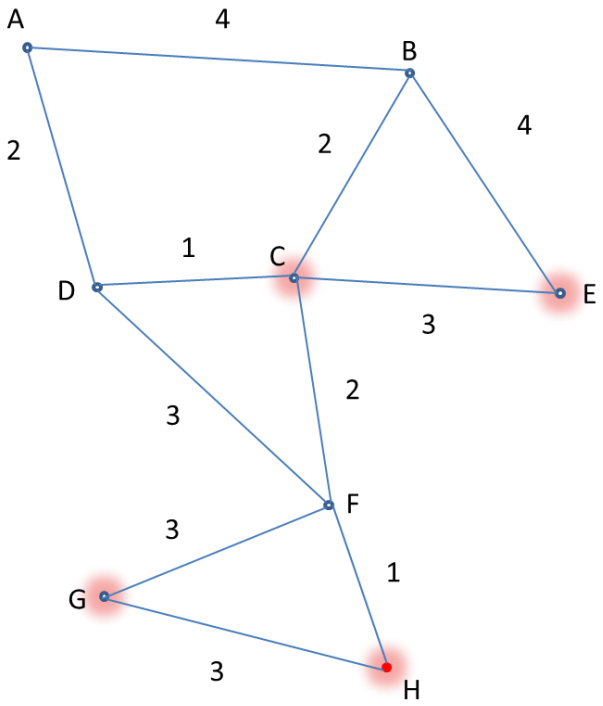
ordenar el vector de valores enteros, y se quiere un método que no dependa del tipo de subrutinas utilizadas para la decodificación.*

2) Reparación: La segunda y tercera fase del método de decodificación fue diseñado en base a la heurística GRASP-PathRelinkingen la cual se construyen diferentes soluciones y se aplican métodos de mejoramiento a cada una de las soluciones iniciales.

   El método de reparación solo se utiliza en casos en que es necesario romper empates como el mencionado en el numeral anterior. Para romper el empate se define un subproblema que corresponde a una variación del PMP estándar. En continuación con el ejemplo de la partícula codificada, el subproblema es: Existen 8 clientes que necesitan ser surtidos por 3 medianas, sin embargo, dado que una mediana ya está localizada en el vértice H, cual es el lugar óptimo de las otras dos medianas si solo se pueden localizar en los vértice G,E y C? A continuación se muestra una gráfica que ilustra los vértices del grafo sobre un plano de dos dimensiones; aunque no se muestran las aristas, estas equivalen a las distancias entre clientes (puntos azules) y las posibles medianas (puntos con sombra roja) únicamente, por lo tanto, la matriz de distancias es de orden 10×4. La columna en rojo indica que esta facilidad se incluye en cualquier cálculo de distancias mínimas ya que corresponde a la facilidad pre-instalada.

* Los algoritmos estables mantienen el orden de las celdas que tienen valores iguales, mientras que los inestables cambian la posición de estos valores en el proceso de ordenar. Algoritmos de ordenar como Insertionsort y Bubblesort son estables, mientras que Heapsort y Quicksort son inestables.

Figura 24. Representación gráfica y matriz no-simetrica de distancias del subproblema



$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{cccc} C & E & G & H \end{array} \\ \left\{ \begin{array}{cccc} 9 & 18 & 24 & 18 \\ 8 & 16 & 28 & 20 \\ 0 & 6 & 10 & 6 \\ 4 & 16 & 24 & 16 \\ 3 & 0 & 8 & 6 \\ 4 & 10 & 6 & 2 \\ 20 & 32 & 0 & 12 \\ 3 & 6 & 3 & 0 \end{array} \right\} \end{array}$$

Vértice	A	B	C	D	E	F	G	H
Población (miles)	3	4	2	4	1	2	4	1

Fuente: Autor

Para resolver el subproblema se utiliza el algoritmo greedy clásico ya que se necesita construir una solución factible con tres medianas. A diferencia del algoritmo GRASP en el cual se utilizan variaciones de greedy como randomized greedy para diversificar la búsqueda, en el presente método de decodificación se utiliza el greedy clásico ya que se necesita un método consistente para decodificar las soluciones. Por ejemplo, si se utiliza randomized greedy y en una corrida del algoritmo PSO el enjambre visita la

misma partícula varias veces, es bastante probable que esa misma partícula corresponda a diferentes soluciones en la misma corrida ya que cada partícula que visita esa posición construye una solución diferente, generando así el caso menos deseado de la propiedad de redundancia que es mapeo de n-a-uno.

3) Mejoramiento: El método de mejoramiento consiste en coger la solución construida en el paso 1 o 2 y aplicar un método de búsqueda local para mapear la partícula a la mejor solución del vecindario de la solución construida. En el presente método se utiliza la técnica robusta de Teitz and Bartz de intercambio de vértices.

A continuación se presenta el pseudocodigo de la Heuristica Greedy y Vertex Substitution y se ilustra un ejemplo gráfico de cómo operan los dos métodos. Para el ejemplo se va utilizar el mismo problema de la codificación ($n = 8$ y $p = 3$) cuya matriz de distancias se encuentra en la Figura 5. Aunque en el ejemplo de codificación se establece una mediana definitiva antes de aplicar la reparación con Greedy, en el ejemplo gráfico se ilustra cómo se aplica Greedy al PMP estándar, es decir, sin facilidades pre-instaladas en el grafo. Este cambio se hace con el propósito de mostrar cómo opera la heurística desde el comienzo hasta el final, así mismo no hay pérdida de información ya que el hecho de que hayan o no facilidades pre-instaladas no afecta el movimiento de adición de vértices Greedy.

3.2.1. Greedy para el PMP

1) Inicializar el conjunto de mediana $V_{incumb} = \phi$
2) Añadir momentáneamente cada vértice $v_i \notin V_{incumb}$ al conjunto de medianas y se forma $V_{incumb,i}$, esto es, $V_{incumb,i} = V_{incumb} \cup v_i$.

i) Para cada vértice v_k en $V_{incumb,i}$ definir un conjunto de vértices de destino P_k de manera que cada vértice $v_l \notin V_{incumb,i}$ sea asignado al vértice $v_k \in V_{incumb,i}$ más cercano. La colección

$$P = \{P_k \mid v_k \in V_{incumb,i}\}$$

de subconjuntos de vértices P_k constituye una partición de V. Calcular la suma total de las distancias mínimas para cada sistema $V_{incumb,i}$.

$$r_{incumb,i} = \sum_{v_k \in V_1} \sum_{v_l \in P_k} r_{lk}$$

$r_{incumb,i}$ Representa el valor potencial del conjunto de medianas si el vértice $v_i \notin V_{incumb}$ se añade al conjunto de medianas.

3) Encontrar el vértice $v_z \notin V_{incumb}$ tal que $r_{incumb,z} = \min_{i \notin V_{incumb}} r_{incumb,i}$. Es decir, encontrar el vértice fuera del conjunto actual (incumbente) de medianas que al añadirlo al conjunto de medianas produce el mínimo incremento en la evaluación de la suma total de distancias mínimas.

   i) Si el tamaño de V_{invumb} es menor que el parámetro p ($|V_{incumb}| < p$) entonces ir al paso 2.

   ii) Parar. El conjunto actual V_{incumb} es la solución optima encontrada por el algoritmo.

3.2.1.1. Ejemplo Gráfico de Greedy

1) $V_{incumb} = \{ \phi \}$

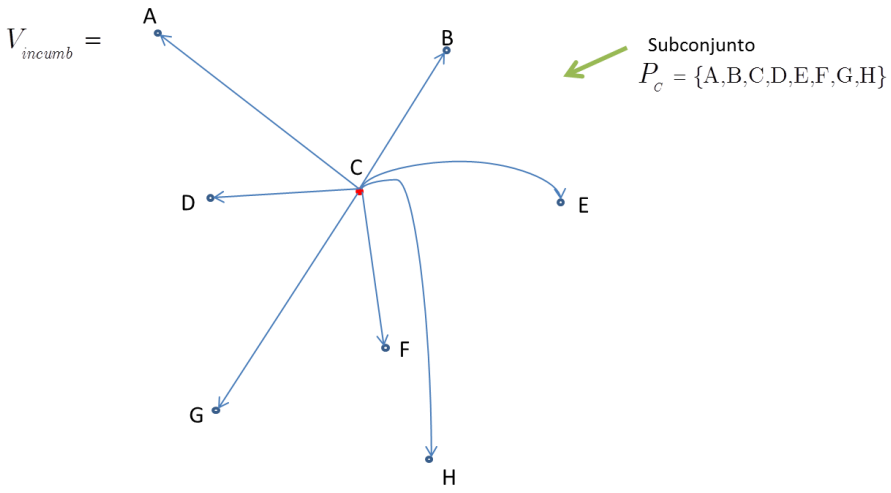
2)

	A	B	C	D	E	F	G	H
A	0	12	9	6	18	15	24	18
B	16	0	8	12	16	16	28	20
C	6	4	0	2	6	4	10	6
D	8	12	4	0	16	12	24	16
E	6	4	3	4	0	5	8	6
F	10	8	4	6	10	0	6	2
G	32	28	20	24	32	12	0	12
H	6	5	3	4	6	1	3	0
	$V_{incumb,A}$	$V_{incumb,B}$	$V_{incumb,C}$	$V_{incumb,D}$	$V_{incumb,E}$	$V_{incumb,F}$	$V_{incumb,G}$	$V_{incumb,H}$
$r_{incumb,i} =$	84	73	51	58	104	65	103	80

3) C

$r_{incumb,C} = 51$ ⟹ $V_{incumb} = \{\phi\} \cup C$ ⟹ $V_{incumb} = \{ C \}$ ⟹

$| V_{incumb} | == 3$? ⟹ **NO** ⟹ **Ir a 2)**



2)

	A	C	B	C	C	D	C	E	C	F	C	G	C	H
A	0	9	12	9	9	6	9	18	9	15	9	24	9	18
B	16	8	0	8	8	12	8	16	8	16	8	28	8	20
C	6	0	4	0	0	2	0	6	0	4	0	10	0	6
D	8	4	12	4	4	0	4	16	4	12	4	24	4	16
E	6	3	4	3	3	4	3	0	3	5	3	8	3	6
F	10	4	8	4	4	6	4	10	4	0	4	6	4	2
G	32	20	28	20	20	24	20	32	20	12	20	0	20	12
H	6	3	5	3	3	4	3	6	3	1	3	3	3	0

	$V_{incumb,A}$	$V_{incumb,B}$	$V_{incumb,D}$	$V_{incumb,E}$	$V_{incumb,F}$	$V_{incumb,G}$	$V_{incumb,H}$
$r_{incumb,i}$ =	42	43	44	48	37	**31**	36

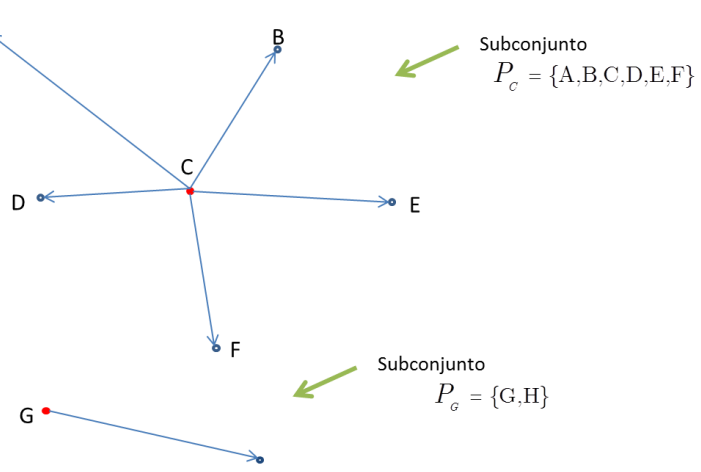
3)

G

$r_{incumb,G} = 31$ ⇒ $V_{incumb} = \{C\} \cup G$ ⇒ $V_{incumb} = \{C, G\}$

⇒ $|V_{incumb}| == 3$? ⇒ **NO** ⇒ **Ir a 2)**

V_{incumb} =



2)

	A	C	G
A	0	9	24
B	16	8	28
C	6	0	10
D	8	4	24
E	6	3	8
F	10	4	6
G	32	20	0
H	6	3	3

$V_{incumb,A}$

	B	C	G
A	12	9	24
B	0	8	28
C	4	0	10
D	12	4	24
E	4	3	8
F	8	4	6
G	28	20	0
H	5	3	3

$V_{incumb,B}$

	C	D	G
A	9	6	24
B	8	12	28
C	0	2	10
D	4	0	24
E	3	4	8
F	4	6	6
G	20	24	0
H	3	4	3

$V_{incumb,D}$

	C	E	G
A	9	18	24
B	8	16	28
C	0	6	10
D	4	16	24
E	3	0	8
F	4	10	6
G	20	32	0
H	3	6	3

$V_{incumb,E}$

	C	F	G
A	9	15	24
B	8	16	28
C	0	4	10
D	4	12	24
E	3	5	8
F	4	0	6
G	20	12	0
H	3	1	3

$V_{incumb,F}$

	C	G	H
A	9	24	18
B	8	28	20
C	0	10	6
D	4	24	16
E	3	8	6
F	4	6	2
G	20	0	12
H	3	3	0

$V_{incumb,H}$

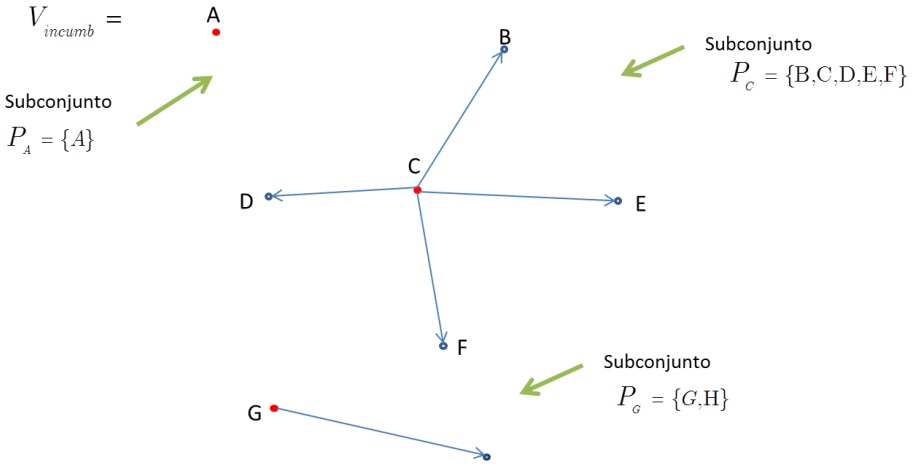
$r_{incumb,i}$ =	$V_{incumb,A}$	$V_{incumb,B}$	$V_{incumb,D}$	$V_{incumb,E}$	$V_{incumb,F}$	$V_{incumb,H}$
	22	23	24	28	25	26

3)

A

$r_{incumb,A} = 22$ → $V_{incumb} = \{C,G\} \cup A$ → $V_{incumb} = \{C,G,A\}$

→ $|V_{incumb}| == 3$? → **SI** → **Parar** → El conjunto $\{C,G,A\}$ es la solución optima encontrada por Greedy .



3.2.2. Heuristica Vertex Substitution

El método Vertex Substitution de Teitz y Bartz (1964) es citado en la literatura como una de las mejores heurísticas para dar solución optimas aproximadas de buena calidad al PMP. Por esta razón, es utilizado como punto de referencia para medir la eficiencia de nuevos algoritmos y es utilizado como subrutina de diferentes algoritmos Metaheuristicos.

Vertex Substituon es un método de búsqueda local (métodos que buscan mejores soluciones dentro de una estructura de vecindario), y pertenece al grupo de técnicas $\lambda - \text{interchange}$,utilizada en varios problemas combinatorios, que fue generalizada posteriormente por Lin y Kernighan (1973) para el problema TSP.

Vertex Substituion puede ser descrita de la siguiente manera.

1) Seleccionar un conjunto inicial V_{incumb} de p vértices. Para cada vértice v_j en V_{incumb} definir un conjunto de vértices de destino P_j de manera que cada vértice $v_i \notin V_{incumb}$ sea asignado al vértice $v_j \in V_{incumb}$ más cercano. La colección

$$P = P_j \mid v_j \in V_{incumb}$$

de subconjuntos de vértices P_j constituye una partición de V. Calcular la suma total de las distancias mínimas para el sistema V_{incumb}.

$$r_{incumb} = \sum_{v_j \in V_1} \sum_{v_i \in P_j} r_{ij}$$

2) Seleccionar algún vértice v_b que no esté en el conjunto V_{incumb} (y que no haya sido intentando en iteraciones anteriores) y marcar como "intentado". Para cada vértice v_j en V_{incumb} substituir con v_b (es decir remover la mediana actual e insertar la que se está probando) y computar el

incremento $\triangle_{bj}$ que produce este intercambio de vértices. $\triangle_{bj} = r_{incumb,bj} - r_{incumb}$ donde $r_{incumb,bj}$ es la evaluación del subconjunto P_j si se intercambia el vértice b que no está en V_{incumb} por j que si pertenece.

3) Buscar el vértice v_k en V_{incumb} que produzca el mayor incremento negativo. Es decir,

$$\triangle_{bk} < 0 \text{ y } \triangle_{bk} = \min \triangle_{bj} \quad (j = 1,2,...,p)$$

   1. Si existe un vértice v_k que cumpla las condiciones anteriores, intercambiar con el vértice v_b y llamar a este nuevo subconjunto de vértice V_{incumb}. Calcular la suma total de distancias mínimas. En este punto se vuelven a rotular todos los vértices que no pertenecen V_{incumb} como no "intentado" para comenzar el ciclo de búsqueda nuevamente con el paso 2.
   2. Si no existe un vértice v_k que cumpla las condiciones anteriores, retener el subconjunto actual como esta y volver al paso 2

4) Repetir los pasos 2 y 3 hasta que todos los vértices en el complemento de V_{incumb} sean rotulados como "intentado". Este procedimiento es llamado un Ciclo.

   Si durante el último ciclo ninguna substitución es hecha en el paso 3.2, ir al paso 5. De lo contrario, si una substitución es hecha seguir las reglas indicadas en el paso 3.1

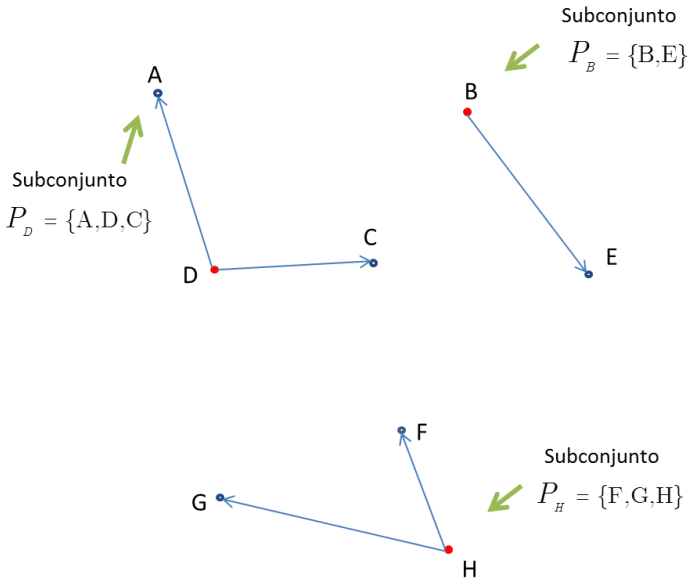
5) Parar. El conjunto actual V_{incumb} es la solución óptima encontrada por el algoritmo.

3.2.2.1. Ejemplo Gráfico de Vertex Substitution

1)

$$V_{incumb} = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{ccc} D & B & H \end{array} \\ \begin{bmatrix} \boxed{6} & 12 & 18 \\ 12 & \boxed{0} & 20 \\ \boxed{2} & 4 & 6 \\ \boxed{0} & 12 & 16 \\ 4 & \boxed{4} & 6 \\ 6 & 8 & \boxed{2} \\ 24 & 28 & \boxed{12} \\ 4 & 5 & \boxed{0} \end{bmatrix} \end{array}$$

$$r_{incumb} = 26$$

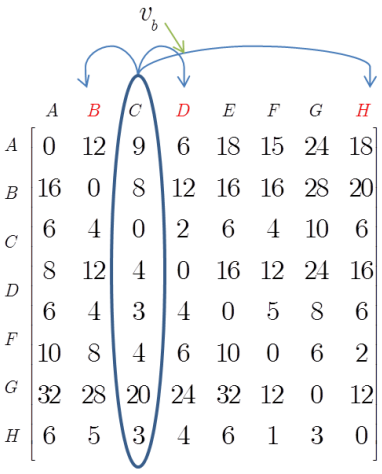


2)

$$\begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{ccc} D & C & H \end{array} \\ \begin{bmatrix} \boxed{6} & 9 & 18 \\ 12 & \boxed{8} & 20 \\ 2 & \boxed{0} & 6 \\ \boxed{0} & 4 & 16 \\ 4 & \boxed{3} & 6 \\ 6 & 4 & \boxed{2} \\ 24 & 20 & \boxed{12} \\ 4 & 3 & \boxed{0} \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{ccc} B & C & H \end{array} \\ \begin{bmatrix} 12 & \boxed{9} & 18 \\ \boxed{0} & 8 & 20 \\ 4 & \boxed{0} & 6 \\ 12 & \boxed{4} & 16 \\ 4 & \boxed{3} & 6 \\ 8 & 4 & \boxed{2} \\ 28 & 20 & \boxed{12} \\ 5 & 3 & \boxed{0} \end{bmatrix} \end{array} \qquad \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \\ F \\ G \\ H \end{array} \begin{array}{c} \begin{array}{ccc} C & B & D \end{array} \\ \begin{bmatrix} 9 & 12 & \boxed{6} \\ 8 & \boxed{0} & 12 \\ \boxed{0} & 4 & 2 \\ 4 & 12 & \boxed{0} \\ \boxed{3} & 4 & 4 \\ \boxed{4} & 8 & 6 \\ \boxed{20} & 28 & 24 \\ \boxed{3} & 5 & 4 \end{bmatrix} \end{array}$$

$$r_{incumb,bj} = 31 \qquad 30 \qquad 36$$

$$\Delta_{CB} = 31 - 26 = 5 \qquad \Delta_{CD} = 30 - 26 = 4 \qquad \Delta_{CH} = 36 - 26 = 10$$



3) Hay algún incremento $(\Delta_{CB},\Delta_{CD},\Delta_{CH})$ que sea negativo ? → **NO** → Rotular *C* como intentado. Existen mas vértices no intentados?

→ **SI** → **Ir a 2)**

2)

	D	E	H
A	6	18	18
B	12	16	20
C	2	6	6
D	0	16	16
E	4	0	6
F	6	10	2
G	24	32	12
H	4	6	0

$r_{incumb,bj} = 34$

$\Delta_{EB} = 34 - 26 = 8$

	B	E	H
A	12	18	18
B	0	16	20
C	4	6	6
D	12	16	16
E	4	0	6
F	8	10	2
G	28	32	12
H	5	6	0

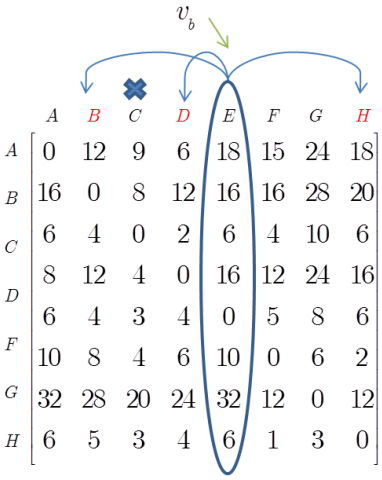
42

$\Delta_{ED} = 42 - 26 = 16$

	E	B	D
A	18	12	6
B	16	0	12
C	6	4	2
D	16	12	0
E	0	4	4
F	10	8	6
G	32	28	24
H	6	5	4

42

$\Delta_{EH} = 42 - 26 = 16$



3) Hay algún incremento $(\Delta_{EB},\Delta_{ED},\Delta_{EH})$ que sea negativo ? → **NO** → Rotular *E* como intentado. Existen mas vértices no intentados?

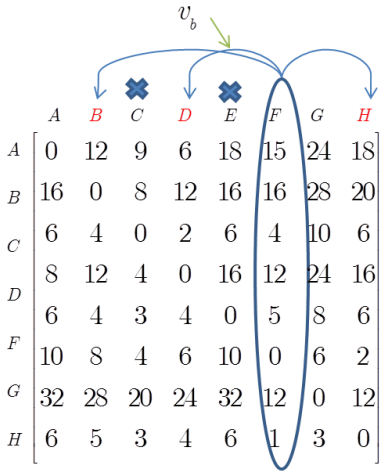
→ **SI** → **Ir a 2)**

2)

	D	F	H
A	6	15	18
B	12	16	20
C	2	4	6
D	0	12	16
E	4	5	6
F	6	0	2
G	24	12	12
H	4	1	0

	B	F	H
A	12	15	18
B	0	16	20
C	4	4	6
D	12	12	16
E	4	5	6
F	8	0	2
G	28	12	12
H	5	1	0

	F	B	D
A	15	12	6
B	16	0	12
C	4	4	2
D	12	12	0
E	5	4	4
F	0	8	6
G	12	28	24
H	1	5	4



$r_{incumb,bj} = 36 \quad 44 \quad 25$

$\Delta_{FB} = 36 - 26 = 10 \qquad \Delta_{FD} = 44 - 26 = 18 \qquad \Delta_{FH} = 25 - 26 = -1$

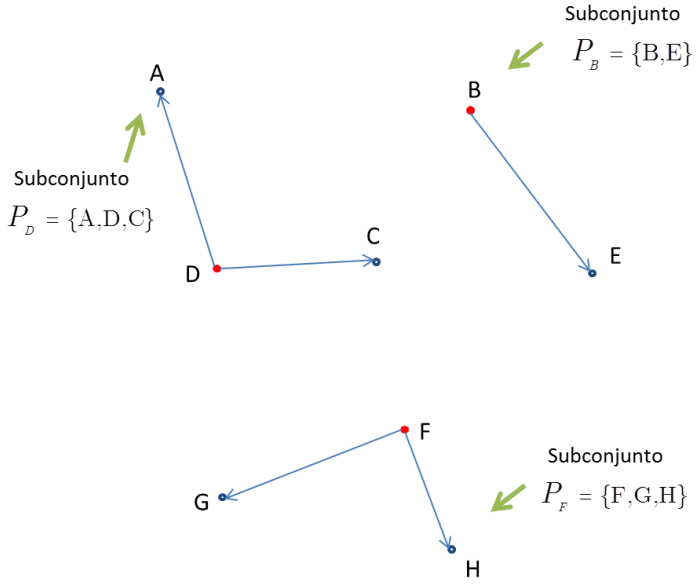
3) Hay algún incremento $(\Delta_{FB}, \Delta_{FD}, \Delta_{FH})$ que sea negativo ? → **SI** → Escoger el menor o máximo negativo, y hacer el intercambio del vértice j por el vértice b

→ $\min(\Delta_{bj}) = \Delta_{FH}$
$V^t_{incumb} = \{B, D, H\};$
$V^{t+1}_{incumb} = \{B, D, F\};$
→ **Ir a 2)**

V_{incumb} =

	F	B	D
A	15	12	6
B	16	0	12
C	4	4	2
D	12	12	0
E	5	4	4
F	0	8	6
G	12	28	24
H	1	5	4

$r_{incumb} = 25$



2)

	D	B	G
A	6	12	24
B	12	0	28
C	2	4	10
D	0	12	24
E	4	4	8
F	6	8	6
G	24	28	0
H	4	5	3

$r_{incumb,bj} = 21$

$\Delta_{GF} = 21 - 25 = -4$

	B	F	G
A	12	15	24
B	0	16	28
C	4	4	10
D	12	12	24
E	4	5	8
F	8	0	6
G	28	12	0
H	5	1	3

33

$\Delta_{GD} = 33 - 25 = 8$

	F	G	D
A	15	24	6
B	16	28	12
C	4	10	2
D	12	24	0
E	5	8	4
F	0	6	6
G	12	0	24
H	1	3	4

25

$\Delta_{GB} = 25 - 25 = 0$

v_b

	A	B	C	D	E	F	G	H
A	0	12	9	6	18	15	24	18
B	16	0	8	12	16	16	28	20
C	6	4	0	2	6	4	10	6
D	8	12	4	0	16	12	24	16
E	6	4	3	4	0	5	8	6
F	10	8	4	6	10	0	6	2
G	32	28	20	24	32	12	0	12
H	6	5	3	4	6	1	3	0

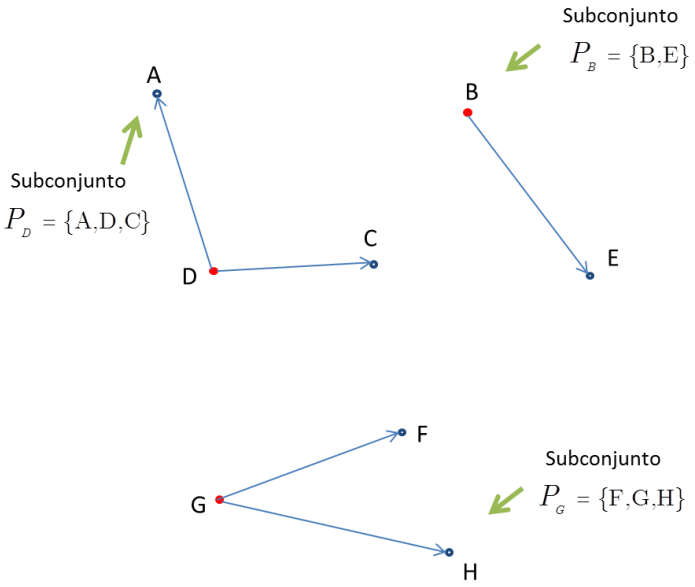
3) Hay algún incremento $(\Delta_{GF},\Delta_{GD},\Delta_{GB})$ que sea negativo ? → **SI** → Escoger el menor o máximo negativo, y hacer el intercambio del vértice j por el vértice b

→ $\min(\Delta_{bj}) = \Delta_{GF}$
$V^t_{incumb} = \{B,D,F\};$
$V^{t+1}_{incumb} = \{B,D,G\};$ → **Ir a 2)**

V_{incumb} =

	D	B	G
A	6	12	24
B	12	0	28
C	2	4	10
D	0	12	24
E	4	4	8
F	6	8	6
G	24	28	0
H	4	5	3

$r_{incumb} = 21$



2)

	D	B	A
A	6	12	0
B	12	0	16
C	2	4	6
D	0	12	8
E	4	4	6
F	6	8	10
G	24	28	32
H	4	5	6

$r_{incumb,bj} = 40$

$\Delta_{AG} = 40 - 21 = 19$

	B	A	G
A	12	0	24
B	0	16	28
C	4	6	10
D	12	8	24
E	4	6	8
F	8	10	6
G	28	32	0
H	5	6	3

25

$\Delta_{AD} = 25 - 21 = 4$

	A	G	D
A	0	24	6
B	16	28	12
C	6	10	2
D	8	24	0
E	6	8	4
F	10	6	6
G	32	0	24
H	6	3	4

27

$\Delta_{AB} = 27 - 21 = 6$

v_b

	A	B	C	D	E	F	G	H
A	0	12	9	6	18	15	24	18
B	16	0	8	12	16	16	28	20
C	6	4	0	2	6	4	10	6
D	8	12	4	0	16	12	24	16
E	6	4	3	4	0	5	8	6
F	10	8	4	6	10	0	6	2
G	32	28	20	24	32	12	0	12
H	6	5	3	4	6	1	3	0

3) Hay algún incremento $(\Delta_{AB},\Delta_{AD},\Delta_{AG})$ que sea negativo ? → **NO** → Rotular *A* como intentado. Existen mas vértices no intentados?

→ **SI** → **Ir a 2)**

2)

	D	B	C
A	6	12	9
B	12	0	8
C	2	4	0
D	0	12	4
E	4	4	3
F	6	8	4
G	24	28	20
H	4	5	3

	B	C	G
A	12	9	24
B	0	8	28
C	4	0	10
D	12	4	24
E	4	3	8
F	8	4	6
G	28	20	0
H	5	3	3

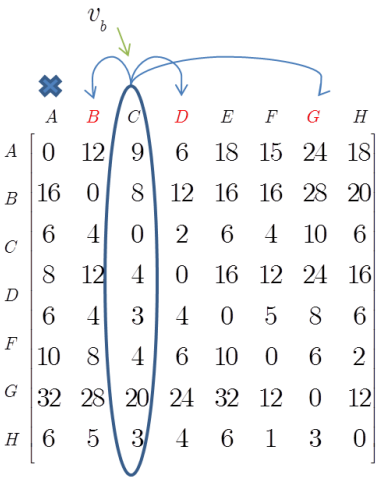
	C	G	D
A	9	24	6
B	8	28	12
C	0	10	2
D	4	24	0
E	3	8	4
F	4	6	6
G	20	0	24
H	3	3	4

$r_{incumb,bj} = 36$ | 23 | 26

$\Delta_{CG} = 36 - 21 = 15$

$\Delta_{CD} = 23 - 21 = 2$

$\Delta_{CB} = 26 - 21 = 5$



3) Hay algún incremento $(\Delta_{CB},\Delta_{CD},\Delta_{CG})$ que sea negativo ? → **NO** → Rotular *A* como intentado. Existen mas vértices no intentados?

→ **SI** → **Ir a 2)**

2)

	D	B	E
A	6	12	18
B	12	0	16
C	2	4	6
D	0	12	16
E	4	4	0
F	6	8	10
G	24	28	32
H	4	5	6

	B	E	G
A	12	18	24
B	0	16	28
C	4	6	10
D	12	16	24
E	4	0	8
F	8	10	6
G	28	32	0
H	5	6	3

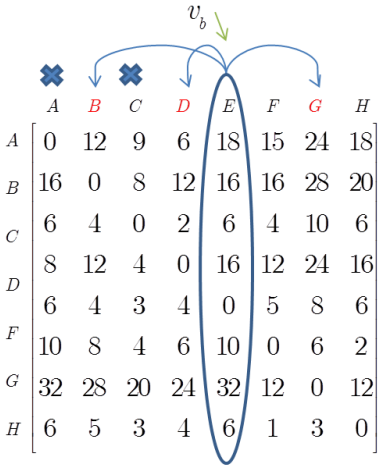
	E	G	D
A	18	24	6
B	16	28	12
C	6	10	2
D	16	24	0
E	0	8	4
F	10	6	6
G	32	0	24
H	6	3	4

$r_{incumb,bj} = 42$ | 37 | 29

$\Delta_{EG} = 42 - 21 = 21$

$\Delta_{ED} = 37 - 21 = 16$

$\Delta_{EB} = 29 - 21 = 8$



3) Hay algún incremento $(\Delta_{EB},\Delta_{ED},\Delta_{EG})$ que sea negativo ? → **NO** → Rotular *A* como intentado. Existen mas vértices no intentados?

→ **SI** → **Ir a 2)**

2)

	D	B	F
A	6	12	15
B	12	0	16
C	2	4	4
D	0	12	12
E	4	4	5
F	6	8	0
G	24	28	12
H	4	5	1

$r_{incumb,bj} = 25$

$\Delta_{FG} = 25 - 21 = 4$

	B	F	G
A	12	15	24
B	0	16	28
C	4	4	10
D	12	12	24
E	4	5	8
F	8	0	6
G	28	12	0
H	5	1	3

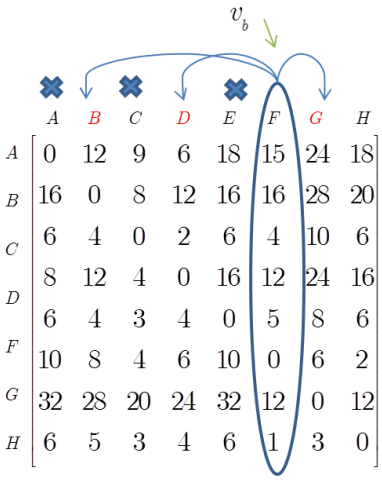
29

$\Delta_{FD} = 29 - 21 = 8$

	F	G	D
A	15	24	6
B	16	28	12
C	4	10	2
D	12	24	0
E	5	8	4
F	0	6	6
G	12	0	24
H	1	3	4

25

$\Delta_{FB} = 25 - 21 = 4$



3) Hay algún incremento $(\Delta_{FB},\Delta_{FD},\Delta_{FG})$ que sea negativo ? → **NO** → Rotular *A* como intentado. Existen mas vértices no intentados?

→ **SI** → **Ir a 2)**

2)

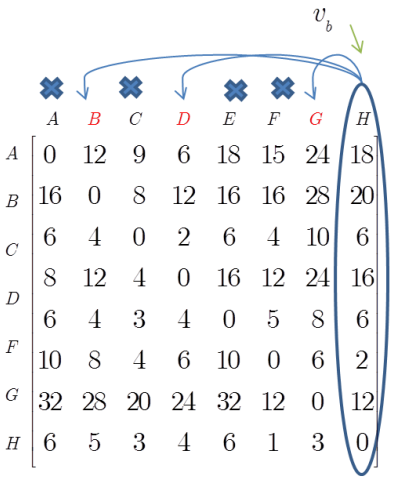
	D	B	H
A	6	12	18
B	12	0	20
C	2	4	6
D	0	12	16
E	4	4	6
F	6	8	2
G	24	28	12
H	4	5	0

	B	H	G
A	12	18	24
B	0	20	28
C	4	6	10
D	12	16	24
E	4	6	8
F	8	2	6
G	28	12	0
H	5	0	3

	H	G	D
A	18	24	6
B	20	28	12
C	6	10	2
D	16	24	0
E	6	8	4
F	2	6	6
G	12	0	24
H	0	3	4

$r_{incumb,bj} = 26$ $\quad$ 34 $\quad$ 24

$\triangle_{HG} = 26 - 21 = 5$ $\quad$ $\triangle_{HD} = 34 - 21 = 13$ $\quad$ $\triangle_{HB} = 24 - 21 = 3$



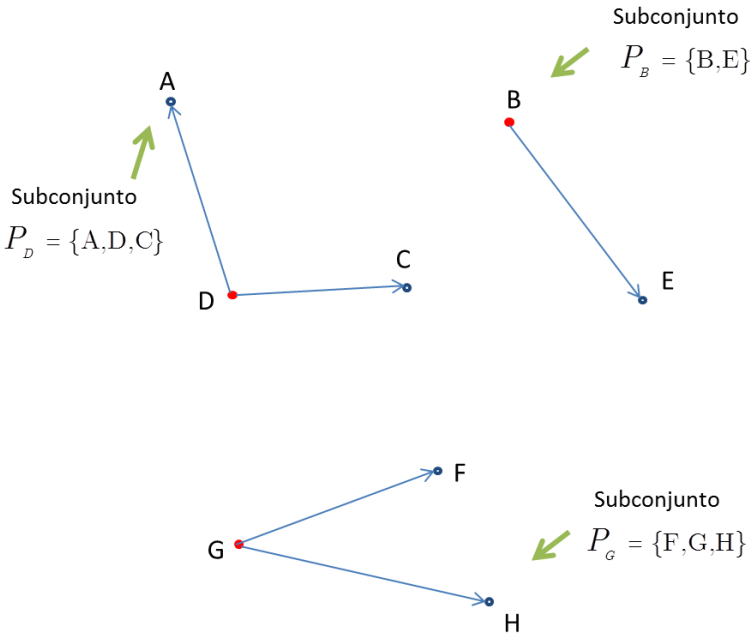
3) Hay algún incremento $(\triangle_{HG}, \triangle_{HD}, \triangle_{HB})$ que sea negativo ? → **NO** → Rotular *H* como intentado. Existen mas vértices no intentados?

→ **NO** → Se completo un ciclo sin hacer alguna sustitución → **Parar/Terminar**

V_{incumb} =

	D	B	G
A	6	12	24
B	12	0	28
C	2	4	10
D	0	12	24
E	4	4	8
F	6	8	6
G	24	28	0
H	4	5	3

$r_{incumb} = 21$



3.2.3. Pseudocódigo de la Decodificación

Entrada: vector x de $n*n$ celdas, que representa una partícula.

Salida: vector $x_{decodificada}$ de p celdas que representa el elemento mapeado del vector x

1. Leer vector x de tamaño $n*n$.
2. Crear arreglo de dos dimensiones de tamaño n. La primera dimensión corresponde al vector de valores enteros (x_{val_ent}), y la segunda dimensión corresponde al vector de índices de los valores enteros ($indices$), esto es, los valores desde 1 hasta n.
   - Para crear el primer vector del arreglo (primera dimensión), se calcula cada celda con la siguiente formula. $x_{val\_ent}(i) = \sum_{j=0}^{n} x(i + jn), \quad \forall i = 1,.....n$
3. Ordenar el vector x_{val_ent} en forma ascendente. Utilizar la misma secuencia de ordenar (de x_{val_ent}) sobre el vector $indices$ para crear una permutación ($indices_{per}$).
   - El orden de la permutación indica cuales vértices formaran parte del conjunto de medianas y cuáles no.
4. Escoger las primeras p celdas de $indices_{per}$ y crear vector $medianas$. El vector $medianas$ contiene un conjunto de medianas factible del problema. (sin embargo, puede haber empate en el vector de valores enteros y se debe aplicar el paso (5).
5. Evaluar el vector x_{val_ent} para saber si existe más de una celda con el valor $x_{val\_ent}(p)$. Si no existe ir al paso (6). Si existe más de una celda con el mismo valor $x_{val\_ent}(p)$, identificar cuales celdas y separar el conjunto de celdas del vector $indices_{per}$. Este subconjunto de $indices_{per}$ se llama $med\_posible$. Así mismo, con los

índices que se escogieron en el paso (4) pero no empataron en el paso (5) crear el vector de medianas definitivas $med\_definitva$.

Aplicar el método de reparación de Greedy para resolver el siguiente subproblema de la p-mediana: La solución inicial (no factible) tiene el conjunto de medianas $med\_definitva$, por lo tanto se necesitan localizar $p-|med\_definitiva|$ medianas; el conjunto de índices donde se pueden localizar las medianas faltantes son los vértices $med\_posible$, y el conjunto de clientes es el conjunto del problema original de tamaño n. La pregunta es: dentro del conjunto de vértice $med\_posible$, cuales entran a las mediana y cuáles no?EL resultado de la unión entre $med\_definitva$ y el conjunto de medianas incluido en Greedy se denomina $medianas$, que constituye una solución factible del problema original.

6. Escoger el vector $medianas$ del paso (4) o (5) como solución inicial para aplicar la heuristica Vertex Substituion. El resultado se denomina $x_{decodificada}$, y contiene el conjunto de medianas que corresponden a la partícula x.

4. Resultados Computacionales

Para analizar el desempeño de la heurística, se utilizó 40 problemas representativos de la librería OR-Library. Los 40 problemas están divididos en 9 grupos desde 100 vértices hasta 900 vértices. Así mismo dentro de cada grupo el número de medianas varía en proporción a la cantidad de vértices. Cada problema fue resuelto con 3, 6 y 9 tamaño de enjambre y 5,10,15 y 20 número de iteraciones. La herramienta utilizada es el software MATLAB versión R2011a, en un equipo estándar con procesador Intel Core i3 con 4 GB de memoria RAM instalada.

Con las soluciones obtenidas mediante el algoritmo QPSO y la solución óptima de cada problema, se calculó el porcentaje de diferencia entre dichas soluciones.

4.1. Desempeño del Algoritmo

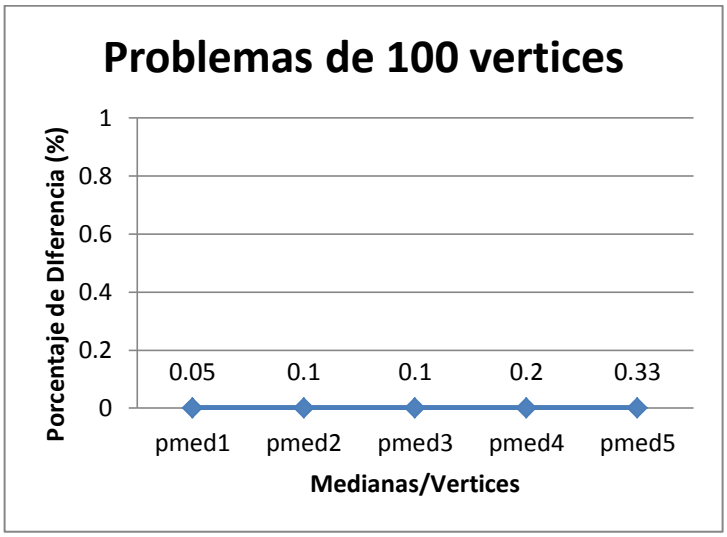
4.1.1. Problemas de 100 vértices

Tabla 1. Porcentaje de Diferencia para problemas de 100 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed1	5	0.05	5819	5819	0
pmed2	10	0.10	4093	4093	0
pmed3	10	0.10	4250	4250	0
pmed4	20	0.20	3034	3034	0
pmed5	33	0.33	1355	1355	0

Fuente: Autor

Figura 25. Porcentaje de Diferencia vs Porcentaje de Medianas (100 vertices)



Fuente: Autor

Se encontró el óptimo para todos los casos, así mismo todas las combinaciones de tamaño de enjambre y número de iteraciones produjeron la solución óptima, por lo tanto se observa un excelente desempeño del algoritmo para este tamaño de problemas, incluso cuando el número de medianas es aproximadamente un tercio del número de vértices.

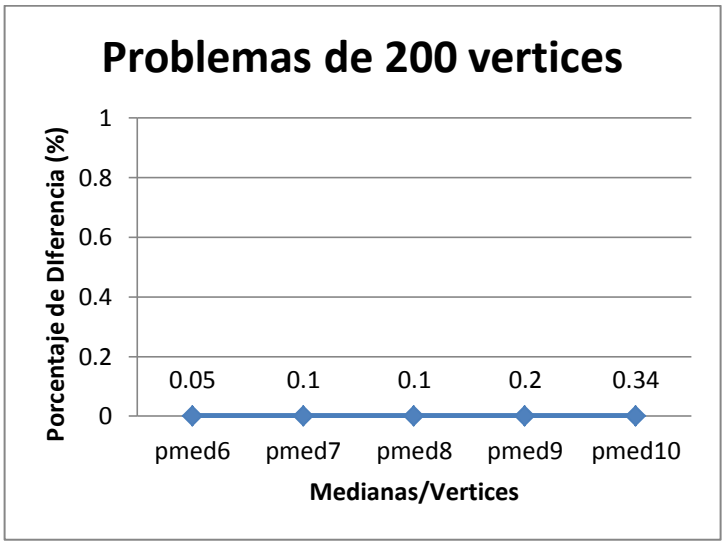
4.1.2. Problemas de 200 vértices

Tabla 2. Porcentaje de Diferencia para problemas de 200 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed6	5	0.03	7824	7824	0
pmed7	10	0.05	5631	5631	0
pmed8	20	0.10	4445	4445	0
pmed9	40	0.20	2734	2734	0
pmed10	67	0.34	1255	1255	0

Fuente: Autor

Figura 26. Porcentaje de Diferencia vs Porcentaje de Medianas (200 vertices)



Fuente: Autor

Para este tipo de problemas también se encontró el óptimo en todos los casos. Para los primero cuatro problemas todas las combinaciones de tamaño de enjambre – número de iteraciones produjeron el valor óptimo. Para el caso de 67 medianas se produjo un error para la combinación 3-5, sin embargo, para valores mayores de los parámetros se encontró el óptimo en todos los casos.

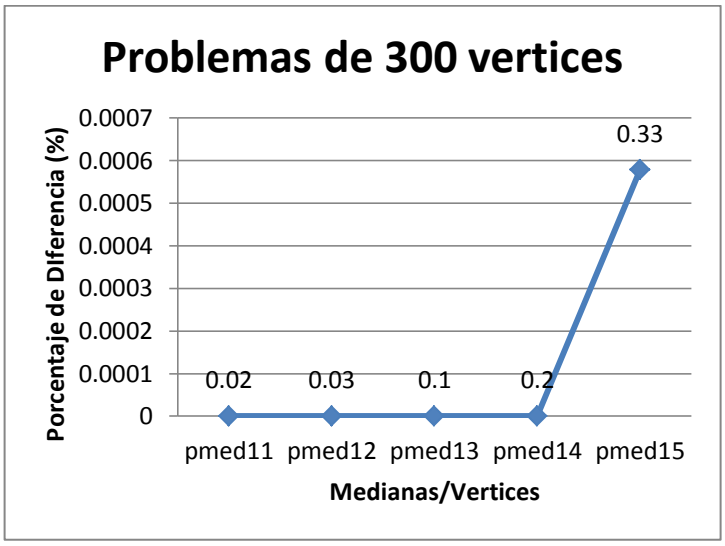
4.1.3. Problemas de 300 vértices

Tabla 3. Porcentaje de Diferencia para problemas de 300 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed11	5	0.02	7696	7696	0
pmed12	10	0.03	6634	6634	0
pmed13	30	0.10	4374	4374	0
pmed14	60	0.20	2968	2968	0
pmed15	100	0.33	1729	1730	0.000578369

Fuente: Autor

Figura 27. Porcentaje de Diferencia vs Porcentaje de Medianas (300 vertices)



Fuente: Autor

Se encontró el óptimo para todas las combinaciones de los parámetros en los primeros tres casos. En el cuarto problema se encontró el valor optimo utilizando los valores más altos de los parámetros. En el quinto problema, no se encontró en ningún caso el valor optimo, sin embargo, el mejor valor encontrado, utilizando los valores más altos de los parámetros, difiere muy poco de valor optimo real.

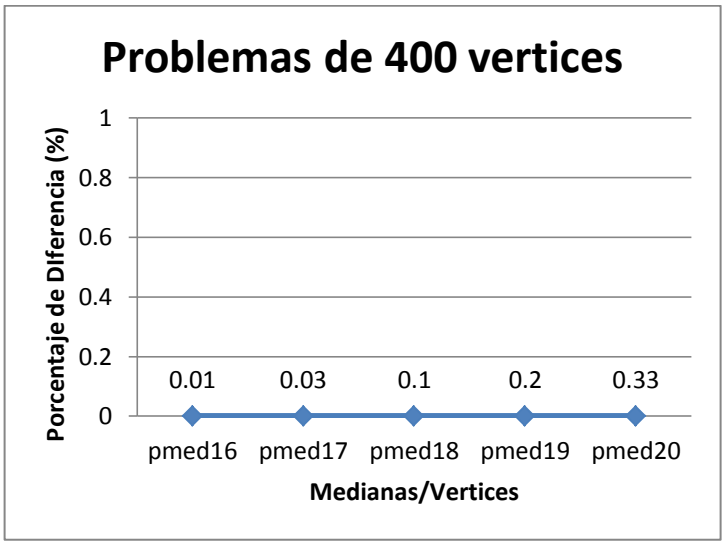
4.1.4. Problemas de 400 vértices

Tabla 4. Porcentaje de Diferencia para problemas de 400 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed16	5	0.01	8162	8162	0
pmed17	10	0.03	6999	6999	0
pmed18	40	0.10	4809	4809	0
pmed19	80	0.20	2845	2845	0
pmed20	133	0.33	1789	1789	0

Fuente: Autor

Figura 28. Porcentaje de Diferencia vs Porcentaje de Medianas (400 vertices)



Fuente: Autor

Para ese tipo de problemas se encontró el valor óptimo para todas las combinaciones de los parámetros en los primeros tres casos. En el cuarto y quinto problema se halló el óptimo utilizando los valores más altos de los parámetros.

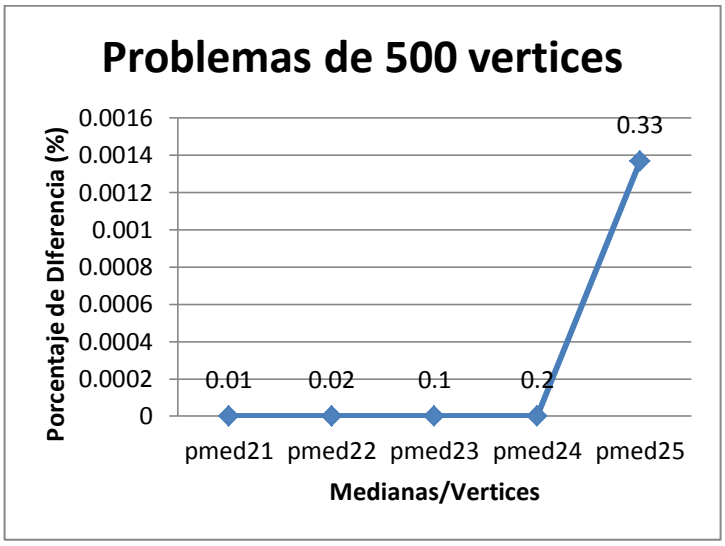
4.1.5. Problemas de 500 vértices

Tabla 5. Porcentaje de Diferencia para problemas de 500 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed21	5	0.01	9138	9138	0
pmed22	10	0.02	8579	8579	0
pmed23	50	0.10	4619	4619	0
pmed24	100	0.20	2961	2961	0
pmed25	167	0.33	1828	1830.5	0.001367615

Fuente: Autor

Figura 29. Porcentaje de Diferencia vs Porcentaje de Medianas (500 vertices)



Fuente: Autor

En este conjunto de problemas se halló el valor óptimo para todas las combinaciones de los parámetros en los primeros tres casos. En el cuarto caso se halló el valor optimo utilizando los valores más altos de los parámetros, y en el quinto caso, aunque no se encontró el valor optimo, el porcentaje de diferencia, utilizando valores altos para los parámetros, es mínimo.

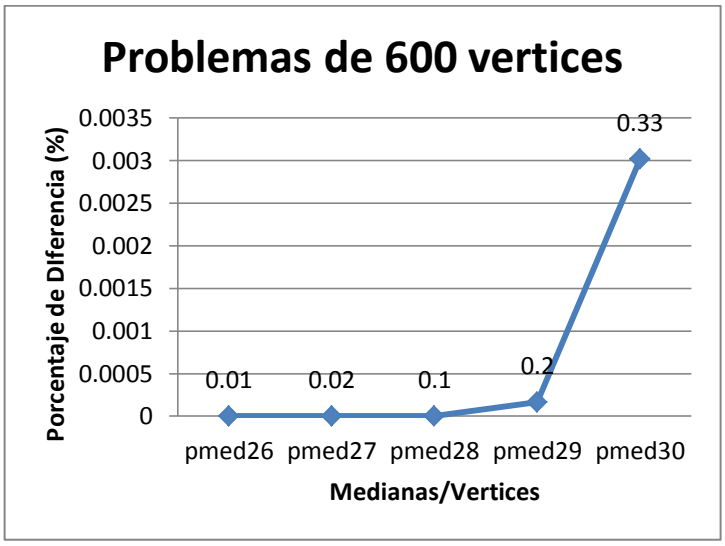
4.1.6. Problemas de 600 vértices

Tabla 6. Porcentaje de Diferencia para problemas de 600 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed26	5	0.01	9917	9917	0
pmed27	10	0.02	8307	8307	0
pmed28	60	0.10	4498	4498	0
pmed29	120	0.20	3033	3033.5	0.000164853
pmed30	200	0.33	1989	1995	0.003016591

Fuente: Autor

Figura 30. Porcentaje de Diferencia vs Porcentaje de Medianas (600 vertices)



Fuente: Autor

En este conjunto de problemas se encontró el valor óptimo para todos los valores de los parámetros en los primeros dos casos. En el tercer caso se encontró el valor óptimo, para algunas combinaciones de los parámetros. En el cuarto y quinto caso, aunque no se encontró el valor óptimo, se obtuvieron buenas aproximaciones para los valores más altos de los parámetros.

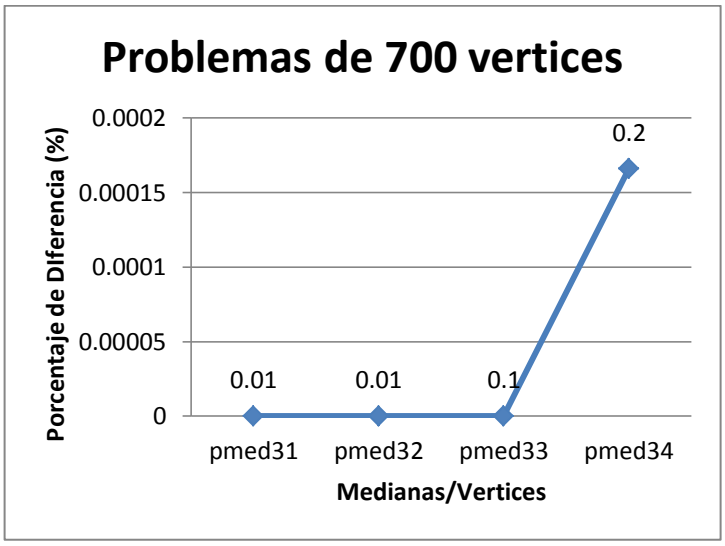
4.1.7. Problemas de 700 vértices

Tabla 7. Porcentaje de Diferencia para problemas de 700 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed31	5	0.01	10086	10086	0
pmed32	10	0.01	9297	9297	0
pmed33	70	0.10	4700	4700	0
pmed34	140	0.20	3013	3013.5	0.000165948

Fuente: Autor

Figura 31. Porcentaje de Diferencia vs Porcentaje de Medianas (700 vertices)



Fuente: Autor

Para este conjunto de problemas se encontró el valor óptimo para todas la combinaciones en los primeros dos casos. En el tercero, se halló el valor óptimo utilizando valores altos para los parámetros. En el cuarto caso, se encontró una buena aproximación del valor optimo utilizando los valores más altos para los parámetros.

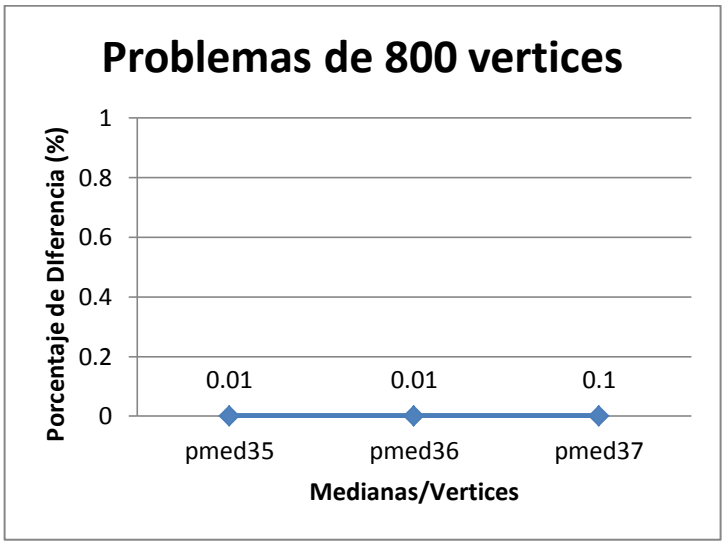
4.1.8. Problemas de 800 vértices

Tabla 8. Porcentaje de Diferencia para problemas de 800 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed35	5	0.01	10400	10400	0
pmed36	10	0.01	9934	9934	0
pmed37	80	0.10	5057	5057	0

Fuente: Autor

Figura 32. Porcentaje de Diferencia vs Porcentaje de Medianas (800 vertices)



Fuente: Autor

En este conjunto de problemas se encontró el valor óptimo para todas la combinaciones de los parámetros en los primeros dos casos. En el tercer caso, se encontró el valor óptimo utilizando valores altos de los parámetros.

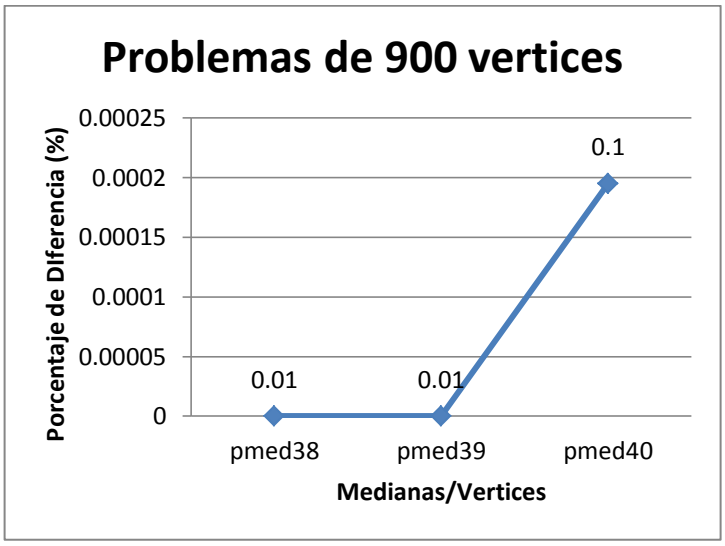
4.1.9. Problemas de 900 vértices

Tabla 9. Porcentaje de Diferencia para problemas de 900 vértices

Problema	Medianas	Medianas/Vértices	Optimo	Mejor Encontrado	Porcentaje de Diferencia
pmed38	5	0.01	11060	11060	0
pmed39	10	0.01	9423	9423	0
pmed40	90	0.10	5128	5129	0.000195008

Fuente: Autor

Figura 33. Porcentaje de Diferencia vs Porcentaje de Medianas (900 vertices)



Fuente: Autor

Para este conjunto de problemas se encontró el valor óptimo para todas la combinaciones de los parámetros en los primeros dos casos. En el tercer caso se encontró una buena aproximación para los valores más altos de los parámetros.

4.2. Análisis de Efectos Principales de los Parámetros del Algoritmo

Dado que el desempeño del algoritmo para los primero problemas fue muy bueno, con desviaciones mínimas respecto al óptimo, se escogieron 4 problemas significativos para el diseño de experimentos: pmed33, pmed34, pmed37 y pmed40.

Con el fin de determinar la influencia de los parámetros utilizados en el algoritmo QPSO sobre el resultado obtenido en la ejecución del mismo se realizó el siguiente estudio. Se identificaron cuatro parámetros del algoritmo y se hizo una

pequeña métrica para hacer un análisis estadístico factorial de efectos fijos en cada uno de los cuatro problemas.

La variable de salida está dada por el índice de error porcentual (desviación del optimo) entre las soluciones obtenidas mediante el algoritmo QPSO propuesto y las soluciones optimas suministradas por la librería OR-Library. Asi,

$$DesvOpt = \frac{Z_{QPSO} - Z_{optimo}}{Z_{optimo}}$$

Los factores que se consideraron determinantes y sus respectivos niveles son:

- Numero de Iteraciones = { 5, 10, 15, 20}.
- Tamaño del Enjambre = { 3, 6, 9 }
- α / β (dado que $\alpha + \beta = 1$) :

Tabla 10. Niveles del Factor α / β

Niveles	**Niveles(α / β)**	α	β
1	0.25/0.75	Bajo	Alto
2	0.5/0.5	Igual	Igual
3	0.75/0.25	Alto	Bajo

Fuente: Autor

- $c_1 / c_2 / c_3$ (dado que $c_1 + c_2 + c_3 = 1$):

Tabla 11. Niveles del Factor $c_1 / c_2 / c_3$

Niveles	**Niveles ($c_1 / c_2 / c_3$)**	c_1	c_2	c_3
1	0.4/0.4/0.2	Alto	Alto	Bajo
2	0.2/0.4/0.4	Bajo	Alto	Alto
3	0.4/0.2/0.4	Alto	Bajo	Alto
4	0.6/0.2/0.2	Alto	Bajo	Bajo
5	0.2/0.6/0.2	Bajo	Alto	Bajo
6	0.2/0.2/0.6	Bajo	Bajo	Alto

Fuente: Autor

El modelo estadístico lineal para describir las observaciones obtenidas es el siguiente:

$$\begin{aligned} &y_{ijklm} = \mu + \tau_i + \beta_j + \rho_k + \gamma_l + \dots \\ &(\tau\beta)_{ij} + (\beta\rho)_{jk} + (\rho\gamma)_{kl} + (\tau\gamma_{il}) + (\tau\rho_{ik}) + (\beta\gamma_{jl}) + \dots \\ &(\tau\beta\rho)_{ijk} + (\beta\rho\gamma)_{jkl} + (\tau\rho\gamma)_{ikl} + (\tau\beta\gamma)_{ijl} + \dots \\ &(\tau\beta\rho\gamma)_{ijkl} + \varepsilon_{ijklm} \end{aligned} \quad \begin{cases} i = 5,10,15 \text{ y } 20 \\ j = 3,6 \text{ y } 9 \\ k = 1,2 \text{ y } 3 \\ l = 1,2,3,4,5 \text{ y } 6 \\ m = 1,2 \text{ y } 3 \end{cases}$$

Dónde:

y_{ijklm}: Respuesta observada cuando el factor número de iteraciones está en el $i-esimo$ nivel, el factor tamaño de enjambre está en el $j-esimo$ nivel, el factor α / β en el $k-esimo$ nivel y el factor $c_1 / c_2 / c_3$ en el $l-esimo$ nivel, durante la $m-esima$ réplica de cada combinación.

μ: Efecto del Promedio Global.

τ_i: Efecto del nivel $i-esimo$ del número de iteraciones.

β_j : Efecto del nivel $j-esimo$ del tamaño del enjambre.

ρ_k : Efecto del nivel $k-esimo$ de las constantes α / β .

γ_l : Efecto del nivel $l-esimo$ de las constantes c_1 / c_2 / c_3 .

$(\tau\beta)_{ij}$: Efecto de interacción entre los factores τ_i y β_j .

$(\beta\rho)_{jk}$:Efecto de interacción entre los factores β_j y ρ_k .

$(\rho\gamma)_{kl}$:Efecto de interacción entre los factores ρ_k y γ_l .

$(\tau\gamma)_{il}$:Efecto de interacción entre los factores τ_i y γ_l .

$(\tau\rho)_{ik}$:Efecto de interacción entre los factores τ_i y ρ_k .

$(\beta\gamma)_{jl}$:Efecto de interacción entre los factores β_j y γ_l .

$(\tau\beta\rho)_{ijk}$: Efecto de interacción entre los factores τ_i , β_j y ρ_k

$(\beta\rho\gamma)_{jkl}$: Efecto de interacción entre los factores β_j , ρ_k y γ_l

$(\tau\rho\gamma)_{ikl}$: Efecto de interacción entre los factores τ_i , ρ_k y γ_l

$(\tau\beta\gamma)_{ijl}$: Efecto de interacción entre los factores τ_i , β_j y γ_l

$(\tau\beta\rho\gamma)_{ijkl}$:Efecto de interacción entre los factores τ_i , β_j , ρ_k y γ_l

ε_{ijklm} : Componente de error aleatorio

Con el experimento se quiere probar el siguiente grupo de Hipótesis:

1) $H_0 : \tau_5 = \tau_{10} = \tau_{15} = \tau_{20}$ / $H_1 :$ al menos una $\tau_i \neq 0$

2) $H_0: \beta_3 = \beta_6 = \beta_9$ / $H_1:$ al menos una $\beta_j \neq 0$

3) $H_0: \rho_1 = \rho_2 = \rho_3$ / $H_1:$ al menos una $\rho_k \neq 0$

4) $H_0: \gamma_1 = \gamma_2 = \gamma_3 = \gamma_4 = \gamma_5 = \gamma_6$ / $H_1:$ al menos una $\gamma_l \neq 0$

5) $H_0: (\tau\beta)_{ij} = 0 \ \forall i, j.$ / $H_1:$ al menos un $(\tau\beta)_{ij} \neq 0$

6) $H_0: (\beta\rho)_{jk} = 0 \ \forall j, k.$ / $H_1:$ al menos un $(\beta\rho)_{jk} \neq 0$

7) $H_0: (\rho\gamma)_{kl} = 0 \ \forall k, l.$ / $H_1:$ al menos un $(\rho\gamma)_{kl} \neq 0$

8) $H_0: (\tau\gamma)_{il} = 0 \ \forall i, l.$ / $H_1:$ al menos un $(\tau\gamma)_{il} \neq 0$

9) $H_0: (\tau\rho)_{ik} = 0 \ \forall i, k.$ / $H_1:$ al menos un $(\tau\rho)_{ik} \neq 0$

10) $H_0: (\beta\gamma)_{jl} = 0 \ \forall j, l.$ / $H_1:$ al menos un $(\beta\gamma)_{jl} \neq 0$

11) $H_0: (\tau\beta\rho)_{ijk} = 0 \ \forall i, j, k.$ / $H_1:$ al menos un $(\tau\beta\rho)_{ijk} \neq 0$

12) $H_0: (\beta\rho\gamma)_{jkl} = 0 \ \forall j, k, l.$ / $H_1:$ al menos un $(\beta\rho\gamma)_{jkl} \neq 0$

13) $H_0: (\tau\rho\gamma)_{ikl} = 0 \ \forall i, k, l.$ / $H_1:$ al menos un $(\tau\rho\gamma)_{ikl} \neq 0$

14) $H_0: (\tau\beta\gamma)_{ijl} = 0 \ \forall i, j, l.$ / $H_1:$ al menos un $(\tau\beta\gamma)_{ijl} \neq 0$

15) $H_0: (\tau\beta\rho\gamma)_{ijkl} = 0 \ \forall i, j, k, l.$ / $H_1:$ al menos un $(\tau\beta\rho\gamma)_{ijkl} \neq 0$

A continuación se presenta los resultados del análisis de varianza realizado con ayuda de la herramienta SSPS v20.

4.2.1. Análisis de Varianza (ANOVA) para el problema pmed33

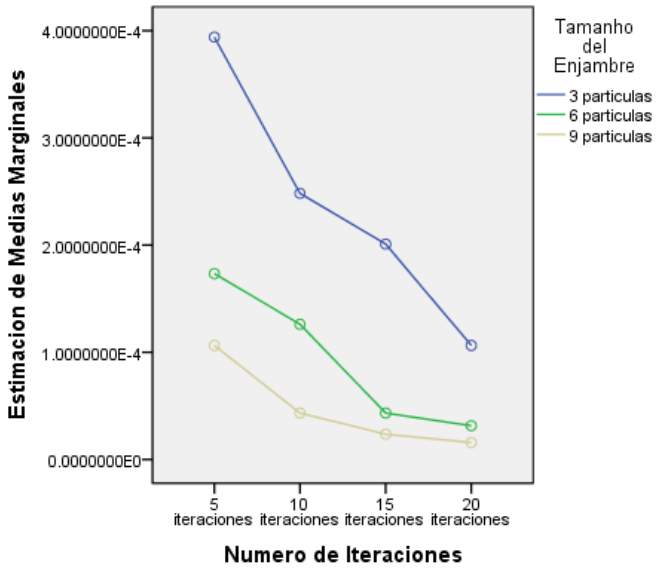
Tabla 12. Prueba de Efectos entre Factores para el problema pmed33

Prueba de Efectos entre Factores					
Variable Dependiente: Desviación del Optimo					
Fuente	Suma de Cuadrados tipo III	Grados de Libertad	Cuadrado Medio	F	Significancia

Modelo Corregido	1.647E-005[a]	215	7.660E-08	1.861	.000
Intersección	1.030E-05	1	1.030E-05	250.350	.000
Iteraciones	2.727E-06	3	9.089E-07	22.089	.000
Enjambre	4.246E-06	2	2.123E-06	51.591	.000
AlfaBeta	2.465E-07	2	1.232E-07	2.995	.051
Const_C	1.601E-07	5	3.202E-08	.778	.566
Iteraciones * Enjambre	6.211E-07	6	1.035E-07	2.516	.021
Iteraciones * AlfaBeta	1.436E-07	6	2.394E-08	.582	.745
Iteraciones * Const_C	7.984E-07	15	5.322E-08	1.293	.202
Enjambre * AlfaBeta	2.448E-07	4	6.120E-08	1.487	.205
Enjambre * Const_C	6.317E-07	10	6.317E-08	1.535	.124
AlfaBeta * Const_C	3.554E-07	10	3.554E-08	.864	.567
Iteraciones * Enjambre * AlfaBeta	2.923E-07	12	2.436E-08	.592	.849
Iteraciones * Enjambre * Const_C	2.365E-06	30	7.883E-08	1.916	.003
Iteraciones * AlfaBeta * Const_C	9.412E-07	30	3.137E-08	.762	.815
Enjambre * AlfaBeta * Const_C	7.151E-07	20	3.575E-08	.869	.627
Iteraciones * Enjambre * AlfaBeta * Const_C	1.980E-06	60	3.301E-08	.802	.853
Error	1.778E-05	432	4.115E-08		
Total	4.455E-05	648			
Total Corregido	3.424E-05	647			
a. R Cuadrado = 0.481 (R Cuadrado Ajustado = 0 .223)					

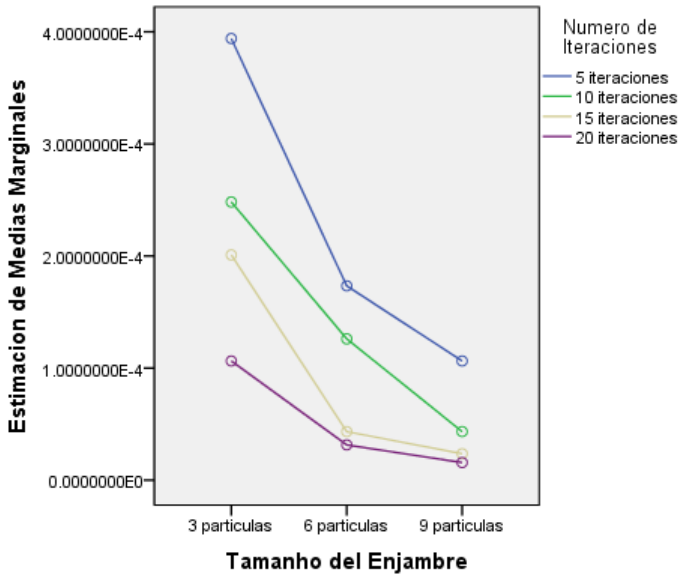
Fuente: Autor

Figura 34. Medias Marginales de la Variable de Respuesta (pmed33) – Tamanho de Enjambre vs Numero de Iteraciones



Fuente: Autor

Figura 35. Medias Marginales de la Variable de Respuesta (pmed33) – Numero de Iteraciones vs Tamanho de Enjambre



Fuente: Autor

De la tabla anterior se puede concluir que no existe interacción significativa entre todos los grupos de factores, excepto por la combinación Iteraciones-Enjambre-C1/C2/C3 cuya significancia (0.003) es menor que α(0.05) por lo tanto hay evidencia para rechazar H_0. Así mismo, se concluye que el efecto producido por los factores Iteraciones y Enjambre es significativo, sin embargo, no hay evidencia de que los factores Alfa/Beta y C1/C2/C3 tengan un efecto significativo sobre la variable salida como lo muestra su valor de significancia (mayor que α(0.05)). En las Figuras 34 y 35 se muestra el comportamiento de los factores que se afirman tiene efecto significativo sobre la variable, así mismo las dos graficas confirman que no existe interacción entre estos dos factores, ya que a medida que aumenta los niveles de un factor la variable de respuesta disminuye independientemente del nivel del otro factor.

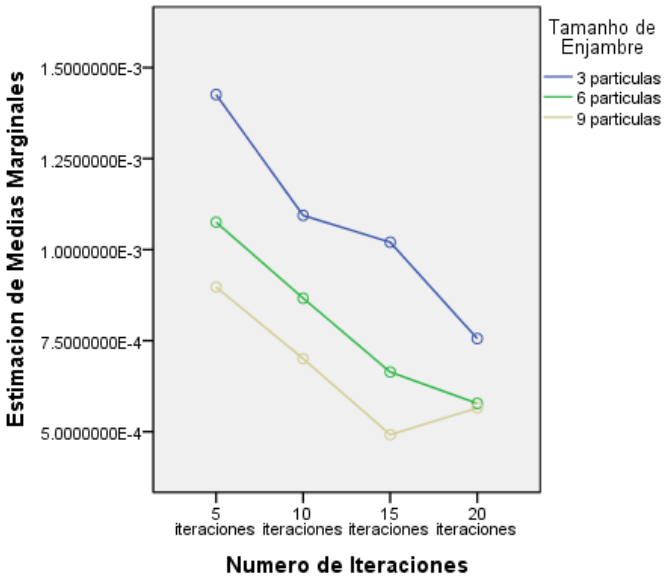
4.2.2. Análisis de Varianza (ANOVA) para el problema pmed34

Tabla 13. Prueba de Efectos entre Factores para el problema pmed34

Prueba de Efectos entre Factores					
Variable Dependiente: Desviación del Optimo					
Fuente	Suma de Cuadrados tipo III	Grados de Libertad	Cuadrado Medio	F	Significancia
Modelo Corregido	9.239E-005[a]	215	4.297E-07	1.840	.000
Intersección	4.622E-04	1	4.622E-04	1979.039	.000
Iteraciones	2.332E-05	3	7.773E-06	33.280	.000
Enjambre	1.894E-05	2	9.472E-06	40.555	.000
AlfaBeta	1.704E-06	2	8.518E-07	3.647	.027
Const_C	2.742E-07	5	5.484E-08	.235	.947
Iteraciones * Enjambre	2.154E-06	6	3.591E-07	1.537	.164
Iteraciones * AlfaBeta	1.632E-06	6	2.720E-07	1.165	.324
Iteraciones * Const_C	2.692E-06	15	1.795E-07	0.769	.713
Enjambre * AlfaBeta	1.884E-07	4	4.709E-08	0.202	.937
Enjambre * Const_C	1.931E-06	10	1.931E-07	0.827	.603
AlfaBeta * Const_C	1.805E-06	10	1.805E-07	.773	.655
Iteraciones * Enjambre * AlfaBeta	2.200E-06	12	1.833E-07	.785	.666
Iteraciones * Enjambre * Const_C	7.086E-06	30	2.362E-07	1.011	.452
Iteraciones * AlfaBeta * Const_C	6.764E-06	30	2.255E-07	.965	.521
Enjambre * AlfaBeta * Const_C	4.742E-06	20	2.371E-07	1.015	.442
Iteraciones * Enjambre * AlfaBeta * Const_C	1.695E-05	60	2.825E-07	1.209	.148
Error	1.009E-04	432	2.336E-07		
Total	6.555E-04	648			
Total Corregido	1.933E-04	647			
a. R Cuadrado = 0.481 (R Cuadrado Ajustado = 0 .223)					

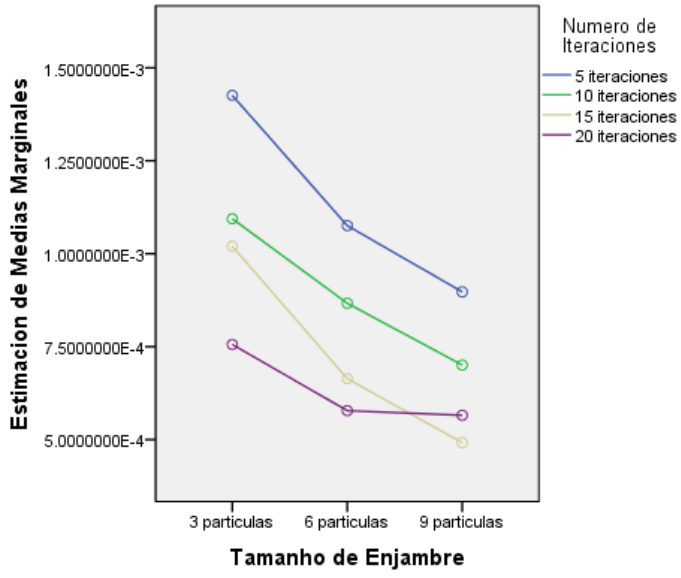
Fuente: Autor

Figura 36. Medias Marginales de la Variable de Respuesta (pmed33) – Tamanho de Enjambre vs Numero de Iteraciones



Fuente: Autor

Figura 37. Medias Marginales de la Variable de Respuesta (pmed33) – Numero de Iteraciones vs Tamanho de Enjambre



Fuente: Autor

De la tabla anterior se concluye que no existe interacción entre los factores descritos, ya que el valor de significancia para cada fuente de análisis es mayor que α. Sin embargo, existe evidencia para rechazar la hipótesis H_0 en el caso de los tres primeros efectos principales, y se puede afirmar que existe un efecto significativo sobre la variable de salida. Para el factor de las constantes C1/C2/C3 se acepta la hipótesis H_0 y se afirma que no existe efecto significativo. En la Figura 36 y 37 se muestra nuevamente el efecto principal que tienen los factores Número de Iteraciones y Tamaño de Enjambre, así mismo, con la muestra que se tomó se observa que en los últimos niveles puede existir una interacción de estos dos factores junto al factor $c_1 / c_2 / c_3$ como lo sugiere la tabla de ANOVA anterior.

4.2.3. Análisis de Varianza (ANOVA) para el problema pmed37

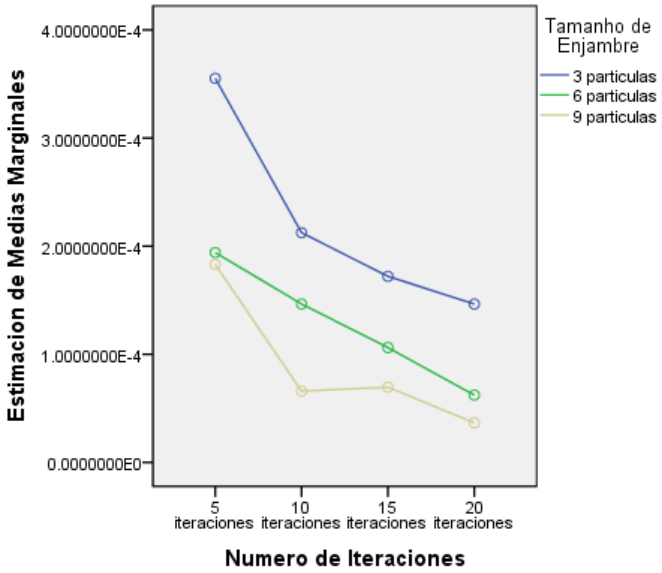
Tabla 14. Prueba de Efectos entre Factores para el problema pmed37

Prueba de Efectos entre Factores					
Variable Dependiente: Desviación del Optimo					
Fuente	Suma de Cuadrados tipo III	Grados de Libertad	Cuadrado Medio	F	Significancia
Modelo Corregido	9.153E-006[a]	215	4.257E-08	1.933	.000
Intersección	1.379E-05	1	1.379E-05	625.984	.000
Iteraciones	2.377E-06	3	7.924E-07	35.978	.000
Enjambre	2.015E-06	2	1.008E-06	45.751	.000
AlfaBeta	2.945E-08	2	1.472E-08	.668	.513
Const_C	1.547E-07	5	3.094E-08	1.405	.221
Iteraciones * Enjambre	2.169E-07	6	3.615E-08	1.641	.134
Iteraciones * AlfaBeta	1.723E-07	6	2.872E-08	1.304	.254
Iteraciones * Const_C	2.455E-07	15	1.637E-08	.743	.741
Enjambre * AlfaBeta	1.020E-07	4	2.550E-08	1.158	.329
Enjambre * Const_C	2.222E-07	10	2.222E-08	1.009	.435
AlfaBeta * Const_C	4.159E-07	10	4.159E-08	1.888	.045
Iteraciones * Enjambre * AlfaBeta	1.727E-07	12	1.439E-08	.653	.796
Iteraciones * Enjambre * Const_C	6.883E-07	30	2.294E-08	1.042	.409
Iteraciones * AlfaBeta * Const_C	6.916E-07	30	2.305E-08	1.047	.402
Enjambre * AlfaBeta * Const_C	5.041E-07	20	2.521E-08	1.144	.301
Iteraciones * Enjambre * AlfaBeta * Const_C	1.144E-06	60	1.907E-08	.866	.750
Error	9.515E-06	432	2.203E-08		
Total	3.246E-05	648			
Total Corregido	1.867E-05	647			

a. R Cuadrado = 0.481 (R Cuadrado Ajustaddo = 0 .223)

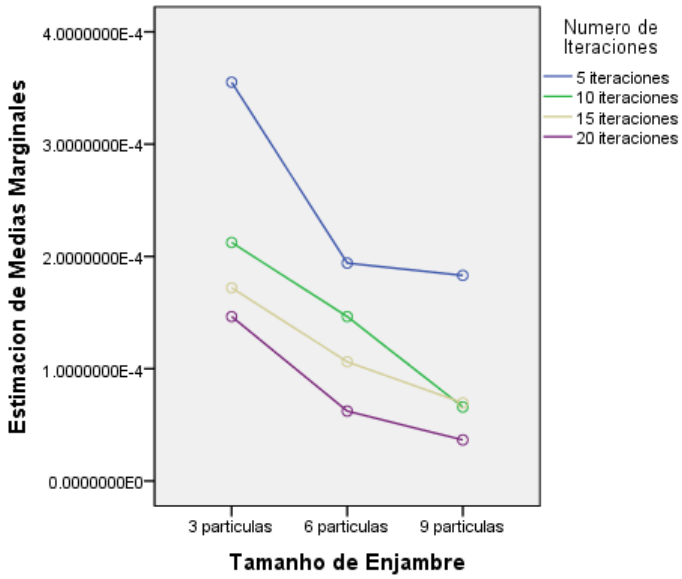
Fuente: Autor

Figura 38. Medias Marginales de la Variable de Respuesta (pmed33) – Tamanho de Enjambre vs Numero de Iteraciones



Fuente: Autor

Figura 39. Medias Marginales de la Variable de Respuesta (pmed33) – Numero de Iteraciones vs Tamanho de Enjambre



Fuente: Autor

De la tabla anterior se concluye que no existe interacción significativa entre los grupos de factores excepto por α / β - $c_1 / c_2 / c_3$ cuyo valor de 0.045 es menor α (0.05). No hay evidencia de efecto significativo de los factores α / β y $c_1 / c_2 / c_3$. Sin embargo, el valor de significancia para los factores Iteraciones y Enjambre es menor que α(0.05); por lo tanto se rechaza la hipótesis H_0 y se concluye que el efecto de estos dos factores es significativo. En las Figuras 38 y 39 se muestra el efecto de los factores Numero de Iteraciones y Tamaño de Enjambre, asi mismo se observa que no existe indicios de interacción entre estos dos factores.

4.2.4. Análisis de Varianza (ANOVA) para el problema pmed40

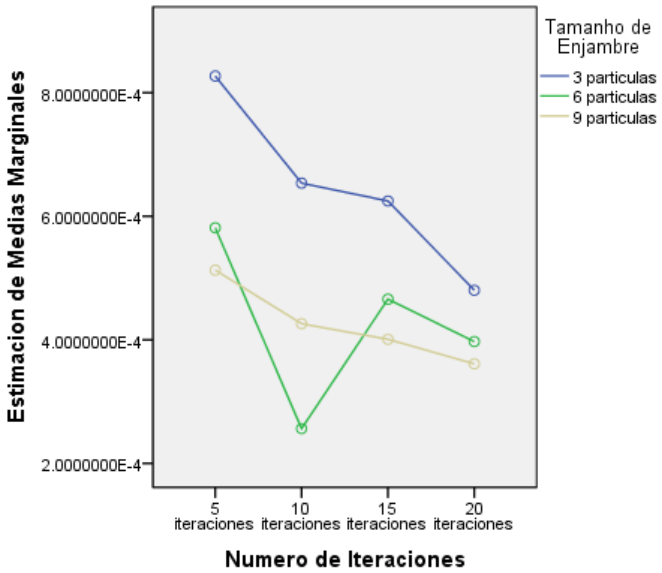
Tabla 15. Prueba de Efectos entre Factores para el problema pmed40

Prueba de Efectos entre Factores					
Variable Dependiente: Desviación del Optimo					
Fuente	Suma de Cuadrados tipo III	Grados de Libertad	Cuadrado Medio	F	Significancia
Modelo Corregido	8.796E-005[a]	215	4.091E-07	1.109	.186

Intersección	.000	1	.000	437.315	.000
Iteraciones	4.906E-06	3	1.635E-06	4.433	.004
Enjambre	7.045E-06	2	3.523E-06	9.549	.000
AlfaBeta	1.173E-06	2	5.863E-07	1.589	.205
Const_C	2.207E-06	5	4.414E-07	1.196	.310
Iteraciones * Enjambre	2.029E-06	6	3.381E-07	.916	.483
Iteraciones * AlfaBeta	1.728E-06	6	2.880E-07	.781	.585
Iteraciones * Const_C	4.706E-06	15	3.138E-07	.851	.621
Enjambre * AlfaBeta	1.346E-06	4	3.365E-07	.912	.457
Enjambre * Const_C	3.071E-06	10	3.071E-07	.832	.597
AlfaBeta * Const_C	5.240E-06	10	5.240E-07	1.420	.168
Iteraciones * Enjambre * AlfaBeta	3.394E-06	12	2.829E-07	.767	.685
Iteraciones * Enjambre * Const_C	1.222E-05	30	4.074E-07	1.104	.325
Iteraciones * AlfaBeta * Const_C	1.053E-05	30	3.510E-07	.951	.543
Enjambre * AlfaBeta * Const_C	6.346E-06	20	3.173E-07	.860	.639
Iteraciones * Enjambre * AlfaBeta * Const_C	2.202E-05	60	3.670E-07	.995	.491
Error	.000	432	3.689E-07		
Total	.000	648			
Total Corregido	.000	647			
a. R Cuadrado = 0.481 (R Cuadrado Ajustado = 0 .223)					

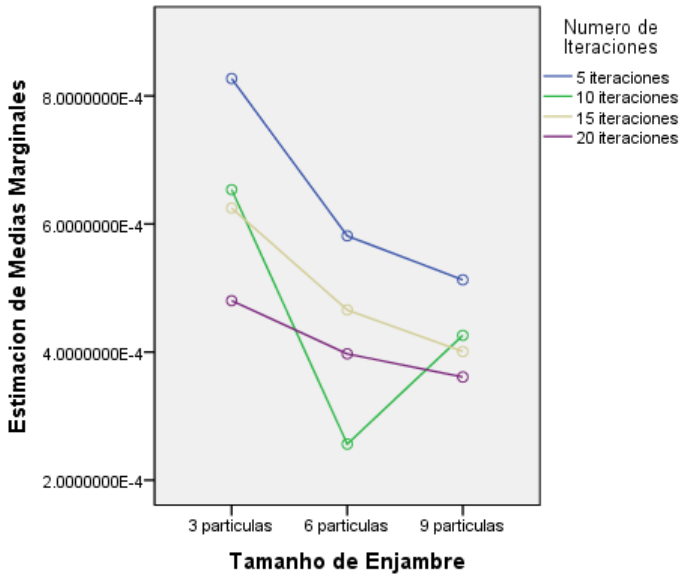
Fuente: Autor

Figura 40. Medias Marginales de la Variable de Respuesta (pmed33) – Tamanho de Enjambre vs Numero de Iteraciones



Fuente: Autor

Figura 41. Medias Marginales de la Variable de Respuesta (pmed33) – Numero de Iteraciones vs Tamanho de Enjambre



Fuente: Autor

De la tabla anterior se concluye que no existe interacción significativa entre los grupos de factores y tampoco efecto significativo por parte de los factores α / β y $c_1 / c_2 / c_3$ sobre la variable de salida. Nuevamente, se puede afirmar que los factores Iteraciones y Enjambre tienen efecto significativo sobre la variable de salida ya que su valor de significancia es menor que α(0.05). En las Figuras 40 y 41 se observa el mismo comportamiento independiente de los factores con efectos principales significativos: Numero de Iteraciones y Tamaño de Enjambre.

5. Conclusiones

- Se alcanzaron todos los objetivos propuestos en la investigación; el algoritmo propuesto arroja buenos resultados para problemas de hasta 900 vértices en tiempos computacionales aceptables.

- Los algoritmos metaheuristicos producen muy buenas soluciones ya que aprovechan características de los algoritmosheurísticos para guiar la búsqueda local y utiliza otras herramientas que les permiten una exploración más diversa del espacio de soluciones.

- Se considera dejar los códigos de programación del algoritmo a disposición del usuario, ya sea para estudiar posibles variaciones aplicado al problema de la P-mediana o para estudiar posibles codificaciones para otros problemas de optimización combinatoria.

- Teniendo en cuenta los resultados arrojados en el experimento se concluye que:
  - La desviación respecto al valor óptimo para un problema de n vértices aumenta si la proporción de medianas (p) y vértices (n) es alta.
  - La variación de los niveles del factor $c_1 / c_2 / c_3$ no tiene efecto significativo sobre la variable de respuesta en ninguno de los cuatro problemas estudiados.
  - La variación de los niveles de los factores α / β no tiene efecto significativo sobre la variable de respuesta en los problemas pmed33, pmed37 y pmed40. Sin embargo, para el problema pmed34 dio como resultado efecto significativo sobre la variable de respuesta por parte de este factor.
  - Se concluye que no existe una interacción significativa entre cualquier grupo de factores (formado por los cuatro factores principales: Iteraciones, Tamaño-Enjambre, α / β y $c_1 / c_2 / c_3$), para los cuatro problemas estudiados, excepto por pmed33 en el cual dio como resultado una interacción significativa entre los factores Iteraciones-Enjambre-$c_1 / c_2 / c_3$ y en el problema pmed37 en cual se sugiere interacción entre α / β y $c_1 / c_2 / c_3$.

- Los factores Iteraciones y Tamaño-Enjambre tiene un efecto significativo sobre la variable respuesta para los cuatro problemas estudiados: a medida que se aumenta el valor de estas dos variables se concluye que se puede obtener mejores cotas del problema en cuestión. Por lo tanto, se recomienda utilizar valores altos para estas dos variables cuando la proporción de medianas y vértices es alta.

6. Recomendaciones

- Divulgar los proyectos de grado de investigación ya que es una experiencia enriquecedora y de gran aporte a la formación personal.

- Desarrollar trabajos con problemas prácticos para estudiar el desempeño del algoritmo en aplicaciones reales.

- Desarrollar software en otros lenguajes que permitan el uso del algoritmo por parte de la industria regional sin la necesidad de incurrir en altos costos por licencias.

7. Trabajos Futuros

- Investigar otros métodos de codificación que intenten aprovechar la estructura de espacios genotípicos para aplicar metaheuristicas avanzadas y de esta manera obtener soluciones de calidad.

- Investigar posibles modificaciones al método de codificación que intenten mejorar la relación entre el espacio genotípico y fenotípico del problema y de esta manera aumentar el desempeño del algoritmo para los casos más difíciles de una instancia que es cuando la proporción de medianas y vértices es alta.

- Ampliar la investigación a otros problemas de localización como el Capacitated P-median Problem, Uncapacitated Facility Location Problem, entre otros

Referencia Bibliográfica

[1] AI, The Jin y KACHITVICHYANUKUL, Voratas . Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem.Computers & Industrial Engineering.vol. 56.no 1. p. 380–387

[2] CHEN, Ai-ling; YANG, Gen-ke y WU, Zhi-ming. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. Journal of Zhejiang University Science A. vol. 7.no. 4. p. 607-614

[3] CHIYOSHI, F. y GALVAO. D.. A statistical analysis of simulated annealing applied to the p-median problem. Annals of Operations Research. 2000.

[4] EL-SHAIEB, A.M..A new algorithm for locating sources among destinations. Managemnt Science. vol. 20, no. 2 p. 221-231. 1973.

[5] ERKUT, Erhan; ALP, Osman y DREZNER, Zvi. An effcieitn genetic algorithm for the p-median problem.Annals of Operations Research. 2003.

[6] FELDMAN, E.; LEHRER, F.A. y RAY, T.L..Warehouse locations under continuous economies of scale.Management Science, 1966.

[7] GALVAO, Roberto Dieguez y RAGGI, Luis Aurelio. A method for solving to optimality uncapacitated location problems. Annals of Operations Research.vol. 18. p. 225-244. 1989

[8] HAKIMI; S.L..Optimum locations of switching centers and the absolute centers and medians on a graph.Operations Research.vol. 12.no. 3. p. 450-459. 1964.

[9] HAKIMI; S.L.. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Operations Research.vol.13, no. 3. p. 462-475. 1965.

[10] HANSEN, P. y MLADENOVIC, N..Variable Neighborhood Search for the p-median. Location Science, 1997.

[11] HOOSAGE, C.M. y GOODCHILD, M.F..Discrete space location-allocation solutions from genetic algorithms.Annals of Operations Research. 1986.

[12] JARVINEN, Pertti; RAJALA, Jaakko y SInervo Heikki. A Branch-and-Bound Algorithm for seeking the P-median.Operations Research.vol.20, no. 1. p. 173-178.

[13] KENNEDY, James y EBERHART, Rusell C..Particle Swarm Optimization.Neural Networks, 1995.Proceedings., IEEE International Conference on.. vol. 4. p. 1942-1948.

[14] KENNEDY, James y EBERHART, Rusell C..A discrete binary version of the particle swarm algorithm.Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation., 1997 IEEE International Conference on. vol. 5. p. 4104-4108.

[15] KHANESAR, Mojtaba Ahmadieh; TESHNEHLAB, Mohammad y SHOOREHDELI, Mahdi Aliyari. A novel binary particle swarm optimization. Control & Automation, 2007.MED '07.Mediterranean Conference on. p. 1-6.

[16] KARIV, O. y HAKIMI, S.L..An Algorithm approach to network location problems. II. The P-medians. SIAM Journal on Applied Mathematics.vol. 37.no. 3. p. 539-560. 1979.

[17] KUEHN, A.A. y HAMBURGER, M.J..A heuristic program for locating warehouses.Management Science. 1963.

[18] MARANZANA, F.E..On the location of supply points to minimize transport costs.Operations Research Quarterly, 1964.

[19] MARTINEZ GARCIA, F. Javier y MORENO PEREZ, Jose A. Jumping Frog Optimization: a new swarm method for discrete optimization. Reporte Tecnico. DEIOC 3/2008

[20] MLADENOVIC, Nenad, *et al*. The p-median problem: A survey of metaheuristic approaches.

[21] MORAGLIO, Alberto *et al.* Geometric Particle Swarm Optimization. Journal of Artificial Evolution and Applications.vol. 2008.

[22] NARULA, Subhash C.; OGBU, Ugonnaya I. y Samuelsson, Haakon M..An algorithm for the P-median problem.Operations Research.vol. 25.no. 4. p. 709-713. 1976

[23] QIN, Jin; LI, Xin y YIN, Yixin. An algorithmic framework of discrete particle swarm optimization.Applied Soft Computing.vol. 12.no. 3. p. 1125–1130

[24] RESENDE, M.G. y WERNECK, R.F..A hybrid heuristic for the p-median problem.Journal of Heuristics, 2004.

[25] REVELLE, C. y SWAIN, R. Central Facilities Location. Geographical Analysis. 1970

[26] ROLLAND, E.; SCHILLING, D.A. y CURRENT, J.R..An efficient tabu search procedure for the p-median problem.European Journal of Operations Research, 1996.

[27] ROSING, K.E. y REVELLE, C.S.. Heuristic Concentration; Two stage solution construction.European Journal of Operational Research, 1997.

[28] SHI, Yu-hui y EBERHART, Rusell C.. A modified particle swarm optimizer. Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on. p. 69-63

[29] TEITZ, M.B. y BART, P. Heuristic methods for estimating the generalized vertex median of a weighted graph. Operation Research.vol. 16.no. 5. p. 955-961. 1968

[30] YANG, Shuyuan ; WANG, Min y JIAO, Li-Cheng Cheng. A quantum particle swarm optimization. Evolutionary Computation, 2004. CEC2004. Congress on.vol. 1. p. 320-324

Apéndice A. Demonstración de Teoremas de Hakimi

Teorema 1: La *mediana absoluta* de un grafo se encuentra siempre sobre un vértice.

Una forma de demostrar este teorema es mostrar que si x_0 es un punto arbitrario en el grafo G y $x_0 \neq v_i$ para $i = 1,2,...,n$, entonces existe siempre un vértice v_m en G tal que

$$\sum_{i=1}^{n} h_i d\ (v_i, x_0) \geq \sum_{i=1}^{n} h_i d\ (v_i, v_m) \quad (1)$$

Es decir, la suma de la distancias de todos los vértices hasta el vértice v_m siempre es menor o igual que la suma de las distancias de todos los vértices hasta un punto x_0 situado en una de las aristas de G.

Hipótesis: La localización óptima de una instalación es un punto x_0 entre los vértices v_p y v_q, es decir pertenece al arco $b(v_p, v_q)$, tal que

$$\sum_{i=1}^{n} h_i d\ (v_i, x_0) < \sum_{i=1}^{n} h_i d\ (v_i, v_p) \quad (2)$$

Para comenzar, se sabe que la distancia entre un vértice v_i y un punto x_0 es la menor trayectoria entre estos dos puntos.

$$d(v_i, x_0) = \min\left[d(x_0, v_p) + d(v_p, v_i), d(x_0, v_q) + d(v_q, v_i)\right] \quad (3)$$

Se considera $i_1, i_2, ..., i_n$ como una reorganización (permutación) del conjunto de enteros $1,2,....,n$ tal que

$$d(x_0, v_{i_k}) = d(x_0, v_p) + d(v_p, v_{i_k}), \quad \text{for } k = 1,2,...,r, \ (r \leq n)$$
$$d(x_0, v_{i_k}) = d(x_0, v_q) + d(v_q, v_{i_k}), \quad \text{for } k = r+1, r+2,...,n. \tag{4}$$

Es decir, se crean dos conjuntos de elementos: el primer conjunto contiene r vértices v_{i_k} que representan los vértices a los cuales se accede más fácilmente a través de v_p desde x_0. Los otros $n-r$ vértices corresponden a los vértices que se accede más fácilmente a través de v_q.

Así pues, se puede calcular la suma de las distancias entre cada vértice v_i y x_0 como

$$\sum_{i=1}^{n} h_i d(v_i, x_0) = \sum_{k=1}^{r} h_{i_k} \left[d(x_0, v_p) + d(v_p, v_{i_k}) \right] + \sum_{k=r+1}^{n} h_{i_k} \left[d(x_0, v_q) + d(v_q, v_{i_k}) \right] \tag{5}$$

Se asume que la mayoría de los vértices v_i (usuarios) tienen acceso más fácil a través del vértice v_p.

$$\sum_{k=1}^{r} h_{i_k} \geq \sum_{k=r+1}^{n} h_{i_k} \tag{6}$$

Partiendo de los conceptos presentados, se sabe que $d(x_0, v_q) = d(v_p, v_q) - d(x_0, v_p)$, luego la ecuación (5) puede ser modificada de la siguiente manera

$$\sum_{i=1}^{n} h_i d(v_i, x_0) = \sum_{k=1}^{r} h_{i_k} \left[d(x_0, v_p) + d(v_p, v_{i_k}) \right] + \sum_{k=r+1}^{n} h_{i_k} \left[d(v_p, v_q) + d(v_q, v_{i_k}) - d(x_0, v_p) \right]$$

(7)

También se sabe qué $d(v_p, v_q) + d(v_q, v_{i_k}) \geq d(v_p, v_{i_k})$. Si sustituimos la desigualdad anterior en la ecuación (7) de la suma total de distancias, obtenemos una desigualdad de la siguiente forma.

$$\sum_{i=1}^{n} h_i d(v_i, x_0) \geq \sum_{k=1}^{r} h_{i_k} \left[d(x_0, v_p) + d(v_p, v_{i_k}) \right] + \sum_{k=r+1}^{n} h_{i_k} \left[d(v_p, v_{i_k}) - d(x_0, v_p) \right] \quad (8)$$

Esto se debe a que la cantidad sustituida es siempre mayor o igual que la cantidad entrante, entonces el aporte de esta nueva cantidad en la suma de la parte de la derecha de la ecuación (8) es menor que con la cantidad anterior, así el valor de la parte derecha solo puede ser a lo más el valor de la parte izquierda.

La parte derecha de la ecuación (8) se puede organizar de la siguiente manera.

$$\sum_{i=1}^{n} h_i d(v_i, x_0) \geq \sum_{i=1}^{n} h_i d(v_p, v_i) + \left[\sum_{k=1}^{r} h_{i_k} - \sum_{k=r+1}^{n} h_{i_k} \right] h_i d(x_0, v_p) \quad (9)$$

Se asumió inicialmente que $\sum_{k=1}^{r} h_{i_k} \geq \sum_{k=r+1}^{n} h_{i_k}$, esto implica que el producto

$$\left[\sum_{k=1}^{r} h_{i_k} - \sum_{k=r+1}^{n} h_{i_k} \right] h_i d(x_0, v_p) \quad (10)$$

es mayor o igual que cero. De esta manera se concluye que

$$\sum_{i=1}^{n} h_i d(v_i, x_0) \geq \sum_{i=1}^{n} h_i d(v_i, v_p) \quad (11)$$

lo cual contradice la hipótesis presentada y prueba la existencia de un vértice que minimiza el costo total. De la misma manera, se puede asumir que la mayoría de usuarios tiene acceso más fácil a través de v_q y con los mismos pasos demostrar que existe un vértice que minimiza las distancias.

El segundo teorema es una generalización del teorema que acaba ser presentado. Y su demonstración es similar.

Teorema 2: Existe por lo menos un subconjunto de vértices $V_p^* \subseteq V$, donde $V = \{v_1, v_2, ..., v_n\}$, tal que

$$\sum_{i=1}^{n} h_i d(v_i, X_p) \geq \sum_{i=1}^{n} h_i d(v_i, V_p^*) \quad (12)$$

para cualquier conjunto arbitrario de puntos X_p localizado en los arcos o los vértices de G.

Hipótesis: La localización óptima de un conjunto de instalaciones es un conjunto de p puntos $X_p = \{x_1, x_2, ..., x_p\}$ sobre los arcos de G tal que

$$\sum_{i=1}^{n} h_i d(v_i, X_p) < \sum_{i=1}^{n} h_i d(v_i, V_p^*) \quad (13)$$

donde $V_p^* \subseteq V$ y $V = \{v_1, v_2, ..., v_n\}$.

Primero se define la distancia entre un vértice v_i y un conjunto de puntos X_p como la distancia mínima entre los puntos v_i y un punto $x \in X$. Esto es

$$d(v_i, X_p) = \min_{x \in X_p} d(x, v_i) \quad (14)$$

Se considera $i_1, i_2, ..., i_n$ como una permutación de los enteros $1, 2, ..., n$ tal que

$$
\begin{aligned}
&d(v_{i_k}, X) = d(v_{i_k}, x_1) && (k = 1,2,...,k_1)\\
&d(v_{i_k}, X) = d(v_{i_k}, x_2) && (k = k+1,...,k_2)\\
&\quad ... \quad ...\\
&d(v_{i_k}, X) = d(v_{i_k}, x_p) && (k = k_{p-1}+1,...,k_p = n)
\end{aligned} \tag{15}
$$

Es decir, se asignan los vértices v_i al punto x más cercano y se crean p subconjuntos en este proceso de asignación.

La suma total de distancias de los vértices v_i al conjunto de puntos X_p se puede expresar de la siguiente manera.

$$
\sum_{i=1}^{n} h_i d(v_i, X) = \sum_{k=1}^{k_1} h_{i_k} d(v_{i_k}, x_1) + \sum_{k=k+1}^{k_2} h_{i_k} d(v_{i_k}, x_2) + ... + \sum_{k=k_{p-1}+1}^{n} h_{i_k} d(v_{i_k}, x_p) \tag{16}
$$

Ahora, se toma el primer término de la ecuación anterior (el subconjunto de vértices v_{i_k} que tiene acceso más fácil a X_p a través de x_1).

$$
\sum_{k=1}^{k_1} h_{i_k} d(v_{i_k}, x_1) = \sum_{k=1}^{k_1} h_{i_k} d(v_{i_k}, x_1) + \sum_{k=k_1+1}^{n} 0 d(v_{i_k}, x_1) \tag{17}
$$

Y se define la ecuación del primer término de la siguiente manera.

$$
\sum_{k=1}^{k_1} h_{i_k} d(v_{i_k}, x_1) = \sum_{k=1}^{n} h'_{i_k} d(v_{i_k}, x_1) \tag{18}
$$

donde

$$
h'_{i_k} = \begin{cases} h_{i_k} & (k = 1,2,...,k_1) \\ 0. & (k = k+1,...,n) \end{cases}
$$

En el *teorema 1* se demostró que existe un vértice v_1^* en G tal que

$$\sum_{k=1}^{n} h'_{i_k}\, d(v_{i_k}, x_1) \geq \sum_{k=1}^{n} h'_{i_k}\, d(v_{i_k}, v_1^*) \quad (19)$$

Se combinan las ecuaciones (18) y (19), y se llega al siguiente resultado

$$\sum_{k=1}^{k_1} h_{i_k} d(v_{i_k}, x_1) \geq \sum_{k=1}^{k_1} h_{i_k} d(v_{i_k}, v_1^*) \quad (20)$$

Es decir que la suma de las distancias entre los vértices v_{i_k} , asignados al punto x_1 ,y el conjunto de puntos X_p se minimiza si $x_1 = v_1^*$.

Este mismo razonamiento se puede utilizar para obtener las desigualdades restantes.

$$\sum_{k=k_1+1}^{k_2} h_{i_k} d(v_{i_k}, x_2) \geq \sum_{k=k_1+1}^{k_2} h_{i_k} d(v_{i_k}, v_2^*)$$

$$\ldots \ \ldots$$

$$\sum_{k=k_{p-1}+1}^{n} h_{i_k} d(v_{i_k}, x_p) \geq \sum_{k=k_{p-1}+1}^{n} h_{i_k} d(v_{i_k}, v_p^*) \quad (21)$$

Se suman las partes correspondientes de las desigualdades (21) y se obtiene

$$\sum_{i=1}^{n} h_i d(v_i, X) \geq \sum_{k=1}^{k_1} h_{i_k} d(v_{i_k}, v_1^*) + \sum_{k=k+1}^{k_2} h_{i_k} d(v_{i_k}, v_2^*) + \ldots + \sum_{k=k_{p-1}+1}^{n} h_{i_k} d(v_{i_k}, v_p^*) \quad (22)$$

Se considera la representación del conjunto de vértices $v_1^*, v_2^*, \ldots, v_p^*$ como V_p^* y se modifica la ecuación anterior de la siguiente forma.

$$\sum_{i=1}^{n} h_i d(v_i, X) \geq \sum_{k=1}^{n} h_{i_k} d(v_{i_k}, V_p^*) \quad (23)$$

Esto prueba que la hipótesis es falsa y la localización óptima de un conjunto de medianas se encuentra en los vértices del grafo.

La definición de estos dos teoremas ha sido fundamental en la creación de algoritmos para resolver el PMP, ya que se puede restringir el proceso de búsqueda a los vértices del grafo.

Apéndice B. Códigos de Programación

El presente anexo contiene las funciones utilizadas para la construcción del algoritmo que resuelve el PMP, a través de QPSO y el método de codificación propuesto.

Función Matriz_Adyacencia

Construye la matriz de adyacencia de alguno de los 40 problemas de OR-Library para la P-mediana y utiliza la función FastFloyd para hallar la matriz de distancias.

Entrada:

- n: Numero de problema que se quiere resolver de la librería OR-library

Salida:

- matriz_Distancias: Matriz de Distancias
- nClientes: número de clientes del problema.
- numero_Facilidades: número de facilidades del problema

```
function [ matriz_Distancias,nClientes, numero_Facilidades] =
Matriz_Adyacencia(n)
%Extrae los problemas OR-Library de una hoja de excel y utiliza el
%algoritmo FloydWarshall para hallar la matriz de distancias
% La funcion toma como argumento el numero del problema del OR-Library
que
% se quiere resolver, y el resultado es la matriz de distancias y el
%numero de clientes de ese problema.

%recuperar datos de la hoja de calculo de excel
[matriz_Datos,~,~] = xlsread('ORlibrary.xlsx',n);

%crear variable numero de clientes
nClientes = matriz_Datos(1,1);

%crear variable numero de facilidades
numero_Facilidades = matriz_Datos(1,3);

%crear variable numero de aristas del grafo fe cada problema
[number_of_Edges,~] = size(matriz_Datos);
```

```
%crear matriz de adyacencia con las aristas del grafo del problema
matriz_Adyacencia = zeros(nClientes);
for z = 2:number_of_Edges
matriz_Adyacencia(matriz_Datos(z,1),matriz_Datos(z,2))=
matriz_Datos(z,3);
matriz_Adyacencia(matriz_Datos(z,2),matriz_Datos(z,1))=
matriz_Datos(z,3);
end

%asignar una magnitud (distancia) infinita a las aristas que no existen
celdas_Cambio = (matriz_Adyacencia == 0);
matriz_Adyacencia(celdas_Cambio) = Inf;

%asignar cero a la diagonal o distancia entre los elementos mismos
for i = 1:nClientes
   matriz_Adyacencia(i,i)=0;
end

%crear matriz de distancias con una implementacion rapida de Floyd-
%Algorithm
matriz_Distancias = FastFloyd(matriz_Adyacencia);

end
```

Función FastFloyd

Encuentra las distancias mínimas entre todos los pares de nodos del grafo utilizando la matriz de adyacencia, y construye la matriz de distancias.

Fuente: ARENDT, Dustin. Vectorized Floyd-Warchall.Vectorized (fast) implementation of the Floyd-Warshall all pairs shorest path algorithm [On line]. . [Citado el 29 de Marzo de 2013; 17:00:00]. Disponible en: http://www.mathworks.nl/matlabcentral/fileexchange/25776-vectorized-floyd-warshall/content/FastFloyd.m

Entrada:

- D: Matriz de Adyacencia

Salida:

- D: Matriz de Distancias

```
% FastFloyd - quickly compute the all pairs shortest path matrix
%
% Uses a vectorized version of the Flyod-Warshall algorithm
% see: http://en.wikipedia.org/wiki/Floyd_Warshall
%
% USAGE:
%
% D = FastFloyd(A)
%
% D - the distance (geodesics, all pairs shortest path, etc.) matrix.
% A - the adjacency matrix, where A(i,j) is the cost for moving from
vertex i to
%     vertex j.  If vertex i and vertex j are not connected then A(i,j)
should
%     be >= the diameter of the network (Inf works fine).
%
% EXAMPLE:
%
% Here I create a random binary matrix and convert it to an integer
format. Then
% I take the reciprocal of the matrix so that all non-adjacent pairs get
a value
% of Inf.  The result is stored in D.
%
% A = int32(rand(100,100) < 0.05);
% D = FastFloyd(1./A)
%

function D = FastFloyd(D)

    n = size(D, 1);

for k=1:n

        i2k = repmat(D(:,k), 1, n);
        k2j = repmat(D(k,:), n, 1);

D = min(D, i2k+k2j);

end

end
```

Función Mejor_Personal_Enjambre

Compara la posición actual de cada partícula con la mejor posición encontrada anteriormente.

Entrada:

- fmejor_Personal: desempeño de la mejor posición encontrada por cada partícula del enjambre
- mejor_Personal: mejor posición encontrada por cada partícula del enjambre
- fposicion_Enjambre: desempeño de la posición actual de las partículas del enjambre
- posicion_Enjambre: posición actual de las partículas del enjambre

Salida:

- fmejor_Personal: desempeño de la mejor posición encontrada por cada partícula del enjambre
- mejor_Personal: mejor posición encontrada por cada partícula del enjambre

```
function [ fmejor_Personal,mejor_Personal ] = Mejor_Personal_Enjambre(...
    fmejor_Personal, mejor_Personal,fposicion_Enjambre,posicion_Enjambre
)

%comparar el fitnes de mejor_Personal y la posicion dle Enjmbre
A = fposicion_Enjambre < fmejor_Personal;

if any( A == 1)

%Actualizar el vector fitnes de mejor_Personal
    fmejor_Personal( A ) = fposicion_Enjambre( A );

%Actualizar el vector de mejor_Peronal
    mejor_Personal( A,: ) = posicion_Enjambre( A,: );

end

end
```

Función Mejor_Global_Enjambre

Compara la mejor posición de cada partícula con la mejor posición encontrada por todo el enjambre.

Entrada:

- fposicion_Enjambre: desempeño de la posición actual de las partículas del enjambre
- posicion_Enjambre: posición actual de las partículas del enjambre
- fmejor_Global:desempeño de la mejor posición de todo el enjambre
- mejor_Global:mejor posición encontrada por todo el enjambre
- binario_Facilidades_Enjambre: conjunto de facilidades correspondiente a cada partícula
- binario_mejor_Global:conjunto de facilidades correspondiente a la mejor posición encontrada por el enjambre.

Salida:

- fmejor_Global:desempeño de la mejor posición de todo el enjambre
- mejor_Global:mejor posición encontrada por todo el enjambre
- binario_mejor_Global:conjunto de facilidades correspondiente a la mejor posición encontrada por el enjambre.

```
function [ mejor_Global,fmejor_Global,binario_mejor_Global ] =...
    Mejor_Global_Enjambre( fposicion_Enjambre,posicion_Enjambre,...
    fmejor_Global,mejor_Global,binario_Facilidades_Enjambre,...
    binario_mejor_Global )

%comparar el fitnes de mejor_Personal y la posicion dle Enjmbre
A = fposicion_Enjambre < fmejor_Global;

if any( A == 1)
```

```
if length(find(A)) > 1
posicion = find(A);
        posicion = posicion( ceil( rand * length( posicion ) ) );
B = zeros(size(A));B(posicion) = 1;
A = logical(B);
end

%Actualizar el vector fitnes de mejor_Personal
    fmejor_Global = fposicion_Enjambre( A );

%Actualizar el vector de mejor_Peronal
    mejor_Global = posicion_Enjambre( A,: );

%Actualizar el binario mejor global
    binario_mejor_Global = binario_Facilidades_Enjambre( A, : );

end

end
```

Función Decodificacion_Enjambre

Decodifica las posiciones del espacio genotípico a soluciones factibles en el espacio fenotípico.

Entrada:

- matriz_Distancias: matriz de distancias
- posicion_Enjambre: posición actual de las partículas del enjambre
- numero_Facilidades: numero de facilidades (p)
- nClientes: numero de clientes o vertices
- tamanho_Enjambre: tamanho del enjambre

Salida:

- fposicion_Enjambre: desempeño de la posición actual de las partículas del enjambre
- binario_Facilidades_Enjambre: conjunto de facilidades correspondiente a cada partícula

```
function [fposicion_Enjambre,binario_Facilidades_Enjambre] =...
    Decodificacion_Enjambre( matriz_Distancias,posicion_Enjambre,...
numero_Facilidades, nClientes, tamanho_Enjambre )


%inicializar vector exito
matriz_Exito = zeros( tamanho_Enjambre, nClientes );

%inicializar fEnjambre y solucion valida del PMP
fposicion_Enjambre = zeros( tamanho_Enjambre, 1 );

%inicializar binario_facilidades del enjambre completo
[ binario_Facilidades_Enjambre ] = zeros( tamanho_Enjambre, nClientes );

%para todas las particulas del enjambre
for i = 1:tamanho_Enjambre

%dada una particula, calcular el in-degree de cada vertice o cliente
    vector_Ante_Exito =  transpose( reshape( posicion_Enjambre(i,:), ...
nClientes, nClientes ) );
    matriz_Exito(i,:) = sum( vector_Ante_Exito );

%ordenar vector de valores enteros de la partiucula i
[ vector_Exito_Sorted,indexes ] = sort( matriz_Exito(i,:),'descend' );

%hallar el indice de referencia de las p medianas y evaluar si existen
%mas celdas con ese mismo valor
    index_Ref = vector_Exito_Sorted( numero_Facilidades );
indexes_sameasindex_Ref = find( vector_Exito_Sorted == index_Ref );

%recuperar los indices que empataron con idex_Ref
indexes_to_Change = indexes( indexes_sameasindex_Ref );

%establecer los vertices (indices) que son medianas definitivas
indexes_Stable = indexes( 1: indexes_sameasindex_Ref(1) - 1 );

%evaluar si es necesario romper empate con el metodo constructivo
if ( length( indexes_to_Change ) == ( numero_Facilidades - ...
( indexes_sameasindex_Ref(1) - 1 ) ) )

%NO REPARACION.construir el conjunto de mdianas defintivos
```

```
binario_Facilidades = zeros( 1 , nClientes );
            facilities_definitive = [ indexes_Stable indexes_to_Change ];
binario_Facilidades( facilities_definitive ) = 1;

else
%SI REPARACION.aplicar metodo greedy y construir conjunto
%definitvo de medianas
            [binario_Facilidades ] = Reparacion(...
matriz_Distancias,indexes_to_Change,indexes_Stable,...
numero_Facilidades, nClientes);

end

%aplicar metodo de mejoramiento a la solucion construida.
[evaluacion_Particula, binario_Facilidades] = Vertex_Substitution(...
       matriz_Distancias,binario_Facilidades);

%esta matriz de soluciones factibles se exporta a mejor personal
   binario_Facilidades_Enjambre(i,:) = binario_Facilidades;

%devolver la evaluacion de la solucion decodificada
   fposicion_Enjambre(i) = evaluacion_Particula;

end

end
```

Función Reparacion

Utiliza algoritmo Greedy para resolver subproblema en la decodificación y romper empate entre vértices en el proceso de "votación".

Entrada:

- matriz_Distancias: Matriz de Distancias
- numero_Facilidades: número de facilidades del problema original (p)
- nClientes: número de clientes o vértices del problema
- indexes_to_Change: índices que empataron en el proceso de votación; constituyen las facilidades que se van a ubicar en el subproblema.
- indexes_Stable: facilidades pre-instaladas del subproblema

Salida:

- binario_Facilidades: conjunto de medianas correspondientes a una partícula después del proceso de reparación.

```
function [binario_Facilidades ] = Reparacion( matriz_Distancias,...
indexes_to_Change,indexes_Stable,numero_Facilidades, nClientes)

%Inicializar indexes_Cambiar, y vertice_Entra
indexes_Cambiar = indexes_to_Change;
vertice_Entra = 0;

%obtener la facilidades que son definitiva en las posiciones
%respectivas de un vector bianrio
    binario_Facilidades = zeros( 1, nClientes );
    binario_Facilidades( indexes_Stable ) = 1;


while ( sum( binario_Facilidades ) < numero_Facilidades )
%actualizar indexes_Cambiar despues de anhadir vertice_Entra al
%conjunto de medianas definitivo
        indexes_Cambiar = indexes_Cambiar( indexes_Cambiar ~=
vertice_Entra );

%obtener los indices (numeros enteros) de las facilidades
%defintivas actuales
        facilidades_Actuales = find( binario_Facilidades );

%crear vector de prueba del tamanho de p, con ceros en las celdas
%que no se han definido
        facilidades_Prueba = [ facilidades_Actuales  zeros(1,...
            numero_Facilidades - sum(binario_Facilidades))];

%Obtener el numero de facilidades definitivas
        numero_facilidades_Actuales = length( facilidades_Actuales );

%Obtener el numero de pruebas que corresponder a los
%indexes_Cambiar restantes
colu = length( indexes_Cambiar );

%Crear vector que contiene las evaluacones de las pruebas con los
%vertices restantes de indexes_Cambiar
vector_Eval = zeros( size( indexes_Cambiar ) );

for i=1:colu
```

```
%Adicionar elemento de prueba a las facilidades definitivas
%para hacer la prueba
            facilidades_Prueba( numero_facilidades_Actuales + 1 ) = ...
                indexes_Cambiar(i);

%Obtener submatriz de orden (numeroclientes x distancias de
%fac.definitivas + elemento.Prueba)
            sub_matriz_Dist = matriz_Distancias( :, facilidades_Prueba(
...
                facilidades_Prueba > 0) );

%Obtener el minimo de cada fila y sumar los elementos, el
%resultado es la evaluacion del elemento prueba i
evaluacion = sum(min(sub_matriz_Dist,[],2));

%Asignar la evaluacion al vector de las evaluaciones de todas
%las pruebas
            vector_Eval(i) = evaluacion;
end

%Obtener la minima evaluacion de todas la pruebas
Position = find( vector_Eval == min( vector_Eval ));

%Si existen empates romper de forma aleatoria
if length(Position)>1
bar=ceil(rand*(length(Position)));               Position=Position(bar
end

%Obtnter el elemento prueba que entra  y asignarlos al vector
%binario de facilidades definitivas
        vertice_Entra = indexes_Cambiar( Position );
        binario_Facilidades( vertice_Entra ) = 1;

end

end
```

Función Vertex_Substitution

Toma la solución que produce el proceso de decodificación y aplica VertexSubstitution para encontrar una mejor solución en el vecindario.

Entrada:

- matriz_Distancias: Matriz de Distancias

- binario_Facilidades: conjunto de medianas correspondientes a una partícula después del proceso de decodificacion y/o reparacion

Salida:

- binario_Facilidades: conjunto de medianas correspondientes a una partícula después del proceso de mejoramiento.
- Z: desempeño de la solución correspondiente al conjunto de facilidades que se produce con el proceso de mejoramiento.

```
function [Z, binario_Facilidades] = Vertex_Substitution(...
matriz_Distancias,binario_Facilidades)

p = sum(binario_Facilidades);

[~,nClientes] = size(matriz_Distancias);

c1 = zeros(nClientes,1); c2 = zeros(nClientes,1);

solo_Clientes = find(~ismember(1:nClientes,find(binario_Facilidades)));

solucion = [ find(binario_Facilidades)  solo_Clientes ] ;

submatriz_Distancias = matriz_Distancias(:,solucion(1:p));

fbest = 0;
%% Inicializar Vectores c1 y c2 para implementar Move %%
%c1 = vector de la minima facilidad para cada clientes
%c2 = vector con la segunda min. facilidad para cada cliente

for i = 1:nClientes
    distMin = 9.9e+12;
for j = 1:p

%evaluar si elemento es menor que el min. actual de la fila
if submatriz_Distancias(i,j) < distMin
            distMin = submatriz_Distancias(i,j); jselec = j;
end

end

%calcular el aporte de la fila i a la solcuion actual
fbest = fbest + distMin;
```

```
%asignar la minima facilidad del cliente i a c1(i)
c1(i) = solucion( jselec );
end

for i = 1:nClientes
distMin = 9.9e+12;
for j = 1:p
jj = solucion(j);

%evaluar si el elemento jj es el minimo actual (if yes, leap)
if c1(i) == jj
continue
end
%evaluar si elemento actual jj es el segundo menor
if matriz_Distancias(i,jj) < distMin
distMin = matriz_Distancias(i,jj); jselec = jj;
end
end
c2(i) = jselec;
end
%% %%%%%% Proceso de Intercambio (Mejora propuesta por Whitaker) %%%%%%
%%
sentinela = 0;
while sentinela == 0   % The exchange process begins here
%inicializar mejor intercambio encontrado
wstar = 0;
for goin = p + 1 : nClientes
ii = solucion(goin); w = 0; u = zeros(nClientes,1);
        %% Calcular mejor candidato en P para remover %%
for i = 1:nClientes

%evaluar si elemento i de entra es < que el < actual de P.
if matriz_Distancias(i,ii) < matriz_Distancias(i,c1(i))

%calcular decremento del intercambio en la fila i
w = w + matriz_Distancias(i,ii) - matriz_Distancias(i,c1(i));
else
%actualizar el delta del objetivo si se elimina c1(i)
u(c1(i)) = u(c1(i)) + min( matriz_Distancias(i,ii),...
                    matriz_Distancias(i,c2(i))) -
matriz_Distancias(i,c1(i) );
end

end

%encontrar el menor elemento de u y asiganr a la variable goout
%el valor del indice que contiene el menor elemento de u
distMin = 9.9e+12;
for j = 1:p
jj = solucion(j);
if u(jj) < distMin
distMin = u(jj);
goout = j;
```

```
end
end


%Actualizar la funcion objetivo del mejor posible intercambio
        w = w + distMin;

%evaluar si el intercambio actual es mejor que el mejor
%intercambio encontrado
if w < wstar
%guardar indice de mejor in y out encontrado.
wstar = w; goinmejor = goin; gooutmejor = goout;
end
end

%evaluar si el mejor intercambio resulta en mejora (decremento)
%True. activar sentinela y terminar algoritmo
%False. actualizar c1 y c2. comenzar nuevo ciclo(Best Improvement)
if wstar >= -.00001
sentinela = 1;
continue
end

    %% Actualizar los vectores c1 y c2 (si el mejor intercambio es un
decremento) %%

%Actualizar funcion objetivo de mejor solucion encontrada
fbest = fbest + wstar;

%Ejecutar el mejor intercambio encontrado (tener en cuenta que
%solucion es nx1, y las primeras p son la solucion real)
goinc = solucion(goinmejor); gooutc = solucion(gooutmejor);
idum = solucion(goinmejor); solucion(goinmejor) = solucion(gooutmejor);
solucion(gooutmejor) = idum;


%gooutc = la instalacion que salio del conjunto de medianas
%goinc = el cliente que entra a formar parte de las instalaciones

%para cada cliente
for i = 1:nClientes

%evaluar si la asignacion minima del cliente i fue removida
if c1(i) == gooutc

%evaluar si la asigancion entrante es < que la 2da menor asignacion
actual
%Verdad. asignar entrante como nueva menor asignacion del cliente i
if matriz_Distancias(i,goinc) <= matriz_Distancias(i,c2(i))
                c1(i) = goinc;
%asignar la 2da menor asign. actual como la menor asign. del ciente i
else
```

```
                c1(i) = c2(i);

%Encontrar segunda mejor asignacion.(caso en que la anterior es la nueva
mejor asign del cliente i)
dmin = 9.9e+12;
for j = 1:p
                    jj = solucion(j);
%Evaluar si facilidad jj es la menor asign. del cliente i
%Verdad. saltar a la siguiente columna (facilidad)
if c1(i) == jj
continue
end
%Evaluar si la facilidad jj es < que la 2da menor asignacion
%Verdad. asignar la facilidad jj  como la 2da menor asigancion
Incumbente%
if matriz_Distancias(i,jj) < dmin
                        dmin = matriz_Distancias(i,jj); jselec = jj;
end
end
%asignar la ultima 2da menor asign. Incumbente como la segunda menor
asignacion del cliente i
c2(i) = jselec;
end
%la asignacion del cliente i no fue removida
else
%evaluar si la menor asign. actual (not from goout) es mayor que la
posible nueva asignacion
%Verdad. asignar como nueva minima asignacion del cliente i
if matriz_Distancias(i,c1(i)) > matriz_Distancias(i,goinc)
c2(i) = c1(i); c1(i) = goinc;
%evaluar si la segunda menor asign. actual es mayor que la posible nueva
asignacion
%Verdad. asignar goin como nueva segunda menor asignacion del cliente i
elseif matriz_Distancias(i,goinc) < matriz_Distancias(i,c2(i))
c2(i) = goinc;
%evaluar si la segunda mejor asignacion pertenece a goooout.
%Verdad. calcular nuevo c2(i) ya que el actual es removido
elseif c2(i) == gooutc
dmin = 9.9e+12;
for j = 1:p
jj = solucion(j);
%Evaluar si la facilidad jj es la menor asign. del cliente i
%Verdad. saltar y evaluar otra columna
if c1(i) == jj
continue
end
%Evaluar si la facilidad jj es < que la segunda menor asignacion del
cliente i
%Verdad. asignar la facilidad jj  como la 2da menor asigancion
Incumbente%
if matriz_Distancias(i,jj) < dmin
                        dmin = matriz_Distancias(i,jj); jselec = jj;
end
end
```

```
%asignar la ultima 2da menor asign. Incumbente como la segunda menor
asignacion del cliente i
c2(i) = jselec;
end
end
end
end
Z = fbest;
medianas = solucion(1:p);
binario_Facilidades = zeros(1,nClientes);
binario_Facilidades(medianas) = 1;

end
```

Función Asignacion_Metodo_Matriz

Asigna los clientes a las facilidades más cercanas de una solución factible del PMP.

Entrada:

- indexFacilities: vector que contiene los vértices que conforman una solución factible del PMP.
- matriz_Distancias: Matriz de Distancias
- nClientes: número de clientes del problema.

Salida:

- fsolucionukn: desempeño de la solución solucionukn
- solucionukn: matriz dispersa con la asignación de los clientes a las facilidades de indexFacilities

```
function [ fsolucionukn,solucionukn ]= Asignacion_Metodo_Matriz( ...
indexFacilities, matriz_Distancias, nClientes )
%Esta funcion resuelve el problema de asignacion dada una solucion
factible
%del PMP


%inicializar Matriz de asignacion
matriz_Asignacion = zeros( size( matriz_Distancias ) );
```

```
%inicializar submatriz de asignacion
submatriz_Asignacion = zeros ( nClientes,length( indexFacilities ) );

%separar columnas de la matriz_Distancias correspondiente a las
facilidades
%de la solucion.
submatriz_Distancias = matriz_Distancias(:,indexFacilities);

%encontrar al minimo de cada fila
eval1 = min(submatriz_Distancias,[],2);

%para cada cliente
for i = 1:nClientes

%encontrar la facilidad mas cercana del cliente i
row = submatriz_Distancias(i,:);
position = find(row == eval1(i,:));

%si existen empates romer de manera aleatoria
if length( position ) > 1
bar = ceil( rand * length( position ) );
position = position( bar );
end
    submatriz_Asignacion(i,position) = 1;
end

%actualizar matriz de asignacion con los indices de las facilidades mas
%cercanas
matriz_Asignacion(:,indexFacilities) = submatriz_Asignacion;

%convertir asinacion en una matriz dispersa y evaluar la solucion
solucionukn = sparse(matriz_Asignacion);
fsolucionukn = sum(eval1);

end
```

Función QPSO_PMP

Resuelve el problema de la P-mediana por medio de la metaheuristica Quantum Particle Swarm Optimization.

Entrada:

- tamanho_Enjambre: tamaño del enjambre de partículas
- nproblema: número de problema de la librería OR-Library

- numero_iteraciones: número de iteraciones

Salida

- solucionOptimadelPMP: matriz de asignación de la solución optima encontrada
- fsolucionOptimadelPMP: desempeño de la solución optima encontrada

```
function [solucionOptimadelPMP,fsolucionOptimadelPMP,Toc3] = QPSO_PMP(...
tamanho_Enjambre,nproblema, numero_iteraciones)

tic

parametro = 0.5;

%Obtener la Matriz de Distancias o Dissimilariry Matrix y el valor del
%numero de clientes
[ matriz_Distancias,nClientes,numero_Facilidades ] =
Matriz_Adyacencia(...
nproblema );

%Inicializar la posicion de las particulas aleatoriamente.
posicion_Enjambre = rand( tamanho_Enjambre, nClientes*nClientes );
posicion_Enjambre = ( posicion_Enjambre > parametro );

%Decodificar enjambre inicial en soluciones factibles del PMP
[fposicion_Enjambre,binario_Facilidades_Enjambre] = ...
    Decodificacion_Enjambre( matriz_Distancias,posicion_Enjambre,...
numero_Facilidades, nClientes, tamanho_Enjambre );

%Inicilizar mejor posicion de cada partticula
[ mejor_Personal ] = posicion_Enjambre;
[ fmejor_Personal ] = fposicion_Enjambre;

%Inicializar quantum enjambre
quantum = zeros( tamanho_Enjambre, nClientes * nClientes );

%Encontrar mejor global
posicion1 = find( fmejor_Personal == min( fmejor_Personal ) );

%Romper posibles empates
if length( posicion1 ) > 1
    posicion1 = posicion1( ceil( rand * length( posicion1 ) ) );
```

```
end
mejor_Global = mejor_Personal( posicion1,: );
fmejor_Global = fmejor_Personal( posicion1 );
binario_mejor_Global = binario_Facilidades_Enjambre( posicion1,: );

%inicializar los parametros alfa y beta; cc1, cc y cc3
alfa = 0.5; beta = 0.5; cc1 = 1/3; cc2 = 5/9; cc3 = (1 - cc1 - cc2);
contador = 0;


 %% Iterar el Enjambre de Particulas %%
while ( contador <= numero_iteraciones )

%Actualizar  el vector Quantum de todo el enjambre
   quantum_mejor_Personal = ( alfa * mejor_Personal ) + ...
       ( beta * ( 1 - mejor_Personal ) );
   quantum_mejor_Global = ( alfa * mejor_Global ) + ...
( beta * ( 1 - mejor_Global ) );
   quantum = ( cc1 * quantum ) + ( cc2 * quantum_mejor_Personal ) +...
       ( cc3 * repmat( quantum_mejor_Global, tamanho_Enjambre, 1 ) );


%observacion del vector quantum para evolucionar la posicion del enjambre
   observacion_Randomica = rand( tamanho_Enjambre, nClientes * nClientes
);
   posicion_Enjambre( observacion_Randomica > quantum ) = 1;
   posicion_Enjambre( ~(observacion_Randomica > quantum) ) = 0;

    %% Actualizar el Mejor Personal y el Mejor Global %%

%Decodificar enjambre inicial..solucion es una solucion factible del PMP
    [fposicion_Enjambre,binario_Facilidades_Enjambre] =...
        Decodificacion_Enjambre( matriz_Distancias,posicion_Enjambre,...
numero_Facilidades, nClientes, tamanho_Enjambre );

%Actualizar el mejor personal
[ fmejor_Personal,mejor_Personal ] =Mejor_Personal_Enjambre(...
        fmejor_Personal, mejor_Personal,fposicion_Enjambre,...
        posicion_Enjambre );

%Actualizar la mejor particula global y actualizar la mejor solucion
factible del PMP
[ mejor_Global,fmejor_Global,binario_mejor_Global ] =...
        Mejor_Global_Enjambre( fposicion_Enjambre,posicion_Enjambre,...
        fmejor_Global,mejor_Global,binario_Facilidades_Enjambre,...
binario_mejor_Global );

%aumentar contador
contador = contador + 1;
solucion = find(binario_mejor_Global);
end
```

```
 [fsolucionOptimadelPMP,solucionOptimadelPMP] =
Asignacion_Metodo_Matriz(...
solucion, matriz_Distancias, nClientes );

display( ['En ' int2str( contador )  ' iteraciones se encontro que el
desempnho de la mejor particula es ' num2str(fsolucionOptimadelPMP)...
' y las siguientes son las facilidades correspondientes 'mat2str( find(
binario_mejor_Global ) ) ] );

Toc3 = toc;

end
```

Hoja-de-Calculo ORlibrary.xlsx

Contiene la información necesaria que describe los grafos, a través de la Matriz de Adyacencia, de cada problema. El libro completo contiene 40 hojas correspondientes a cada problema.

Contenido:

- Celda A1: Numero de Vértices o Clientes
- Celda B1: Numero de Aristas que conectan los clientes.
- Celda C1: Numero de Facilidades

Las celdas de abajo describen las 'B1' aristas del grafo.

- Las dos primeras celdas indican cuales vértices conforman una arista.
- La tercera celda indica la longitud de esa arista o la distancia entre los vertices que se conectan.

Apéndice C. Análisis de algoritmos y clases de complejidad

La teoría de la computación comienza con la siguiente pregunta. Cuáles son las limitaciones y capacidades de un computador?

Esta pregunta ha sido el objeto de estudio de matemáticos logistas y cientistas de la computación desde la década de los años 30, y la interpretación que se le da a la pregunta es algo diferente dependiendo del área de estudio.

En la primera mitad del siglo 20, matemáticos como Kurt Godel, Alan Turing y Alonzo Church descubrieron que muchos problemas no pueden ser resueltos por un computador. Un ejemplo de este fenómeno es el clásico problema de determinar si existe un algoritmo que pueda evaluar la veracidad de cualquier expresión lógica de primer orden o calculo predicado. Así, desde la perspectiva de la teoría de la computabilidad, la pregunta anterior se puede reescribir de la siguiente manera. Cuales problemas puede resolver un computador y cuáles no?

El estudio de esta pregunta tuvo resultados profundos, ya que llevo al desarrollo de modelos computacionales los cuales ayudaron a la construcción de computadores reales.

Aunque muchos problemas pueden ser resueltos en teoría por un computador, en la práctica puede no ser el caso. Los problemas computacionales (que pueden ser sueltos por un computador) vienen en diferentes formas. Algunos son más difíciles de resolver que otros, por ejemplo, organizar un conjunto de millones de enteros de acuerdo a un criterio (sorting problem) es una tarea que puede realizar un computador pequeño en menos de un minuto; por otro lado, encontrar un horario óptimo para un conjunto de miles de clases de una universidad, que está restringido por condiciones de espacio y tiempo (scheduling problem) es una tarea que en teoría puede resolver cualquier computador, pero requiere siglos de procesamiento y una cantidad exorbitante de memoria incluso en un supercomputador. Esto nos lleva a pensar, que en la práctica existen problemas

que no se pueden resolver debido a su nivel de dificultad. De esta manera, Que hace que algunos problemas computacionales sean más difíciles de resolver?

Esta es la pregunta principal de la teoría de la complejidad, y la respuesta todavía no se sabe con certeza. Sin embargo, diferentes investigadores han encontrado una forma elegante y útil de clasificar los problemas de acuerdo al nivel de complejidad computacional. Por medio del uso de este esquema, se demuestra un método que da evidencia de que algunos problemas son más difíciles que otros, sin dar prueba verídica de que realmente lo son.[12]

Problema, Algoritmo y Modelo de Computación.

Antes de mencionar las principales clases de complejidad es necesario definir que es un algoritmo y que significa resolver un problema por medio de un algoritmo.

La teoría de la complejidad (NP y NP-complete) está basada en problemas de decisión. Estos problemas tienen únicamente dos posibles soluciones: "si/acepta" y "no/rechaza". La razón por la cual se restringe el análisis a este tipo de problemas es que poseen una contraparte matemática formal, denominada "lenguaje", adecuada para el estudio de la teoría de la computación. No obstante, esto no limita el estudio de los problemas de optimización, ya que todos los problemas de optimización se pueden convertir en uno de decisión si se proporciona una cota para saber si la solución encontrada supera esa cota (en el caso de maximización) y se acepta la instancia del problema. Por ejemplo, la versión de decisión del TSP se define: Dada una cota k y un grafo G, existe algún tour que visite todos los vértices del grafo y cuya distancia sea menor que k ?)[13]

[12] SIPSER, Michael. Introduction to the theory of computation.ed.2.Thomson Course Technology. Boston, Massachussets, Estados Unidos, 2006. p. 1-2

[13] GAREY, Michael R. y JHONSON, David S.. Computers and Intractability: A guide to the Theory of NP-completeness. W.H. Freeman. San Francisco, Estados Unidos, 1979. p. 19

Un lenguaje es un conjunto finito o infinito de cadenas de caracteres (string) que se construyen con un alfabeto o conjunto de símbolos. Por ejemplo, si se escoge el alfabeto binario $\{0,1\}$, los siguientes conjuntos forman lenguajes de ese alfabeto:

- $\{01, 0101001, 010, 1010, 1111, 10001, 0001\}$
- $\{01, 10, 1010, 010101, 11110000, 0101010011\}$
- $\{w \in \{0,1\}^* : w \text{ tiene el mismo numero de 0's y 1's}\}$. Esto es, w es una cadena de caracteres que pertenece al conjunto de todas las posibles cadenas de caracteres que se pueden formar con el alfabeto $\{0,1\}$, tal que w tiene la misma cantidad de los dos símbolos del alfabeto. Como se puede observar, el lenguaje anterior cumple las condiciones descritas (igual cantidad de ceros y unos), no obstante la cardinalidad del conjunto es finito por lo tanto forma un lenguaje diferente que está contenido en este lenguaje infinito (ya que es posible formar infinitas combinaciones con la misma cantidad de ceros y unos).

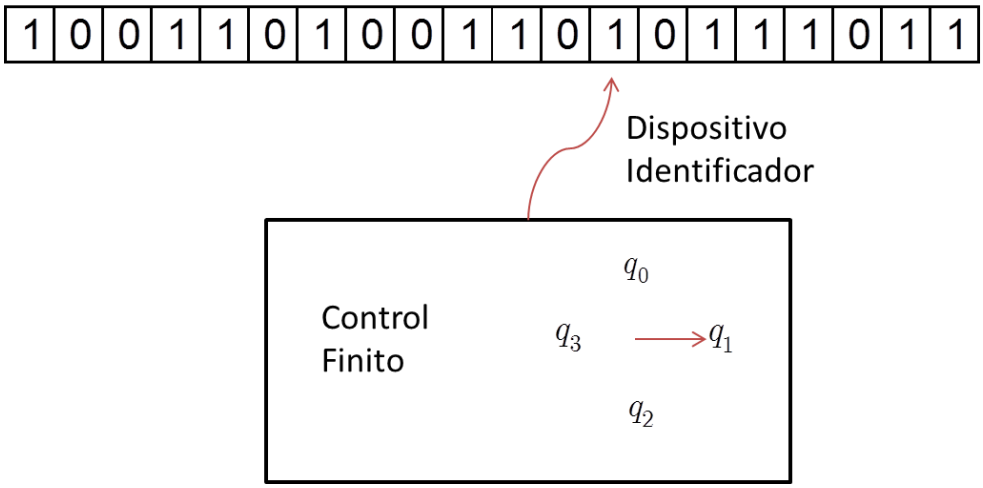
En teoría de la computación, un problema consiste en decidir si una cadena de caracteres dada (entrada) pertenece o es miembro de un lenguaje en particular. Por ejemplo la siguiente cadena de caracteres $\{1,0,1,0\}$ pertenece a los tres lenguajes descritos anteriormente, y la respuesta a cada problema es "verdad" o "si". Aunque en este caso es fácil responder la pregunta por medio de chequeo, se requiere de un método consistente que describa las propiedades intrínsecas del lenguaje y cuya única tarea sea aceptar o rechazar el certificado de membrecía de cualquier cadena de caracteres. Un método diseñado específicamente para esta tarea se denomina Aparato de Reconocimiento de Lenguaje (Language Recognition Device).Por ejemplo, el siguiente método reconoce el lenguaje $\{w \in \{0,1\}^* : w \text{ no tiene a } 111 \text{ como sub-cadena de caracteres}\}$:

1. Mantener un contador, que empieza en zero y se inicializa en zero cada vez que el símbolo 0 es encontrado en la cadena de caracteres de entrada.
2. sumar uno cada vez que el símbolo 1 sea encontrado en la cadena de caracteres de entrada.
3. Parar con respuesta "No" si el contador alguna vez alcanza tres, de lo contrario dar respuesta "Si" si se evalúa toda la cadena de caracteres y el contador nunca alcanza tres.

El método o algoritmo anterior es una representación de un LRD llamado Automata Finito. Un autómata o máquina de estado finita (FSM por sus siglas en inglés) es un modelo computacional teórico (maquina abstracta) que realiza cómputos en forma automática sobre una entrada para producir una salida. Este modelo está conformado por un alfabeto, un conjunto de estados y un conjunto de transiciones entre dichos estados. La Figura 42 muestra una caja negra que dice control finito y es la representación gráfica de la maquina abstracta. Inicialmente, la flecha roja del control finito se encuentra en el estado inicial q_0; el dispositivo lector (flecha curva) toma la cadena de caracteres de entrada y dependiendo del símbolo que registra la cadena y del estado actual del control finito hace un cambio a otro estado del sistema. Como se sabe cuál es el siguiente estado del sistema?. Dentro de la caja de control finito hay un "procesador", función de transición o tabla de transiciones que indica cual combinación de estado y símbolo produce un nuevo único estado. [14]

[14] LEWIS, Harry R. y PAPADIMITRIOU, Christos H. Elements of the theory of computation. ed. 2.PRENTICE-HAL. New Yersey, Estados Unidos 1998. p. 56-60

Figura 42. Representación grafica de un autómata finito



Fuente: LEWIS, Op cit,. p. 56

A continuación se muestra un ejemplo con la descripción del autómata que reconoce el lenguaje $\{w \in \{0,1\}^* : w \text{ does not have } 111 \text{ as s substring}\}$.

La máquina abstracta tiene un estado inicial $s = q_0$ donde inicia la computación o chequeo de la cadena de caracteres. El conjunto de posibles estados en los que puede estar la maquina es $K = \{q_0, q_1, q_2, q_3\}$. El alfabeto es binario $\sum = \{1,0\}$. Los estados de aceptación son $F = \{q_0, q_1, q_2\}$, por lo tanto el único estado donde se rechaza la cadena de caracteres es q_3. La función de transición (Tabla 2) es $\delta : K \times \sum \rightarrow K$, es decir, a cada par de elementos ordenados, en el cual el primero es un estado K y el segundo un símbolo del alfabeto $\sum$, le corresponde un único estado del conjunto K. Esta función hace que el proceso de computo sea determinístico, es decir, en cada instante de tiempo discreto la maquina solo puede estar en un único estado, contrario al modo de computo no-determinístico en el cual la función de transición se sustituye por una relación de transición, por lo

tanto la combinación de un estado y un símbolo del alfabeto produce como resultado varios estados y así en un instante de tiempo discreto la maquina puede estar en más de un estado. En la Figura 43 se muestra una ilustración del modo de computación determinístico y no-determinístico. Cada reglón separado por líneas punteadas representa un instante de tiempo discreto, y cada linea punteada representa la acción de leer un símbolo de una cadena de caracteres dada. Los tres puntos negros indica que pueden existir más ramificaciones de computación. Aunque existen ramificaciones que pueden terminar prematuramente (aceptar o rechazar antes de leer toda la cadena de caracteres debido a que el próximo símbolo combinado con el estado actual de la ramificación no tiene ninguna correspondencia en la relación de transición) solo se tiene en cuenta el estado de la ramificación más larga.[15]

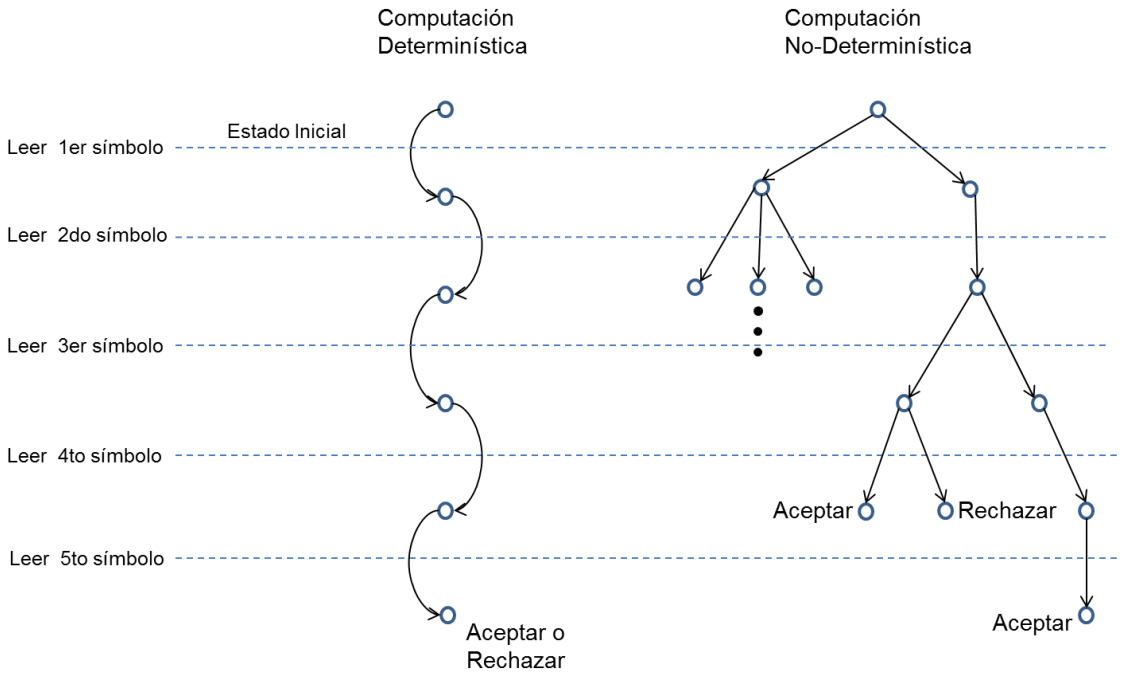
Tabla 16. Tabla de transición de estados de un autómata finito

q	σ	$\delta(q,\sigma)$
q_0	0	q_0
q_0	1	q_1
q_1	0	q_0
q_1	1	q_2
q_2	0	q_0
q_2	1	q_3
q_3	0	q_3
q_3	1	q_3

Fuente: Autor

[15] SIPSER, Op cit,. p. 47-48

Figura 43. Representación gráfica de modo de cómputo determinista y no-determinista

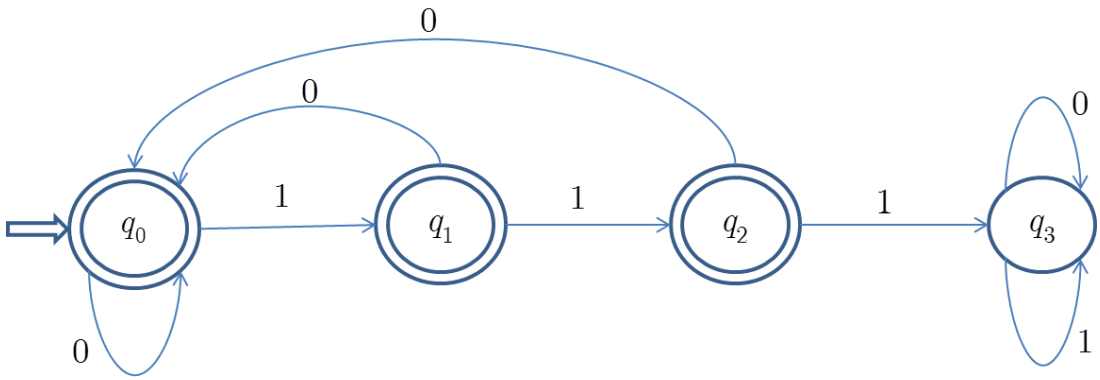


Fuente: SIPSER, Op cit,. p. 49.

La siguiente figura es un diagrama de estado de lenguaje descrito anteriormente y muestra cómo funciona el "procesador" o función de transición de la maquina abstracta. Cada estado, representado por un circulo, tiene dos flechas que representan los símbolos del alfabeto (si el alfabeto tiene 3 símbolos entonces cada estado tiene que tener 3 flechas salientes). El proceso de computación comienza en el estado q_0 y este recibe el primer símbolo de la cadena de caracteres de prueba. Dependiendo del símbolo en la primera casilla la maquina va al siguiente estado que le corresponde a ese símbolo. Sin embargo, como

reconoce la maquina una cadena de caracteres que no tenga la subsecuencia $111?$. Como se puede observar, si el dispositivo lector registra la subsecuencia 111 (y no se ha producido esta subsecuencia antes) el control finito cambia del estado q_0 a q_3 en tres pasos (instantes de tiempo discreto); a partir de ahí no importa que cantidad de 0 o 1 se reciba de la cadena de caracteres, la maquina rechaza la entrada ya que solo se requiere una subsecuencia de estas características para reconocer que una cadena de caracteres en particular no pertenece al lenguaje descrito. Si la maquina nunca llega al estado q_3 la cadena de caracteres de prueba (entrada) se acepta. Este diagrama de estado, es una forma de visualizar el método o algoritmo descrito anteriormente, en el cual se mantiene un contador y se inicializa en zero cada vez que un símbolo 0 es recibido (como muestran la flechas con el símbolo 0 que salen de los estados q_0, q_1 y q_2)

Figura 44. Diagrama de estado de un automata finito



Fuente: LEWIS, Op cit,. p. 60

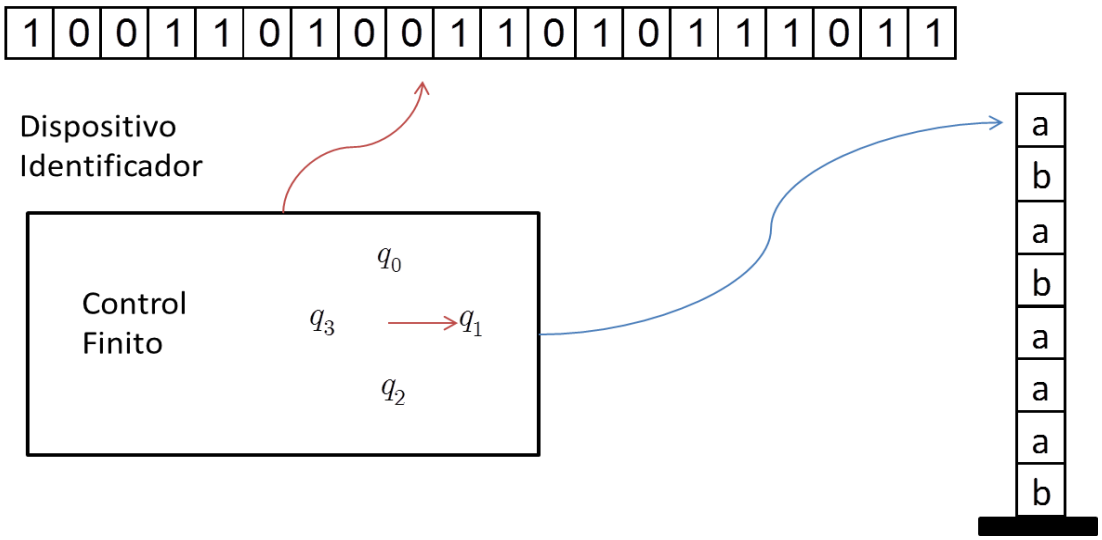
Los autómatas finitos son útiles para realizar ciertos tipos de tareas, como reconocer palabras en textos largos o analizador lexicográfico (es el componente que clasifica el código fuente en componentes lógicos como identificadores, palabras claves o reservadas y signos de puntuación) en los compiladores de lenguajes de programación. Específicamente, los autómatas finitos son útiles para reconocer un tipo de lenguajes formales denominado Lenguajes Regulares. [16]Sin embargo, existen lenguajes (o problemas) de gran importancia que los autómatas finitos no son capaces de reconocer (o resolver) debido a la limitada memoria que tienen. A continuación se hace una descripción corta de otros dos modelos de computación con más poder que los autómatas finitos.

Autómata con Pila o Pushdown Autómata

El autómata con pila es en esencia un autómata finito con una habilidad adicional: una pila (stack) sobre la cual se puede almacenar una cadena de caracteres. Esta estructura de datos permite al autómata finito "recordar" o guardar una cantidad infinita de información, sin embargo, contrario a un computador regular de escritorio que también tiene la capacidad de guardar cantidades enormes de información, el autómata con pila solamente puede tener acceso a los elementos de la estructura de datos (pila) en modo LIFO (last-in first-out).

[16]En matemáticas, lógica, y ciencias de la computación, un **lenguaje formal** es un lenguaje cuyos símbolos primitivos y reglas para unir esos símbolos están formalmente especificados. Al conjunto de los símbolos primitivos se le llama el alfabeto (o vocabulario) del lenguaje, y al conjunto de las reglas se lo llama la gramática formal (o sintaxis).

Figura 45. Representación gráfica de un autómata finito con pila



Fuente: LEWIS, Op cit,. p. 130

De especial importancia son los autómatas con pila no-determinísticos, ya que únicamente estos reconocen un conjunto completo de lenguajes denominado Lenguajes Libre de Contexto: los autómatas con pila determinísticos apenas pueden reconocer un subconjunto propio de estos lenguajes. Los Lenguajes Libres de Contexto son un tipo de lenguaje formal que se pueden describir por medio de un conjunto de reglas de formación denominado gramática libre de contexto. Aunque el propósito original de estas gramáticas, que es describir los lenguajes naturales, no ha sido alcanzado, existen dos aplicaciones de gran importancia y amplio uso:

- Las gramáticas libres de contexto permiten describir la mayoría de los lenguajes de programación, de hecho, la sintaxis de la mayoría de lenguajes de programación está definida mediante gramáticas libres de contexto.

- El desarrollo de XML (Extensible Markup Language) ha permitido la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil. Una parte esencial de XML, DTD (Documentation Type Definition) es esencialmente una gramática libre de contexto.

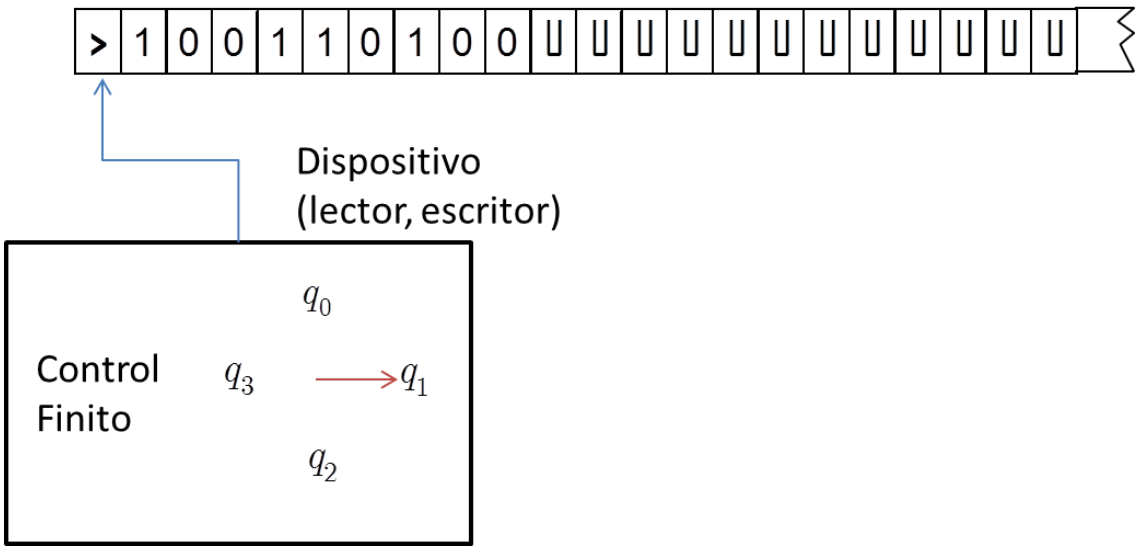
Máquina de Turing

Los autómatas finitos son buenos modelos para describir aparatos sencillos con poca memoria y los autómatas con pila para describir aparatos con una cantidad enorme de memoria de "acceso" restringido, no obstante, los modelos anteriores no pueden considerarse modelos de computación general debido a las limitaciones para reconocer lenguajes (resolver problemas) relativamente sencillos. En 1936, Alan Turing desarrollo un modelo similar a un autómata finito con memoria ilimitada y sin restricciones de acceso; este dispositivo, aunque ineficiente en poder de computación, ha sido considerado por expertos como un modelo preciso de un proceso computacional, o aquel que se puede llevar a cabo por medio de un computador. [17]

La máquina de Turing utiliza una cinta infinita como dispositivo de memoria ilimitada. Al igual que los modelos anteriores, la maquina está representada por un control finito que se encuentra en diferentes estados, y además puede leer o escribir símbolos en la cinta infinita. Aunque la cinta es infinita, uno de los lados tiene un límite sobre el cual se empieza a escribir la cadena de caracteres o entrada de la cual se va a evaluar membrecía en el lenguaje de la maquina en particular.

[17]HOPCROFT, Jhon E.; MOTWANI, Rajeev y ULLMAN, Jeffrey D. Introduction to Automata Theory, Languages and Computation.ed.2.PEARSON EDUCATION. p. 307

Figura 46. Representación Gráfica de la máquina de Turing simple



Fuente: LEWIS, Op cit,. p. 181

La máquina abstracta tiene un conjunto de estados finitos Q, un conjunto de símbolos (alfabeto) para la cadena de caracteres (entrada) $\sum$, un conjunto de símbolos (alfabeto) permitidos en la cinta infinita Γ, donde $\sum \subseteq \Gamma$; un estado inicial q_0 y subconjuntos de estados: de aceptación y rechazo. El "procesador" o función de transición de la máquina de Turing es $\delta : Q \times \Gamma \to Q \times \Gamma \times \{\mathrm{L,R}\}$. Esto es, cuando la maquina esta en cierto estado q y el dispositivo lector-escritor esta sobre una celda de la cinta que contiene el símbolo a, entonces el comando $\delta(q,a) = (r,b,L)$ hace que el dispositivo escriba el símbolo b en la celda de a y el control finito va al estado r. El tercer componente indica si el dispositivo lector-escritor se mueve hacia la derecha (R de right) o hacia la izquierda (L de left), en este caso el comando indica un movimiento hacia la izquierda.

Existe una asimetría respecto a la forma en que la máquina de Turing define un lenguaje en particular. Cuando una maquina computa sobre una cadena de caracteres (entrada), existen tres resultado posibles: la maquina acepta, rechaza o corre para siempre (looping). [18]

Una máquina de Turing diseñada para detenerse únicamente cuando se acepta la entrada, se dice que reconoce o "semidecide" el lenguaje, ya que la maquina tiene únicamente dos opciones: acepta el lenguaje y para, o continúa el proceso de computación por siempre. En este sentido existe una asimetría, ya que si una maquina tarda mucho tiempo en procesar un cadena de caracteres de entrada es imposible saber si esa entrada pertenece al lenguaje y la maquina simplemente se tarda mucho tiempo, o definitivamente la entrada no pertenece al lenguaje. Este tipo de lenguajes se denominan Lenguajes Recursivamente Enumerables, Parcialmente Decidible o Turing-Reconocible.

Existe un tipo de máquina de Turing que dada una cadena de caracteres de entrada, en algún momento se detiene ya sea para aceptar o rechazar: la maquina no tiene la opción de computar por siempre. Cuando una máquina de estas escanea toda la cadena de caracteres y acepta, se dice que "decide" el lenguaje. Los tipos de lenguajes que aceptan o "deciden" estas máquinas se denominan Lenguajes Recursivos, Turing-Decidible o Decidible. Este conjunto de lenguajes es un subconjunto propio de los Lenguajes Recursivamente Enumerables mencionados en el párrafo anterior. La importancia de esta máquina abstracta yace en que es la representación más precisa de la noción intuitiva que se tiene de un algoritmo: nada que no pueda ser computado por una máquina de Turing puede ser considerado un algoritmo. Esta aseveración se conoce como la tesis de Chuch-Turing.

La tesis de Church-Turing es una hipótesis (proposición aceptable que ha sido formulada a través de la recolección de información y datos)ampliamente

[18] PAPADIMITRIOU, Christos H. Computational Complexity. ADDISON WESLEY LONGMAN. New York, Estados Unidos, 1995. p. 24-26.

aceptada, no obstante no siendo un teorema, no puede ser demostrada y simplemente afirma que: cierto concepto informal denominado "algoritmo" o "método efectivo de cálculo" corresponde a una máquina de Turing que se detiene sobre cualquier entrada. De igual forma, afirma que cualquier intento general de definir un modelo para computar *algoritmos,* terminara en un modelo que solo puede computar funciones parciales recursivas, que es equivalente a lo que un computador o máquina de Turing puede decidir. Aunque es casi imposible encontrar una demonstración (en parte debido a la falta de consistencia en la definición de algoritmo o método efectivo de cálculo), es teóricamente posible desaprobar esta tesis encontrando un modelo de computación razonable más poderoso que pueda resolver problemas (decidir lenguajes) que la máquina de Turing no puede[19] .Varios sistemas como Lambda-Calculus (Alonzo Church) y Funciones Recursivas (Kurt Godel y Stephen Klenee) fueron propuestos como modelos generales de computación, y todos demostraron ser equivalentes en poder a la máquina de Turing. No obstante, comparado con los sistemas mencionados, la máquina de Turing es considerado el sistema más sencillo, elegante y comprensible que expresa el concepto de computabilidad efectiva.

Modelos Alternativos de la máquina de Turing , llamados variantes de la máquina de Turing, se han propuesto como modelos generales de un computador real; aunque unos son más eficientes que otros en el tiempo de ejecución, todos son equivalente en poder o capacidad de reconocer los mismos lenguajes. Esta característica de invarianza ante ciertos cambios se denomina Robustez, y por esta razón la máquina de Turing es considerado un modelo Robusto.[20] A continuación se mencionan las principales variantes.

- Máquina de Turing Multi-Cinta: Es una máquina de Turing que tiene k dispositivos lector-escritor que operan sobre k cintas.

[19] LEWIS. Op cit,. p. 245-247
[20] SIPSER. Op cit,. p. 148

- Random Access Turing Machine: Es una maquina abstracta que opera sobre una estructura de datos llamada arreglo de registros. El arreglo de registros es finito y tiene la capacidad de contener enteros de gran tamaño, que representa la dirección de las celdas de la cinta de la máquina de Turing. La ventaja de este dispositivo, es que tiene acceso a cualquier celda de la cinta en tiempo constante, contrario a la máquina de Turing que el acceso a diferentes celdas es secuencial.
- Máquina de Turing No-Determinista: Es una maquina abstracta que es utilizada en experimentos mentales para evaluar las capacidades y limitaciones de los computadores. A diferencia de la máquina de Turing determinista, la función de transición es una relación de transición, por lo tanto cada configuración (estado actual del control finito, contenido actual de la cinta y ubicación actual del dispositivo lector-escritor) está relacionada con al menos una o ninguna configuración. Al igual que en la maquina sencilla, la maquina no determinista puede reconocer un lenguaje o decidir un lenguaje. Cuando la maquina está diseñada para reconocer, se requiere que al menos una rama de computación acepte la entrada y las otras pueden seguir computando por siempre. De manera similar, para decidir un lenguaje se requiere que al menos una rama de computación acepte la entrada y las demás pueden rechazar. El tiempo de computación de una máquina que decide un lenguaje es el tiempo de ejecución de la ramificación más larga. [21]

  Existen dos formas de visualizar la forma como opera una maquina no determinista: la primera es pensar que la maquina tiene la capacidad de ramificar y ejecutar al mismo tiempo varios procesos, alguna o varias de estas ramificaciones aceptan, o ninguna entra en esa configuración. La otra forma de visualizar, que es bastante útil en la definición de una clase de

[21] Non-determnistic Turing Machines [Anonimo]. Disponible en: http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Non-deterministic_Turing_machine.html

problemas, es pensar en una máquina que tiene la capacidad adicional de "adivinar" en cada instante la mejor configuración que lleva finalmente a aceptar el lenguaje; es decir, la maquina nodetermnisticamente adivina alguna de las ramificaciones que aceptan la entrada.. Aunque la maquina nodeterminista tiene la misma capacidad que la maquina determinista, la primera no es un modelo razonable de computación. La característica de No-determinismo juega un papel central en complejidad computacional, no tanto por su afinidad con el concepto de computabilidad, si no por su afinidad con las aplicaciones de la computación como: la lógica, matemática combinatoria y inteligencia artificial. [22]

Cuando se implementa una solución para un problema de decisión u optimización, por que se utilizan lenguajes de programación con comandos diferentes a los de la máquina de Turing?. Usualmente es muy tedioso programar un algoritmo con el nivel de detalle requerido por la máquina de Turing, por lo tanto, se abstraen funcionalidades y detalles de este dispositivo y se crean lenguajes de programación de más alto nivel que permiten una descripción más comprensible (parecido a las instrucciones de un lenguaje natural) de un algoritmo para resolver un problema en particular[23]. La mayoría de los lenguajes de programación (lenguajes de programación funcionales: LISP, Haskell; lenguajes de programación de objeto orientado: Java, Python; lenguajes de programación multi-paradigma: C++) son llamados Turing Complete. Los lenguajes de programación Turing Complete son un sistema de reglas para manipular datos que operan equivalente en capacidad a la máquina de Turing que se detiene sobre una entrada.

[22]PAPADIMITRIOU. Op cit,. p. 49-50
[23] SIPSER, Op cit,. p. 157

A continuación se presenta una descripción más sencilla y comprensible (libre de detalles correspondientes a la máquina de Turing) de lo que es un algoritmo para introducir el análisis de algoritmos.

Análisis de Algoritmos

Un algoritmo es un método paso a paso para resolver un problema en particular. Aunque existen diferentes tipos de algoritmos, el presente trabajo está relacionado con los algoritmos que se pueden ejecutar en una computadora "tradicional", es decir, una computadora personal con un solo procesador que ejecute instrucciones paso a paso. [24]

En general, los algoritmos poseen las siguientes características.

- *Entrada* El algoritmo recibe datos de entrada
- *Salida* El algoritmo produce una salida
- *Precisión* los pasos se establecen con precisión
- *Determinismo* los resultados intermedios de cada paso de ejecución son únicos y están determinados solo por las entradas y los resultados de los pasos anteriores.
- *carácter finito* el algoritmo termina, es decir, se detiene después de ejecutar un numero finito de instrucciones
- *Corrección* la salida producida por el algoritmo es correcta, es decir, el algoritmo resuelve el problema sin errores
- *Generalidad* el algoritmo se aplica a un conjunto de entradas

El análisis de algoritmos cosiste en determinar estimaciones teóricas de los recursos (tiempo y espacio) que necesita cualquier algoritmo que resuelva un problema computacional dado. Sin embargo, en este apartado solo se hace referencia al tiempo.

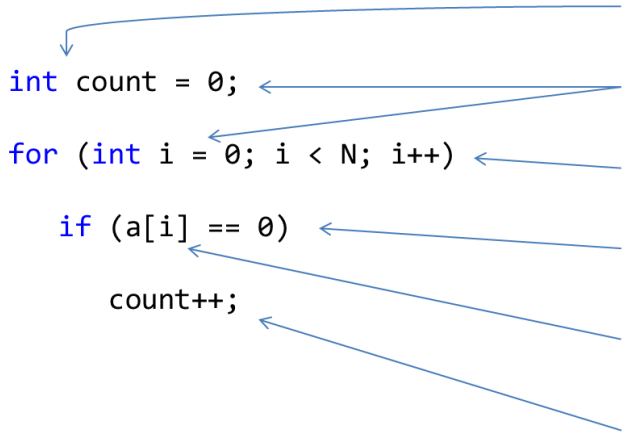
[24] JOHNSONBAUGH, Richard. Matematicas Discretas. ed. 6.Pearson Education.Mexico, 2005. p. 145

Una forma de hacer estimaciones es obtener modelos matemáticos precisos del tiempo de corrida de un algoritmo. Este método consiste en identificar todas las operaciones básicas del programa (operaciones aritméticas básicas, comparaciones, declaraciones de variable, acceso a una matriz y otras), calcular el costo de cada operación, hallar la frecuencia de ejecución y calcular el tiempo total de corrida que es la sumatoria del costo de ejecución de cada operación en una corrida completa. Cuando se calcula el costo de cada operación básica se utiliza un método, como la media de una muestra, para estimar un valor constante que represente el costo de esa operación independiente de sus operadores; esta asunción es útil para el análisis teniendo en cuenta que la variación de tiempo en la ejecución de estas operaciones está en un rango menor a los nanosegundos[25].

Para ilustrar este método se utiliza dos problemas sencillos. El primer problema es: Dado un conjunto de enteros, contar el número de subconjunto de UN elemento que sumen cero?. A continuación se muestra un código simplificado en Java de un algoritmo que resuelve el problema por medio busqueda exaustiva o fuerza bruta sobre un arreglo a.

[25]SEDGEWICK, Robert y WAYNE, Kevin. Algorithms Part 1 [On line]. University of Princeton. 4 de Febrero de 2013. video 2-3. [Citado el 19 de Marzo de 2013; 17:00:00]. Disponible en: https://www.coursera.org/

Figura 47. Código para resolver el problema 1-sum



Operación Básica	Frecuencia
Declaración de Variable	2
Asignación de variable	2
Comparación <=	N+1
Comparación ==	N
Acceso al arreglo a[]	N
Incremento	0 *a* N

Fuente: SEDGEWICK. Op cit,.video 2-3. "Mathematical Models"

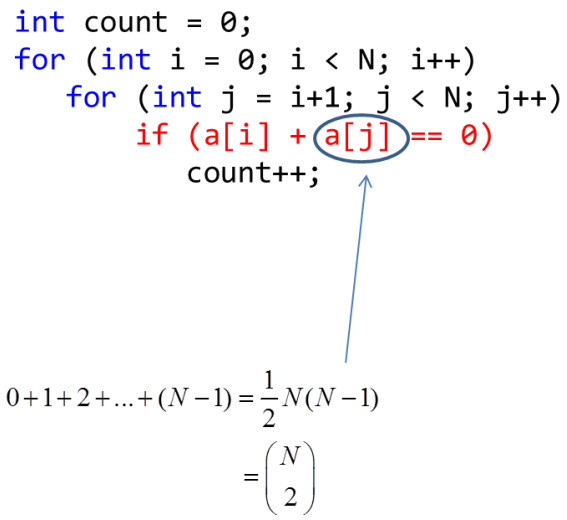
Si se multiplica la frecuencia y el costo de cada operación y se suman los valores resultantes (Figura 48) se obtiene un modelo preciso del tiempo de corrida del algoritmo. Desafortunadamente, este método es poco práctico cuando se trabajan problemas grandes debido a la cantidad de detalles que se tienen que tener en cuenta. A continuación se muestra el mismo proceso para el problema 2-sum (dado un conjunto de enteros, cuantos subconjuntos de 2 elementos suman cero?)

Figura 48. Modelo matemático de costo (tiempo) para un algoritmo

$$\text{Tiempo Total de Corrida} = \sum f_1 * \text{costo}_1 + f_2 * \text{costo}_2 + f_3 * \text{costo}_3 + \ldots$$

Fuente: Autor

Figura 49. Código para resolver el problema 2-sum



Operación Básica	Frecuencia
Declaración de Variable	$N+2$
Asignación de variable	$N+2$
Comparación <=	$1/2(N+1)(N+2)$
Comparación ==	$1/2(N)(N-1)$
Acceso al arreglo a[]	$(N)(N-1)$
Incremento	$1/2(N)(N-1)$ a $N(N-1)$

Fuente: SEDGEWICK. Op cit,.video 2-3. "Mathematical Models".

El algoritmo utiliza un arreglo a de tamaño N para guardar los datos de entrada que es el conjunto de enteros. Sobre esta estructura utiliza dos variables iterativas (i, j) como índice de la estructura para escanear todas las posibles combinaciones de pares. Como se puede observar, el cálculo de frecuencia de las operaciones básicas requiere de técnicas matemáticas (conteo) más sofisticadas; nuevamente, aunque en este caso es posible hacer la descripción completa, existen problemas prácticos de gran interés para los cuales seria extremamente difícil hacer esta tabulación. Por consiguiente, para obtener un modelo de costo sencillo y útil, se hacen dos simplificaciones. Primero, se escoge una única operación básica, la más representativa, como parámetro para el tiempo de corrida del algoritmo. En este caso, la operación básica que más se ejecuta y que

define el resultado del problema (cuantos pares que sumen cero?) es el acceso al arreglo a (comando en color rojo de la Figura 49). De esta forma, el modelo de costo se reduce a la ecuación: $\mathrm{N(N\text{-}1)=N^2\text{-}N}$

La segunda simplificación es ignorar todos los términos de grado inferior. Esto es posible, ya que en instancias grandes de los problemas el grado superior de la ecuación de costo predomina sobre los grados inferiores. Por ejemplo, $x^2 \approx x^2 + x$ cuando $x > 1.000'000.000$ o $x \to \infty$.Así mismo, en los términos inferiores se refleja el tipo de codificación o implementación utilizada; teniendo en cuenta que el objetivo del análisis de algoritmos es hacer estimaciones respecto a la propiedades inherentes del algoritmo independiente de factores ajenos, se justifica el procedimiento descrito. Esta notación se denomina Tilde Notatión o Notación Virgulilla, y consiste en reducir la ecuación a su término dominante multiplicado por una constante. Específicamente, la Notación Virgulilla indica una igualdad aproximada asintótica (cuando se tiende a infinito) de dos funciones, por lo tanto representan buenos modelos aproximados de las funciones de costo de un algoritmo. La definición técnica es: $f(N) \sim g(N)$ significa $\lim_{N\to\infty} \frac{f(N)}{g(N)} = 1$. A continuación se muestra el costo de las operaciones básicas del problema 2-sum utilizando Notación Virgulilla, el reglón en rojo representa la ecuación de la primera simplificación, y la tercera ecuación de ese reglón es un modelo aproximado del tiempo de corrida del algoritmo que resuelve el problema 2-sum por medio de fuerza bruta.

Tabla 17. Notación Virgulilla de las Operaciones Básicas del problema 2-sum

Operación Básica	Frecuencia	Notación Virgulilla
Declaración de Variable	$N+2$	$\sim N$
Asignación de variable	$N+2$	$\sim N$
Comparación <=	$1/2(N+1)(N+2)$	$\sim \frac{1}{2} N^2$
Comparación ==	$1/2(N)(N-1)$	$\sim \frac{1}{2} N^2$
Acceso al arreglo a[]	$N(N-1)$	$\sim N^2$
Incremento	$1/2(N)(N-1)$ *a* $N(N-1)$	$\sim \frac{1}{2} N^2$ *a* $\sim N^2$

Modelo Aproximado del tiempo de corrida del problema 2-sum.

Fuente: SEDGEWICK. Op cit,.video 2-4. "Order-of-growth Classifications".

La segunda simplificación, reduce las estimaciones a los siguientes ordenes de crecimiento (order-of-growth) asintotico:

Tabla 18. Ordenes de Complejidad

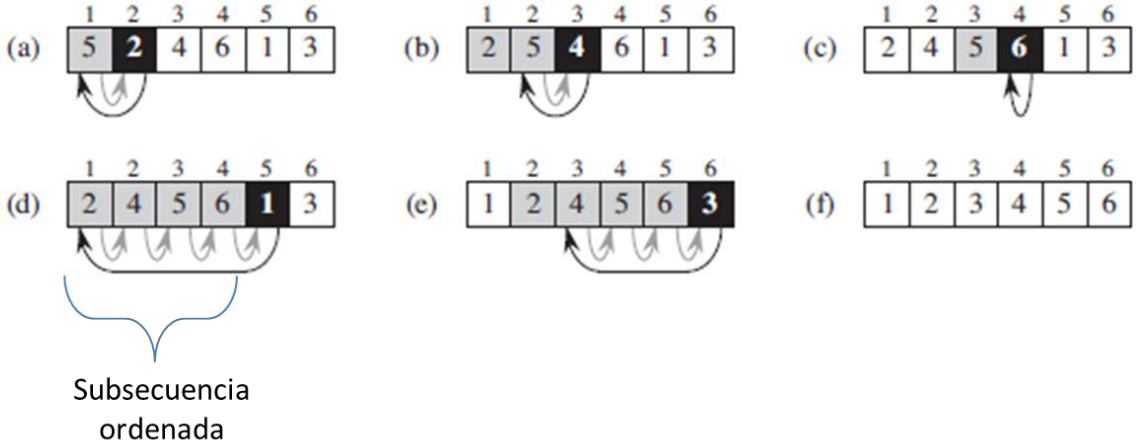
1	orden constante
log n	orden logarítmico
n , n log n	orden lineal
n^2	orden cuadrático
n^a	orden polinomial (a > 2)
a^n	orden exponencial (a > 2)
n!	orden factorial

Fuente: Autor

EL modelo de costo no es suficiente para el análisis de la eficiencia de un algoritmo, el modelo o estructura de la entrada de un algoritmo para un tamaño dado puede causar variabilidad en el tiempo de corrida. A continuación se ilustra con un ejemplo sencillo este fenómeno y los tres tipos de análisis que surgen.

El *sorting problem* o problema de ordenar es: dada una secuencia de elementos ordenar la secuencia de acuerdo a algún criterio. En este caso se utiliza números enteros y el criterio es en forma ascendente. Existen varios algoritmos para resolver este problema, y a continuación se hace el análisis de Insertion Sort.
La forma como opera Insertion Sort se parece a la forma como se ordenan las primeras cartas de una naipe. Como se ilustra en la Figura 50, se empieza tomando el segundo elemento, se compara con el primero y se ordena formando la primer subsecuencia ordenada. Luego se mira el siguiente elemento (carta) adyacente a la subsecuencia ordenada y se ordena dentro de esta subsecuencia. Este proceso se repite hasta que no exista el siguiente elemento adyacente a la subsecuencia actual debido a que la subsecuencia alcanzo el tamaño de la secuencia entera.

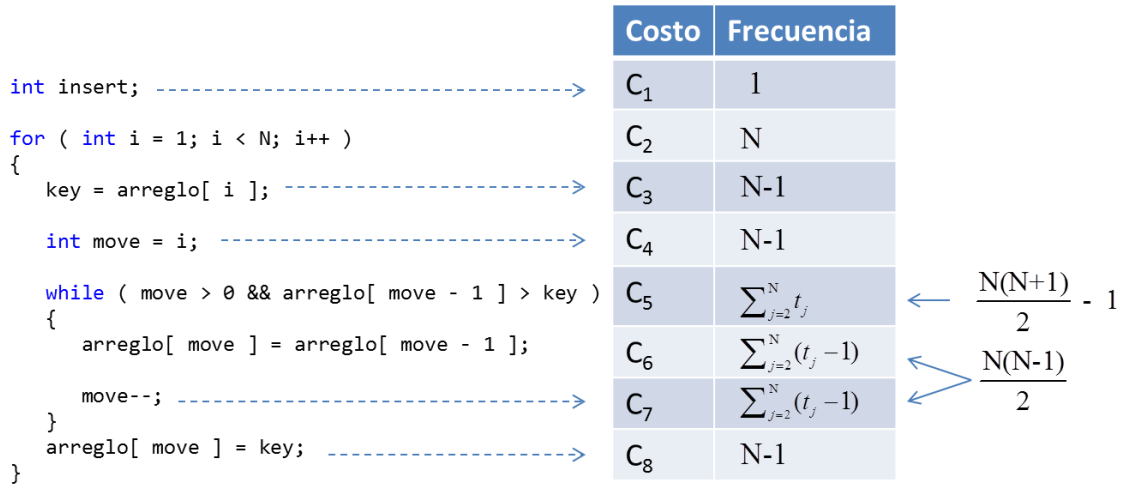
Figura 50. Movimientos básicos de Insertion Sort



Fuente: CORMEN, Thomas H. *et al*. Introduction to Algorithms.ed. 3.THE MIT PRESS.Cambridge, Massachusetts, Estados Unidos, 2009. p. 18.

En la figura anterior, se observa que la entrada del algoritmo fue la secuencia $(5,2,4,6,1,3)$. Que ocurre con el tiempo de corrida si la secuencia de entrada es $(2,4,5,6,1,3)$?. Primero se va a revisar el pseudocodigo del algoritmo con la frecuencia de ejecución de los pasos básicos (se hace distinción de paso básico y operación básica, ya que un paso básico puede contener varias operaciones básicas que resultan en un paso con tiempo constante.)

Figura 51. Código de Insertion Sort



Fuente: CORMEN. Op cit,. p. 18.

Para la secuencia $(5,2,4,6,1,3)$, cada vez que se selecciona el elemento de comparación (key en el pseudocodigo y celda negra en la Figura 50) se supone que los elementos anteriores ya están ordenados. En el while loop, se pregunta si el elemento de comparación (key) es menor que el elemento anterior, es decir, el último elemento de la subsecuencia ordenada. Si key es menor que el elemento anterior, se tiene que reubicar este elemento dentro de la lista y por lo tanto se ejecutan los comandos dentro del while loop. Ahora se revisa la secuencia $(2,4,5,6,1,3)$. Cuando se escoge a 4 como key, no se ejecuta el while loop ya que 2 es menor. Cuando se escoge 5 tampoco se ejecuta ya que $(2,4,5)$ es una subsecuencia ordenada y el último elemento de esta subsecuencia es menor, y lo mismo ocurre cuando se escoge 6. El while loop solo se ejecuta 2 veces (para 1 y 3) cuando la secuencia $(2,4,5,6,1,3)$ es la entrada, en cambio se ejecuta todas las veces cuando la entrada es $(5,2,4,6,1,3)$. Esto indica que para un mismo tamaño de entrada, existen distintas estructura de la entrada que causa variabilidad en el tiempo de corrida.

En el caso más extremo, que ocurre si la entrada es $(1,2,3,4,5,6)$?. El while loop nunca se ejecuta y lo máximo que tiene que hacer el algoritmo para ordenar esta secuencia es hacer un chequeo lineal. Este caso de análisis se llama Mejor Caso de Análisis (Best-Case Analysis) y consiste en hacer estimaciones del mínimo tiempo necesario para ejecutar el algoritmo entre todas las entradas (instancias) de tamaño N. El Mejor Caso en este problema ocurre cuando la secuencia de entrada ya está ordenada. De manera similar el Peor Caso de este problema, que corresponde al análisis del tiempo máximo entre todas las instancias, ocurre cuando la secuencia de entrada está "totalmente desordenada" u ordenada en dirección contraria; en este caso, el while loop se ejecuta todas la veces, así mismo, los comandos dentro del while loop se ejecutan el número máximo posible de veces ya que cada elemento de comparación debe ser arrastrado hasta el comienzo de la secuencia entera. En algunos casos también es útil conocer el Caso Promedio de tiempo de corrida de un algoritmo, este caso se puede interpretar como el tiempo de corrida esperado para una entrada aleatoria de un tamaño dado. Para el Caso Promedio, se requiere de análisis probabilístico para asignar distribuciones a las entradas de los problemas y de esta manera obtener información útil del conjunto de instancias.

Aunque en distintas áreas de aplicación se utilizan diferentes tipos de análisis, cuando se clasifican algoritmos de acuerdo al orden de complejidad se prefiere el análisis del Peor Caso. Primero, el análisis del peor caso proporciona una cota superior del tiempo de corrida entre todas las entrada, por lo tanto es un certificado de que ninguna entrada tomara más tiempo que el especificado por la cota. Segundo, en muchos problemas el Peor Caso ocurre frecuentemente, por ejemplo en el algoritmo de búsqueda sobre base de datos es frecuente que el elemento buscado no se encuentre en la base por lo tanto es necesario escanear

toda la estructura. Finalmente, en varios problemas el Caso Promedio es más cercano al Peor Caso, ejemplo Insertion Sort.[26]

En el estudio del comportamiento asintótico de las funciones de tiempo existen diferentes notaciones para hacer una caracterización del algoritmo. Hasta el momento se describió la Notación Virgulilla que permite reducir la ecuación a su término dominante multiplicado por una constante. Aunque son buenos modelos aproximados para estimar el tiempo de corrida de un programa en partícular, esa constante es el efecto de las propiedades del sistema o maquina en la cual se implementa el algoritmo, y nuevamente el objetivo es abstraer lo esencial de esos factores ajenos.[27]

La notación más aceptada para clasificar algoritmos es Big Theta o $\Theta(g(n))$, debido a que se considera una cota estrecha de la función de tiempo en algún tipo de análisis. Se dice que $f(n) = \Theta(g(n))$ o " f de n esa theta de g de n " si excepto por factores constantes y un número finito de excepciones, la función $f(n)$ está acotada arriba y abajo por $g(n)$, por lo tanto f y g crecen a la misma velocidad. En la Figura 52.a se muestra una representación grafica de la funciones f y g.

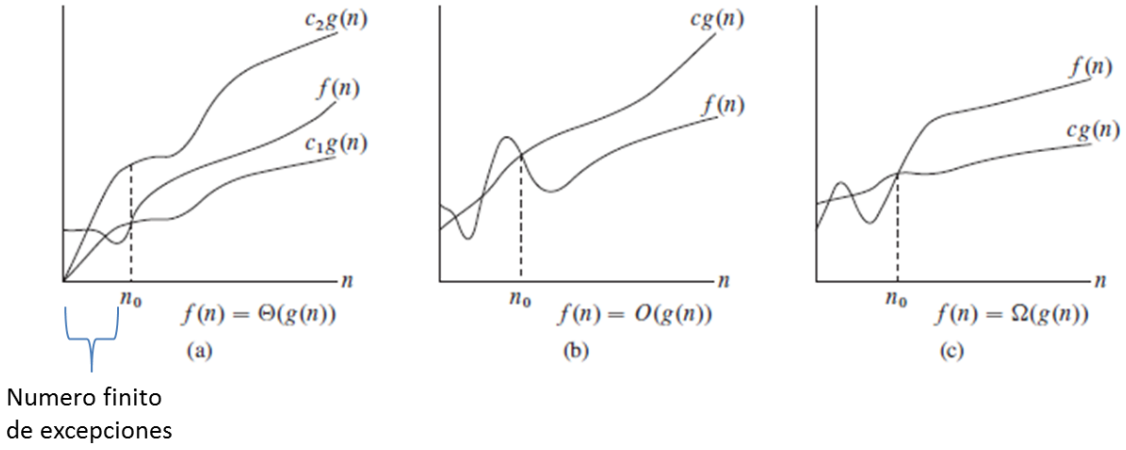
Otra notación ampliamente utiliza debido a su gran utilidad en análisis del Peor Caso es Big Ohmicron o $O(g(n))$. Se dice que $f(n) = O(g(n))$ o " f de n es Big-Oh de g de n " si excepto por un factor constante y un número finito de excepciones, la función $f(n)$ está acotada arriba por $g(n)$. Esta notación busca establecer cotas superiores de la función de tiempo, no obstante es no es una forma fuerte de clasificar algoritmos ya que no es una cota estrecha asintótica y permite cotas superiores triviales en el análisis de una función. Por ejemplo, el problema 2-sum descrito anteriormente es $O(n^2)$ en el peor caso, sin embargo, también es $O(n^3)$ o

[26] CORMEN, Op cit,. p. 28.
[27] SEDGEWICK. Op cit,.video 2-5. "Theory of Algorithms"

$O(2^n)$ en el peor caso. En la literatura es utilizada para caracterizar algoritmos ya que el análisis de cotas superiores estrechas en el peor caso implica cotas superiores para todo el conjunto de entradas de un problema. En la Figura 52.b se muestra una representación gráfica de una cota superior de la función f.

La notación Big Omega o $\Omega(g(n))$ proporciona cotas inferiores de la función de tiempo bajo algún tipo de análisis. Se dice que $f(n) = \Omega(g(n))$ o "f de n esa Big-Oh de g de n" si excepto por un factor constante y un número finito de excepciones, la función $f(n)$ está acotada abajo por $g(n)$. Es utilizada en el análisis de Mejor Caso para determinar el mínimo tiempo de corrida en el mejor caso y por ende entre todas las entradas del problema de un tamaño dado. En la Figura 52.c se muestra una representación gráfica de una cota superior de la función f.

Figura 52. Notación Asintótica



Fuente: CORMEN, Op cit,. p. 45

Un error común es asociar estrictamente el mejor caso con Big Omega y el peor caso con Big Ohmicron. Las tres notaciones pueden ser utilizadas en los tres tipos de análisis descritos. " f de Big Theta de g de n " en el peor caso significa que la función está acotada por arriba y por abajo en el peor caso; luego esto implica que la función en el peor caso es Big Ohmicron de g de n. Así mismo, no es contradictorio decir que el peor caso es Big Omega de g de n ya que la forma más fácil de resolver el peor caso de f es g de n.[28]

A continuación se hace un análisis del peor caso y del mejor caso de Insertion Sort. Para esto, es necesario revisar la Figura 51, donde se muestra la frecuencia y costo de cada paso básico. En el peor caso, la secuencia está totalmente desordenada por lo tanto el algoritmo ejecuta todos los pasos el máximo número de veces; el modelo matemático del tiempo de corrida en el peor caso es:

$$T(N) = c_1 + c_2 n + c_3(n-1) + c_4(n-1) + c_5(\frac{n(n+1)}{2} - 1) + c_6(\frac{n(n-1)}{2}) + c_7(\frac{n(n-1)}{2}) + c_8(n-1)$$
$$= (\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2})n^2 + (c_2 + c_3 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8)n - (c_3 + c_4 + c_5 + c_8 + c_1)$$
$$= an^2 + bn + c$$

Ahora, un cota asintótica estrecha del peor caso se obtiene si ignoramos los términos de grado inferior y la constante del grado superior. De esta manera $T(n) = \Theta(n^2)$ en el peor caso. Para demostrar que la función es theta de n^2 se debe proporcionar dos constantes que acoten la función por arriba y abajo para un conjunto infinito mayor que n_0. La cota superior se puede determinar de la siguiente manera: primero, es necesario tener en cuenta las siguientes afirmaciones $1 < n^2, \forall n > 1$ y $1 < n, \forall n > 1$ (algo obvio pero útil al momento de construir la demonstración). Si $n = n$ y se multiplica por $1 < n$ entonces $n < n^2, \forall n > 1$, de la misma manera si se multiplica por una constante b entonces $bn < bn^2, \forall n > 1$.(a) Ahora, si $1 = 1$ se multiplica por $1 < n^2$ y por una constante c

[28]CORMEN. Op cit,. p. 49

entonces $c < cn^2, \forall n > 1$.(b) Finalmente, se tiene que $an^2 = an^2$ (c); si se suman las expresiones anteriores (a, b y c) se obtiene $an^2 + bn + c < an^2 + bn^2 + cn^2, \forall n > 1$. Esta última expresión se puede simplificar a $an^2 + bn + c < (a + b + c)n^2, \forall n > 1$ para obtener una constante $C_1 = |a| + |b| + |c|$ que acota por arriba la función $T(n)$ para todo $n > 1$ en el Peor Caso. Para obtener una constante y acotar por abajo la función $T(n)$ en el peor caso se puede observar que $an^2 + bn + c < an^2, \forall n > 1$. Luego, una constante que acota la función por abajo para todo n mayor o igual que 1 es $C_2 = a$. De esta forma, se demuestra que existen constantes C_1 y C_2 que acotan el peor caso de la función $T(n)$ para todo $n \geq 1$. El hecho de que $f(n) = O(n^2)$ en el peor caso, implica que $f(n) = O(n^2)$ sobre todas las entradas de tamaño n.No obstante, es importante no confundir $f(n) = \Theta(n^2)$ en el peor caso con $f(n) = \Theta(n^2)$ sobre todo el conjunto de entradas; aunque la cota superior en los dos casos puede coincidir, en definitiva es diferente una cota inferior para el peor caso y el mejor caso, que es donde se acota por debajo el conjunto completo de entradas, como se muestra a continuación en el análisis del Mejor Caso de Insertion Sort.

Para comenzar el análisis del Mejor Caso se hace un modelo matemático para esta estructura de entrada. Como se mencionó antes, el Mejor Caso ocurre cuando la entrada ya está ordenada, de esta manera el comando while loop (Figura 51) no se repite $\frac{N(N+1)}{2} - 1$ veces si no $N - 1$ veces, además los comandos dentro del while loop nunca se ejecutan en la corrida. Si se ignora el aporte de costo de estos pasos se obtiene el siguiente modelo matemático del Mejor caso.

$$\begin{aligned} T(N) &= c_1 + c_2 n + c_3(n-1) + c_4(n-1) + c_5(n-1) + c_8(n-1) \\ &= (c_2 + c_3 + c_4 + c_5 + c_8)n - (c_3 + c_4 + c_5 + c_8 + c_1) \\ &= an + b \end{aligned}$$

Si se ignora los términos de grado inferior y la constante del grado superior se obtiene que $f(n) = \Theta(n)$ para el mejor caso. Esto implica que $f(n) = \Omega(n)$ tanto para el mejor caso como para el conjunto total de entradas de tamaño n. De esta manera podemos decir que la función de tiempo de corrida de Insertion Sort es $f(n) = O(n^2)$ y $f(n) = \Omega(n)$ para el conjunto total de entradas.

Relación entre las clases de Complejidad

Aunque los modelos computacionales mencionados (Maquina Turing Multi-cinta, Random Access Turing Machine y maquina No determinista de Turing) tienen la capacidad de resolver (reconocer) el mismo conjunto de problemas (lenguajes), no todos ejecutan esta tarea con la misma eficiencia.

Cada máquina de Turing multi-cinta puede ser simulada por una máquina de Turing simple con una pérdida polinomial de eficiencia. Es decir, si una maquina multi-cinta resuelve (decide) un problema (lenguaje) en un tiempo dado, es posible diseñar una máquina de una sola cinta que resuelva el mismo problema, en un tiempo de ejecución mayor que está relacionado de forma polinomial con el tiempo de ejecución de la maquina multi-cinta. Este fenómeno ocurre también con el modelo RAM y otras variantes determinísticas, por lo tanto se dice que estos modelos computacionales son polinomialmente equivalentes.[29]

Sin embargo, existe una gran diferencia con la maquina no determinista, ya que esta puede ser simulada por una maquina determinista (de una cinta o multi cinta), pero con pérdida de eficiencia exponencial. Aunque no se sabe con certeza si está perdida exponencial es inherente al modelo o una consecuencia del limitado

[29] SIPSER.Op cit,. p. 149

entendimiento de que se tiene del No-determinsimo, esta distinción es importante ya que define las dos clases más importantes de problemas: P y NP. [30]

Como se demostró en el apartado anterior, el Sorting Problem puede ser resuelto en tiempo polinomial por medio del algoritmo Insertion Sort. Una forma de ver la ventaja del orden de complejidad de este algoritmo es que permite ordenar, en el peor caso, listas de millones de elementos en pocos minutos/segundos.[31] No obstante, existen problemas como el Traveling Salesman Problem que aunque puede ser resuelto por un algoritmo, si se analiza el peor caso de cualquier algoritmo conocido que resuelva ese problema, se termina con cotas superiores de tiempo de corrida exponencial. El TSP consiste en: dado un conjunto de ciudades, cual es el recorrido que tiene menos costo (tiempo, distancia o dinero), si es obligatorio visitar todas las ciudades y regresar al mismo punto?. Si se utiliza un algoritmo de Fuerza Bruta que evalúa todos los posibles recorridos, la cota superior del tiempo de corrida es factorial: un orden de crecimiento mayor que el exponencial. De la misma forma, si se tiene que visitar más de 39 ciudades y el algoritmo tiene la capacidad de escanear o revisar billones de tours por segundos, le tomaría más tiempo que la vida del universo para encontrar la solución óptima[32]. A continuación se muestra una tabla con los tiempos de corrida de algoritmos de distintas clases asumiendo que la operación básica de computación tarda 1 milisegundo.

[30] PAPADIMITRIOU. Op cit,. p. 45

[31] En la década de los 50 John Von Neumann propuso un algoritmo, basado en el paradigma de diseño de algoritmos Dividir and Conquer, cuya función de tiempo en el peor caso esta acotada $O(n \log n)$

[32] LEWIS. Op cit,. p. 275-276

Tabla 19. Tabla de Complejidades para diferentes instancias de un problema.

n	$T_A(n) = n$	$T_B(n) = n \log n$	$T_C(n) = n^2$	$T_D(n) = n^3$	$T_E(n) = 2^n$
2	0.002s	0.001s	0.004s	0.008s	0.004s
4	0.004s	0.002s	0.016s	0.064s	0.016s
8	0.008s	0.007s	0.064s	0.512s	0.256
16	0.016s	0.019s	0.256s	4.096s	1.055min
32	0.032s	0.048s	1.024s	32.768s	1.65meses
64	0.064s	0.116s	4.096s	4.369min	$5{,}9*10^6$seg
128	0.128s	0.270s	16.384s	34.952min	10^{26}seg

Fuente: HORNER, Douglas. Resolução do problema das P-Medianas não capacitado. Tesis Doctoral. Facultad de Ingeniera de la Producción. p. 33 . Universidad Federal de Santa Catarina, Brasil. 2009.

En la actualidad, existen empresas cuya logística de transporte los obliga a tratar con problemas de recorridos mayores 39 destinos, luego sería imposible, desde el punto de vista práctico, implementar una solución como la descrita anteriormente. Esto indica que una cota superior polinomial para un algoritmo es una característica deseable ya que permite resolver de forma "eficiente" problemas prácticos de gran tamaño. [33]

[33]. En este punto surge la pregunta de cómo hacen las empresas para lidiar con este problema, y esta es la razón por la cual ha aumentado la investigación en método heurísticos o aproximados que permitan obtener mejores cotas y soluciones para los problemas de optimización en general

Clase P

Aunque las diferencias de orden polinomial en tiempo de corrida son importantes en el desarrollo de software e implementación de códigos, en el estudio de la complejidad es más importante determinar si un problema tiene o no propiedades polinomiales. Esta distinción es importante ya que el tiempo de corrida de un algoritmo polinomial y uno nopolinomial para instancias grandes de un problema dado puede diferir en siglos. Así, se considera que un algoritmo es eficiente si corre en tiempo polinomial y poco eficiente si corre en tiempo exponencial o mayor.

La clase de problemas P, es el conjunto de problemas (lenguajes) que se pueden resolver (decidir) en tiempo de ejecución polinomial por una máquina de Turing determinista. Cuando se analiza el carácter polinomial de un algoritmo es necesario hacer dos cosas. Primero, determinar una cota superior polinomial (usualmente en notación Big-Omicron o Big-*O*) para el número de pasos significativos que el algoritmo utiliza para procesar un conjunto de datos de tamaño n. El segundo paso consiste en evaluar si cada uno de estos pasos puede ser ejecutado en tiempos de corrida polinomial en un modelo de computación determinístico. A continuación se muestra la demonstración de membrecía del problema Reachability o Path.El problema es: Dado dos puntos s y y sobre un grafo G, existe un trayecto que conecte esos dos puntos?. Para resolver este problema se va implementar un algoritmo de búsqueda sobre grafos. Primero, se hace una descripción informal del algoritmo y luego se explica el tipo de búsqueda.

Durante el algoritmo se mantiene un conjunto S que contiene inicialmente al vértice número $S = \{s\}$. Cada vértice puede estar marcado o no-marcado. Cuando un nodo está marcado significa que perteneció en alguna iteración anterior (o actual) al conjunto S. En cada iteración se remueve (esto NO significa que se elimina el estado de marcado) un elemento del conjunto, y para este elemento se

evalúan todos los posibles vértices de conexión; si se encuentra algún vértice no marcado que esté conectado al vértice de evaluación, se marca y se añade al conjunto S. El proceso termina cuando todos los elementos del conjunto S han sido removidos; en este punto, si el vértice y esta marcado se responde "si/acepta" al problema, de lo contrario "no/rechaza".

Aunque se hizo una descripción informal de la búsqueda sobre el grafo, todavía no se ha mencionado un aspecto importante que define el tipo de búsqueda: cual es el criterio para remover un elemento del conjunto S?. Si se escoge el elemento con más tiempo (iteraciones) de estadía en el conjunto; es decir, se implementa el conjunto S como una cola (queue), el resultado es un tipo de búsqueda denominado Breath-First, y el trayecto encontrado entre los dos puntos (si existe) es el trayecto más corto o *Shortest Path*. Si se escoge el elemento más reciente de S, es decir, se implementa S como una pila (stack), el resultado es una modificación del tipo de búsqueda Depth-First. De la misma manera, si se escogen otros criterios para remover elementos de S, el resultado son diferentes tipos de búsqueda sobre grafos, lo importante es que el resultado en todos los casos es el mismo. A continuación se muestra una gráfica que ilustra cómo opera el tipo de búsqueda Breath-First.

Figura 53. Secuencia de pasos del algoritmo de búsqueda Breath-First

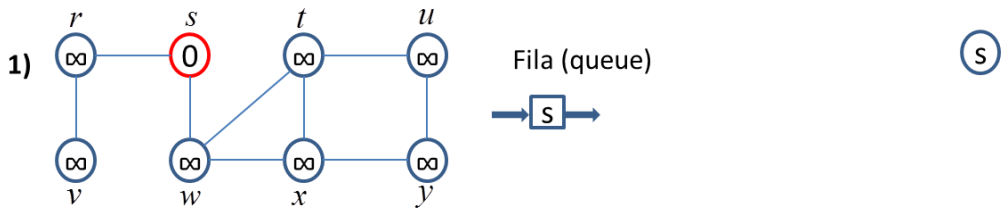


Figura 53. (Continuacion)

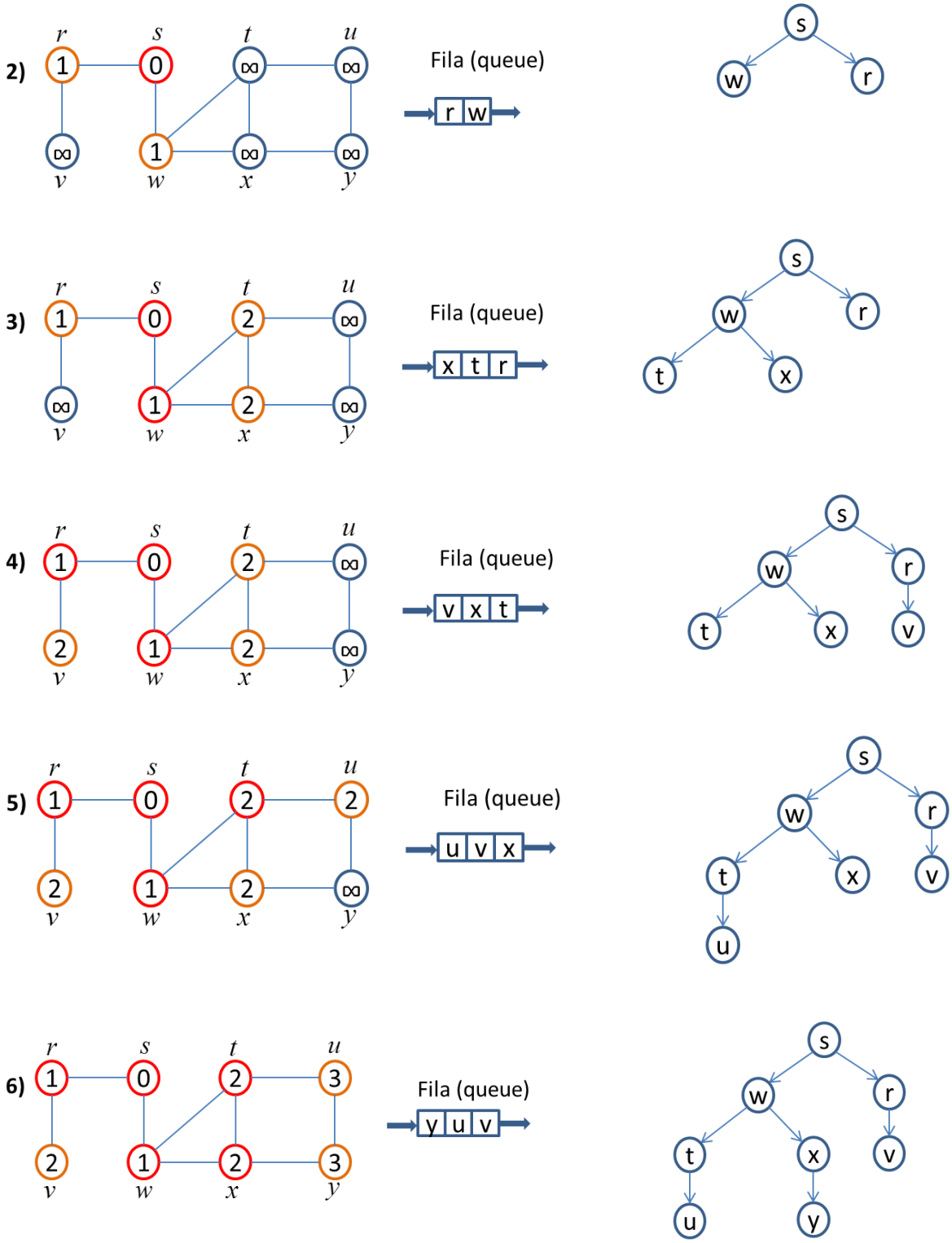
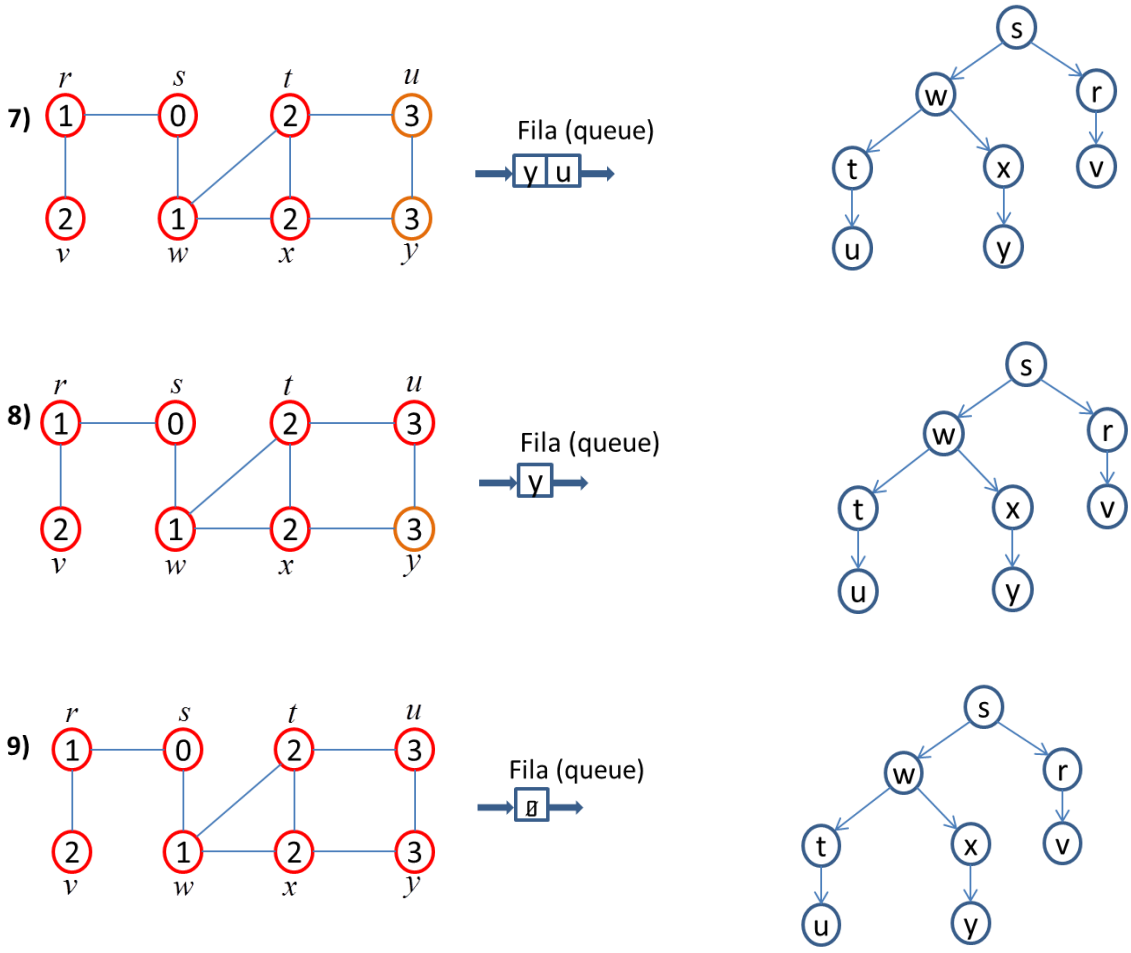


Figura 53. (Continuacion)



Fuente: CORMEN. Op cit,. p. 596

La grafica de la izquierda es el grafo sobre el cual se hace la búsqueda y la gráfica de la derecha es un árbol (estructura discreta) que representa el orden y la secuencia de búsqueda sobre el grafo. Inicialmente, el único vértice que pertenece a S (o fila) es s. El algoritmo se ubica en el reglón de s en la matriz de adyacencia (Figura 55), saca a s de la fila, evalúa todas las posibles aristas, marca los vértices que tienen coneccion directa (w y r) y añade los vértices marcados a la cola o conjunto S en el orden que indican las flechas de la figura. Según el orden de la

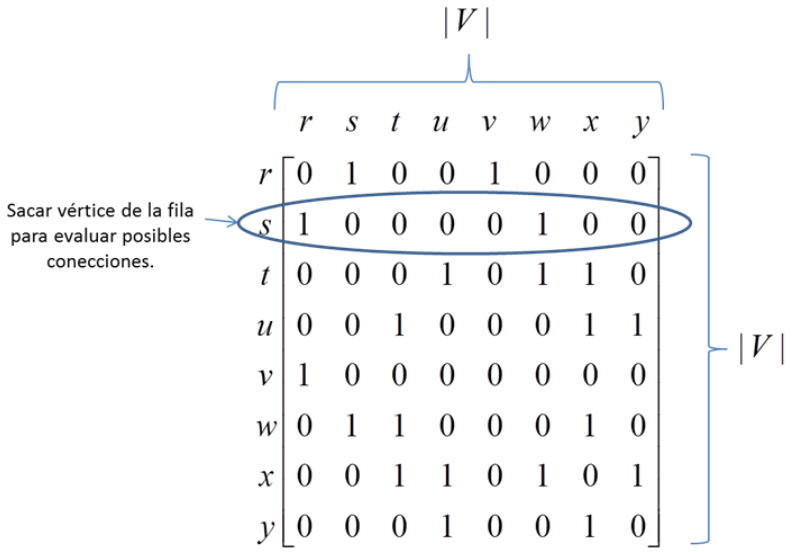
cola, el siguiente vértice para evaluar es w; se remueve de la cola y el algoritmo se ubica en el reglón respectivo de la matriz para evaluar todas las posibles aristas, luego marca los vértices con los que tiene coneccion directa (t y x) y añade los vértices marcados a la fila. El siguiente vértice para evaluación, según el orden de la fila, es r. El algoritmo se ubica en el reglón de la matriz y repite el mismo proceso que con los vértices anteriores. El algoritmo termina cuando la fila o conjunto S este vacío. A continuación se muestra un algoritmo en pseudocodigo y se demuestra que corre en tiempo lineal, así mismo se muestra la matriz de adyacencia del ejemplo anterior.

Figura 54. Pseudcodigo del Algoritmo Breath-First para resolver Reachability

```
1  procedure BFS(G, s):
2      crear fila (queue) S
3      Encolar o anhadir s a la fila S
4      marcar s
5      while S no es vacio:
6          t ← Q.dequeue() //desencolar S y asignar valor a t
7          if t == u:
8              return t
9          for all aristas e in G.AristasAdyacentes(t) do
10             //asignar una arista adyacente de t a u
11             u ← G.AristasAdyacentes(t, e)
12             if u no está marcado:
13                 marcar u
14                 encolar u en S
15     return none
```

Fuente: Autor

Figura 55. Matriz de Adyacencia para ilustrar Breath-First



Fuente: Autor

Primero se asume que las acciones de encolar, desencolar, marcar un vértice, evaluar si un vértice está marcado y evaluar si la fila esta vacía se ejecutan en tiempo constante (reglones 3,4,5,6,7,8,12,13,14 de la Figura 54). Ahora se evalúan las dos estructuras que definen la frecuencia de ejecución del algoritmo en el análisis del peor caso: *while* y *for.* La estructura *while* evalúa si la fila se encuentra vacía, si no está vacía desencola (remueve) un vértice para hacer la evaluación; debido a que este vértice no vuelve a la fila y todos los vértices tienen que pasar por la fila, el número máximo de evaluaciones que hace es el número total de vértices $|V|$. Un aspecto importante a tener en cuenta, es que el peor caso ocurre cuando no existe un trayecto, asi, el reglón 7 y 8 nunca se ejecutan en este tipo de análisis. Ahora, para cada uno de estos vértices desencolados, el algoritmo tiene que evaluar si existen connecciones adyacentes (aristas) con los demás vértices, por lo tanto, se ubica en el reglón correspondiente de la matriz de adyacencia y hace un escaneo lineal sobre las columnas para marcar los vértices adyacentes (reglones 9 – 14). La operación de cada reglón tarda $|V|$ pasos. Si son

$|V|$ pasos de desencolar y en cada paso se hace un escaneo lineal de $|V|$ pasos sobre las columnas, el número total de pasos es $|V|^2 + \text{ecuacion orden inferior}$, donde la $\text{ecuacion orden inferior}$ describe la forma en que se implementan las tareas dentro de *while* y *for.* No obstante, se quiere un análisis de algoritmo independiente de la maquina en que se ejecuta y la forma en que se codifica; por tanto, se remueve cualquier constante y ecuación de orden inferior, y se establece que una cota superior para el peor caso de reachability es $|V|^2$, o en otras palabras, Reachability es $O(|V|^2)$. Este es un certificado (no confundir con certificado de NP) de que utilizando Breath-First el peor caso entre todas las instancias de Reachability se puede resolver en tiempo polinomial por una maquina determinista que se detiene sobre cualquier entrada, y por ende Reachability pertenece a P.

Clase NP

Los algoritmos de tiempo exponencial usualmente surgen cuando se resuelven problemas utilizando métodos exhaustivos de búsqueda sobre el espacio de soluciones, estos métodos son llamados Búsqueda de Fuerza Bruta.[34] Por ejemplo, una forma de descomponer un número compuesto (no primo) en divisores no triviales, que cuando se multiplican dan el número original, es intentar con todos los divisores posibles del número. Esto resulta en un espacio de soluciones de tamaño exponencial y un algoritmo de búsqueda de esta forma resolvería el problema en tiempo exponencial. En algunos casos, los algoritmos de Fuerza Bruta pueden ser evitados cuando a través de un conocimiento especial

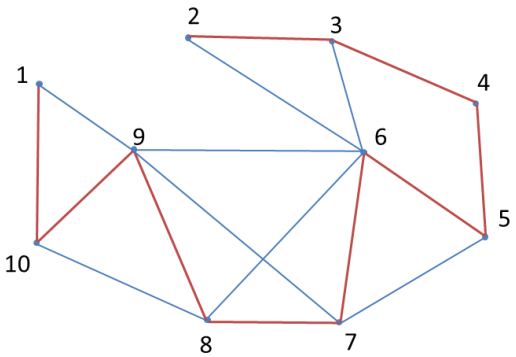
[34] SIPSER, Michael. Introduction to the theory of computation.ed.2.Thomson Course Technology. Boston, Massachussets, Estados Unidos, 2006. p. 257.

de la estructura del problema se obtiene un algoritmo que corre en tiempo polinomial para alcanzar la solución.

Si se hubiera utilizado un algoritmo de Fuerza Bruta para Reachability, es decir un algoritmo que evalúa todos los posibles trayectos entre dos puntos, el peor caso estuviera acotado por $|V|^{|V|}$. Sin embargo, existe un algoritmo inteligente como Breath-First que ejecuta la misma tarea de forma más eficiente. Desafortunadamente, no todos los problemas prácticos tiene algoritmos eficientes, y los algoritmos más "inteligentes" (es decir, cualquier algoritmo que no sea Enumeracion) diseñados para estos problemas, todavía caen en el grupo de orden exponencial.

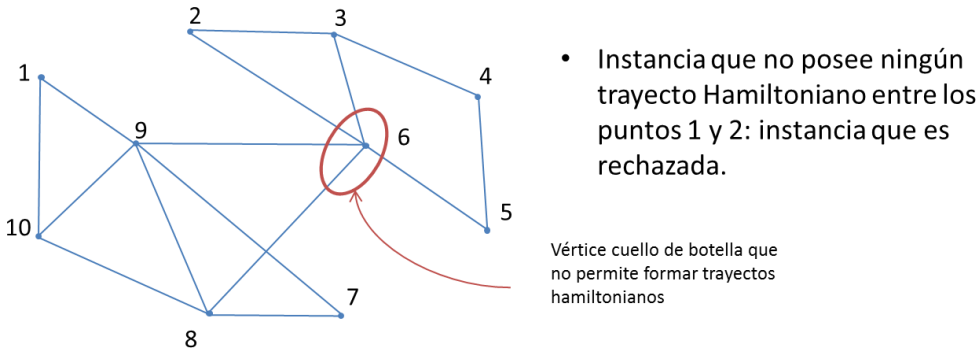
El grupo de problemas NP, aunque no se pueden decidir en tiempo polinomial (una creencia ampliamente aceptada que se deriva de la afirmación $\mathrm{P} \neq \mathrm{NP}$), poseen una característica especial: se pueden verificar en tiempo polinomial por una maquina determinista. Para describir esta propiedad vamos a utilizar como ejemplo un problema sobre grafos llamado Hamiltonian Path. El problema es: En un grafo G, dado dos vértices s y u, existe un trayecto que una los dos puntos y que pase por todos los vértices del grafo exactamente una vez?

Figura 56. Instancias distintas para un mismo tamaño del problema Hamiltonian Path



- Instancia que muestra un trayecto Hamiltoniano entre los puntos 1 y 2: instancia que es aceptada.

Figura 56. (Continuación)



Fuente: Autor

Nuevamente, no se sabe de un algoritmo que corra en tiempo polinomial para resolver el problema. No obstante, supóngase que alguien afirma que cierta instancia del problema resulta en "aceptar" (es decir, esa instancia tiene el trayecto Hamiltoniano como ilustra la Figura 56). Si se es algo escéptico, se demanda que esa persona muestre ese trayecto, y en efecto esa persona (utilizando un método no-realista que permite "adivinar" el trayecto) muestra la solución. El problema es tan grande que es imposible hacer una revisión visual, sin embargo, es posible diseñar un algoritmo determinístico que escanee el trayecto de esa instancia en particular en tiempo polinomial. Esta es la característica principal de los problemas NP: las instancias de un problema que son aceptadas, tiene al menos un certificado que puede ser verificado en tiempo polinomial por una maquina determinista, además el tamaño del certificado está acotado a lo más por una función polinomial del tamaño de la entrada (instancia). En el caso de Hamiltonian Path, el certificado polinomial de una instancia dada, es el trayecto Hamiltoniano para esa instancia.

No todos los problemas tienen este certificado polinomial; por ejemplo el complemento de Hamiltonian Path que se lee: Dado un grafo G, es cierto que NO existe un trayecto Hamiltoniano entre los vértices s y u?. Si alguien afirma y determina (de alguna manera) que cierta instancia del problema no tiene ningún trayecto Hamiltoniano, no se conoce alguna forma de verificar la no-existencia de este trayecto sin usar el mismo algoritmo exponencial que decide o resuelve este problema.

Existe otra forma de caracterizar este tipo de problemas utilizando una maquina no-determinística de Turing; razón por la cual esta clase se denomina NP (Nondetrministic Polynomial Time). Se dice que una maquina no-determinista de Turing esta acota por un polinomio si todas la ramas de computación no-deterministas terminan en un numero finito de pasos acotado por una función polinomial. Así, un problema pertenece a NP si se puede decidir con una maquina no-determinista cuya ejecución está acotada por un polinomio. La computación de una maquina no-determinista para resolver un problema NP como Hamiltonian Path procede de la siguiente manera: no-deterministicamente "adivinar" (generar infinitas ramas de computación sobre la entrada y, si existe, seleccionar el certificado NP) una permutación de los nodos del grafo; luego escanear deterministicamente la cadena correspondiente a la rama seleccionada para evaluar si cumple con las condiciones del resultado (no repeticiones de nodos, aristas entre todos los nodos, y comienzo y final en los nodos especificados). Los dos pasos se pueden ejecutar en tiempo polinomial, por lo tanto el problema es NP.[35]

Una pregunta importante respecto a esta clase de problemas es: como se sabe que no existen algoritmo polinomiales que decidan los problemas NP?. En realidad, no ha sido posible probar un solo problema que pertenezca a NP y no a P, por lo tanto, existe la posibilidad de que $\mathrm{P}=\mathrm{NP}$. Por otro lado, existen muchos

[35]SIPSER. Op cit,. p. 266

problemas en NP como Hamiltonian Path para los cuales, después de años de investigación, no ha sido posible encontrar algoritmos polinomiales; situación que refuerza la creencia que $P \neq NP$ .Aunque no ha sido posible probar ninguna de las dos afirmaciones, los expertos concuerdan en que es mejor actuar bajo la asunción de $P \neq NP$ y de esta manera buscar métodos alternativos que permitan tratar problemas prácticos de gran importancia, en lugar de gastar años intentando probar lo contrario.[36]

NP-Complete

En 1970 Stephen Cook y Leonid Vivid hicieron avances respecto a la pregunta $P = NP$, y descubrieron un grupo especial de problemas que está relacionado a la complejidad de toda la clase NP . Este grupo especial de problemas se llama $NP\text{-complete}$

Desde un punto de vista teórico, si alguien intenta demonstrar igualdad entre P y NP lo único que tiene que hacer es encontrar un algoritmo polinomial para un problema de este grupo; de igual manera, si alguien intenta probar lo contrario puede concentrar sus esfuerzos en acotar alguno de estos problemas.

Desde el punto de vista práctico, si se demuestra que algún problema pertenece a $NP\text{-complete}$, entonces existen altas probabilidades de que este problema no posea un algoritmo polinomial ya que son los problemas más difíciles de toda la clase y por lo tanto se puede invertir tiempo en otras formas de abordar el problema.

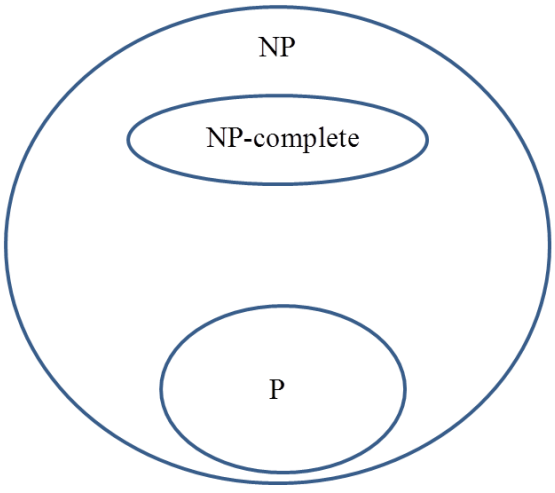
Como se establece que un problema pertenece a $NP\text{-complete}$?. Una forma es satisfacer las siguientes dos condiciones: 1) demostrar que el problema pertenece a NP y 2) demostrar que todo problema en NP se puede reducir polinomialmente al problema en cuestión. Una forma más sencilla es, además de demonstrar que

[36] GAREY, Michael R. y JHONSON, David S..Computers and Intractability: A guide to the Theory of NP-completeness. W.H. Freeman. San Francisco, Estados Unidos, 1979. p. 33

pertenece a NP, encontrar una reducción polinomial de un problema $\mathrm{NP\text{-}complete}$ al problema que se quiere evaluar membrecía ,sin embargo, se tiene que conocer al menos un problema que pertenezca a este grupo.

Una reducción polinomial es una transformación o función de mapeo que se ejecuta en tiempo polinomial o de manera "eficiente". Se dice que un problema A se reduce polinomialmente a un problema B, si para cada entrada $w \in A$ existe una entrada equivalente $f(w) \in B$, es decir, la entrada w es "aceptada" si y solo si la entrada $f(w)$ es "aceptada" (Figura 58). Cuando se tiene una reducción polinomial de A a B, es posible adaptar un algoritmo polinomial de B (si existe) para obtener un algoritmo polinomial que decida A; en otras palabras, la existencia de una reducción polinomial evidencia que B es al menos tan difícil de resolver que A: "Si B se puede resolver eficientemente, entonces A se puede resolver eficientemente" o de forma equivalente "Si A requiere tiempo exponencial, entonces B también requiere tiempo exponencial".[37] A continuación se muestra un representación gráfica de cómo se piensa que es la estructura de NP sí $\mathrm{NP} \neq \mathrm{P}$.

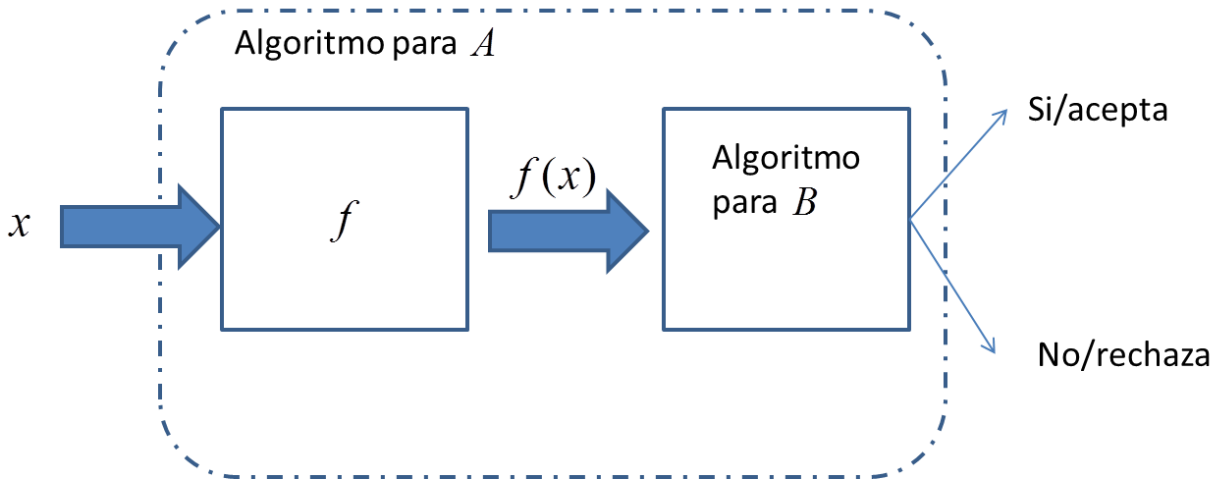
Figura 57. Representación Grafica de la estructura de NP



Fuente: GAREY. Op cit,. p. 37

[37] LEWIS. Op cit,. 301-302

Figura 58. Representación Gráfica de Reducción Polinomial de A a B.



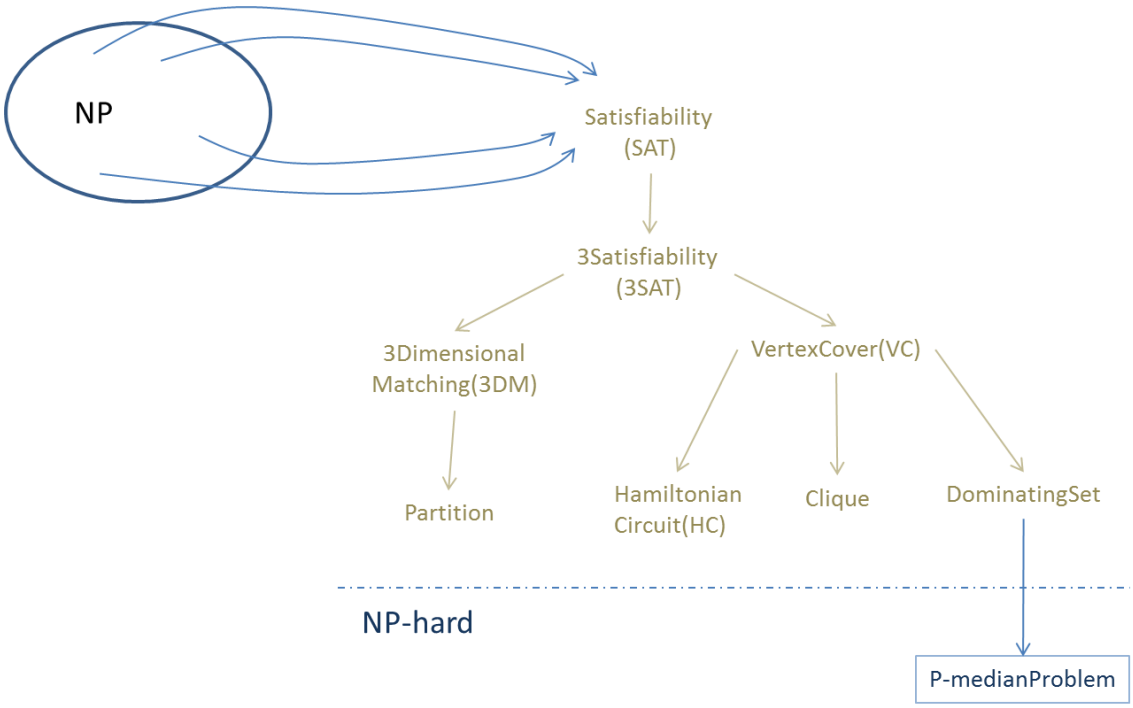
Fuente: PAPADIMITRIOU. Op cit,. p. 160

En la década de los 70, Cook demostró que cualquier problema en NP se puede reducir polinomialmente al problema de determinar si una expresión lógica de primer orden se puede satisfacer con al menos una asignación de sus variables (o Satisfiable Assignment Problem), que también es NP. Por lo tanto, SAT es el primer problema que fue clasificado en este grupo. Partiendo de este problema, se pudo demonstrar que Hamiltonian Path y otros problemas prácticos de gran importancia también pertenecen a esta clase especial (para esas demonstraciones se hace una reducción polinomial de SAT al problema en cuestión)[38]. A continuación se muestra un posible mapa de reducciones de problemas desde SAT hasta el PMP. Kariv y Hakimi (1979) demostraron que el problema de encontrar un subconjunto V_p de p vértices tal que cada vértice del grafo pertenece a V_p o es adyacente a V_p (DominatingSet Problem) , es

[38]KARP, Richard M. (1972). "Reducibility Among Combinatorial Problems". In R. E. Miller and J. W. Thatcher (editors).*Complexity of Computer Computations*.New York: Plenum. pp. 85–103.

polinomialmente reducible a encontrar una P-mediana en el grafo, incluso cuando el grafo tiene una estructura simple (grafo plano con vértices de grado máximo igual a 3).

Figura 59. Mapa de Reducciones Polinomiales desde SAT hasta el PMP



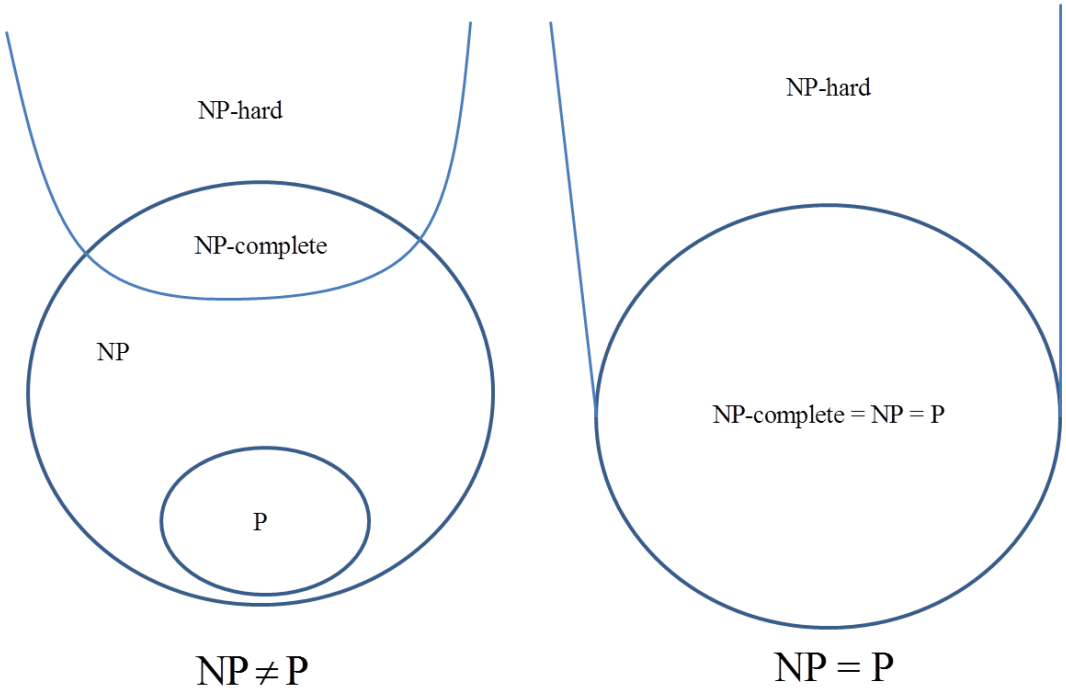
Fuente: Autor

NP-Hard

Algunos problemas son tan difíciles de resolver, que aunque se puede probar la condición 2 de NP-complete (cada problema en NP es reducible al problema en cuestión), no es posible probar la condición 1: el problema es NP. Si ocurre esta situación, se dice que el problema es NP-hard. Una prueba de que un problema en particular es NP-hard, indica altas probabilidades de que el problema requiere de un algoritmo exponencial, sin embargo, la prueba de que ese problema no está

en NP, no dice nada acerca de la dificultad de los problemas $NP\text{-complete}$, ya puede ocurrir que el problema requiere de un algoritmo exponencial incluso sí $P = NP$. Una aplicación del concepto de $NP\text{-Hard}$ son los complementos de los problemas NP, los cuales requieren de tiempo exponencial ya sea para decidir o verificar el problema. [39]Otra aplicación de $NP\text{-hard}$ se ve en la relación entre la versión de optimización y la versión de decisión de un problema $NP\text{-complete}$ (recordar que la teoría de $NP\text{-complete}$ solo aplica directamente a problemas de decisión), se dice que si la versión de decisión de un problema es $NP\text{-complete}$ entonces la versión de optimización es $NP\text{-hard}$ .A continuación se muestra la estructura de las clases mencionadas bajo la suposición que $P = NP$ y $P \neq NP$.

Figura 60. Representacion grafica de la estructura de NP y $NP\text{-hard}$



Fuente: NP-Hard [Anonimo][On line]. Wikipedia. The free encyclopedia. [Citado el 20 de marzo de 2013].Disponible en: http://en.wikipedia.org/wiki/NP-hard

[39] HOPCROFT. Op cit,. p. 422-423

Apéndice D. Resultados Computacionales

A continuación se presentan los resultados arrojados por el algoritmo en la solución de los 40 problemas de la librería OR-Library.

1.1. PROBLEMAS DE 100 NODOS

Vértices	100
Medianas	5
Valor Optimo	5819

Resultados promedio de dos corridas para el algoritmo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	5819	5819	5819
10	5819	5819	5819
15	5819	5819	5819
20	5819	5819	5819

Porcentaje de diferencia con valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	100
Medianas	10
Valor Optimo	4093

Resultados promedio de dos corridas para el algoritmo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	4093	4093	4093
10	4093	4093	4093
15	4093	4093	4093
20	4093	4093	4093

Porcentaje de diferencia con valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	100
Medianas	10
Valor Optimo	4250

Resultados promedio de dos corridas para el algoritmo			
Iteraciones	Tamaño del Enjambre		
	3	6	9

Porcentaje de diferencia con valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9

5	4250	4250	4250
10	4250	4250	4250
15	4250	4250	4250
20	4250	4250	4250

5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	100
Medianas	20
Valor Optimo	3034

Resultados promedio de dos corridas para el algoritmo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	3034	3034	3034
10	3034	3034	3034
15	3034	3034	3034
20	3034	3034	3034

Porcentaje de diferencia con valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	100
Medianas	33
Valor Optimo	1355

Resultados promedio de dos corridas para el algoritmo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	1355	1355	1355
10	1355	1355	1355
15	1355	1355	1355
20	1355	1355	1355

Porcentaje de diferencia con valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

1.2. PROBLEMAS DE 200 NODOS

Vértices	200
Medianas	5
Valor Optimo	7824

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	7824	7824	7824

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0

10	7824	7824	7824
15	7824	7824	7824
20	7824	7824	7824

10	0	0	0
15	0	0	0
20	0	0	0

Vértices	200
Medianas	10
Valor Optimo	5631

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	5631	5631	5631
10	5631	5631	5631
15	5631	5631	5631
20	5631	5631	5631

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	200
Medianas	20
Valor Optimo	4445

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	4445	4445	4445
10	4445	4445	4445
15	4445	4445	4445
20	4445	4445	4445

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	200
Medianas	40
Valor Optimo	2734

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	2734	2734	2734
10	2734	2734	2734
15	2734	2734	2734
20	2734	2734	2734

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	200
Medianas	67
Valor Optimo	1255

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	1256	1255	1255
10	1255	1255	1255
15	1255	1255	1255
20	1255	1255	1255

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.000797	0	0
10	0	0	0
15	0	0	0
20	0	0	0

1.3. PROBLEMAS DE 300 NODOS

Vértices	300
Medianas	5
Valor Optimo	7696

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	7696	7696	7696
10	7696	7696	7696
15	7696	7696	7696
20	7696	7696	7696

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	300
Medianas	10
Valor Optimo	6634

Promedio de las dos corridas para cada combinación

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo

Iteraciones	Tamaño del Enjambre		
	3	6	9
5	6634	6634	6634
10	6634	6634	6634
15	6634	6634	6634
20	6634	6634	6634

Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	300
Medianas	30
Valor Optimo	4374

Promedio de las dos corridas para cada combinación

Iteraciones	Tamaño del Enjambre		
	3	6	9
5	4374	4374	4374
10	4374	4374	4374
15	4374	4374	4374
20	4374	4374	4374

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo

Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	300
Medianas	60
Valor Optimo	2968

Promedio de las dos corridas para cada combinación

Iteraciones	Tamaño del Enjambre		
	3	6	9
5	2968	2968	2968
10	2968	2969	2968
15	2968	2968	2968
20	2968	2968	2968

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo

Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0.000168	0
15	0	0	0
20	0	0	0

Vértices	300
Medianas	100
Valor Optimo	1729

Promedio de las dos corridas para cada combinación

Iteraciones	Tamaño del Enjambre		
	3	6	9
5	1734	1731	1731

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo

Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.002892	0.001157	0.001157

10	1732	1731	1730
15	1731	1730	1731
20	1730	1731	1731

10	0.001735	0.000868	0.000578
15	0.001157	0.000578	0.000868
20	0.000578	0.001157	0.000868

1.4. PROBLEMAS DE 400 NODOS

Vértices	400
Medianas	5
Valor Optimo	8162

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	8162	8162	8162
10	8162	8162	8162
15	8162	8162	8162
20	8162	8162	8162

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	400
Medianas	10
Valor Optimo	6999

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	6999	6999	6999
10	6999	6999	6999
15	6999	6999	6999
20	6999	6999	6999

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	400
Medianas	40
Valor Optimo	4809

Promedio de las dos corridas para cada combinación	
Iteraciones	Tamaño del Enjambre

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo	
Iteraciones	Tamaño del Enjambre

	3	6	9
5	4809	4809	4809
10	4809	4809	4809
15	4809	4809	4809
20	4809	4809	4809

	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	400
Medianas	80
Valor Optimo	2845

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	2849	2847	2846
10	2846	2846	2847
15	2846	2846	2845
20	2846	2845	2845

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.00123	0.000703	0.000351
10	0.000351	0.000176	0.000527
15	0.000176	0.000351	0
20	0.000351	0	0

Vértices	400
Medianas	133
Valor Optimo	1789

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	1791	1791	1789
10	1790	1790	1790
15	1790	1789	1789
20	1790	1789	1789

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.001118	0.001118	0
10	0.000559	0.000279	0.000279
15	0.000279	0	0
20	0.000559	0	0

1.5. PROBLEMAS DE 500 NODOS

Vértices	500
Medianas	5
Valor Optimo	9138

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	9138	9138	9138
10	9138	9138	9138
15	9138	9138	9138
20	9138	9138	9138

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	500
Medianas	10
Valor Optimo	8579

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	8579	8579	8579
10	8579	8579	8579
15	8579	8579	8579
20	8579	8579	8579

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	500
Medianas	50
Valor Optimo	4619

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	4619	4619	4619
10	4619	4619	4619
15	4619	4619	4619
20	4619	4619	4619

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	500

Medianas	100
Valor Optimo	2961

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	2965	2963	2961
10	2966	2962	2962
15	2963	2961	2963
20	2961	2961	2961

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.001182	0.000675	0
10	0.00152	0.000338	0.000169
15	0.000675	0	0.000507
20	0	0	0

Vértices	500
Medianas	167
Valor Optimo	1828

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	1834	1833	1833
10	1833	1832	1834
15	1834	1832	1831
20	1833	1833	1831

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.003009	0.002462	0.002462
10	0.002735	0.002188	0.003009
15	0.003009	0.002188	0.001368
20	0.002735	0.002462	0.001368

1.6. PROBLEMAS DE 600 NODOS

Vértices	600
Medianas	5
Valor Optimo	9917

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	9917	9917	9917
10	9917	9917	9917
15	9917	9917	9917
20	9917	9917	9917

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	600
Medianas	10
Valor Optimo	8307

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	8307	8307	8307
10	8307	8307	8307
15	8307	8307	8307
20	8307	8307	8307

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	600
Medianas	60
Valor Optimo	4498

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	4499	4499	4498
10	4499	4498	4499
15	4498	4499	4499
20	4498	4498	4498

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.000222	0.000222	0
10	0.000111	0	0.000111
15	0	0.000111	0.000111
20	0	0	0

Vértices	600
Medianas	120
Valor Optimo	3033

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	3034	3035	3035
10	3036	3035	3037
15	3034	3036	3035
20	3034	3035	3034

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.00033	0.000495	0.000495
10	0.000989	0.000659	0.001154
15	0.00033	0.000989	0.000495
20	0.000165	0.000495	0.000165

Vértices	600
Medianas	200
Valor Optimo	1989

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	1995	1998	1998
10	2001	1998	1996
15	2000	1998	1997
20	1999	1998	1998

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.003017	0.004525	0.004274
10	0.005782	0.004274	0.003268
15	0.005279	0.004525	0.004022
20	0.004776	0.004525	0.004525

1.7. PROBLEMAS DE 700 NODOS

Vértices	700
Medianas	5
Valor Optimo	10086

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	10086	10086	10086
10	10086	10086	10086
15	10086	10086	10086
20	10086	10086	10086

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	700
Medianas	10
Valor Optimo	9297

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	9297	9297	9297
10	9297	9297	9297
15	9297	9297	9297
20	9297	9297	9297

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	700
Medianas	70
Valor Optimo	4700

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	4703	4702	4700
10	4703	4700	4700
15	4702	4702	4700
20	4700	4700	4700

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.000638	0.000319	0
10	0.000638	0	0
15	0.000319	0.000319	0
20	0	0	0

Vértices	700
Medianas	140
Valor Optimo	3013

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	3016	3017	3015
10	3016	3015	3016
15	3016	3014	3014
20	3016	3015	3015

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.000996	0.001162	0.000664
10	0.000996	0.000664	0.000996
15	0.00083	0.000166	0.000332
20	0.000996	0.000664	0.000498

1.8. PROBLEMAS DE 800 NODOS

Vértices	800
Medianas	5
Valor Optimo	10400

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	10400	10400	10400
10	10400	10400	10400
15	10400	10400	10400
20	10400	10400	10400

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	800
Medianas	10
Valor Optimo	9934

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	9934	9934	9934
10	9934	9934	9934
15	9934	9934	9934
20	9934	9934	9934

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	800
Medianas	80
Valor Optimo	5057

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	5060	5058	5058
10	5059	5058	5058
15	5058	5057	5057
20	5058	5058	5058

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.000593	0.000198	9.89E-05
10	0.000395	9.89E-05	0.000198
15	9.89E-05	0	0
20	9.89E-05	9.89E-05	0.000198

1.9. PROBLEMAS DE 900 NODOS

Vértices	900
Medianas	5
Valor Optimo	11060

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	11060	11060	11060
10	11060	11060	11060
15	11060	11060	11060
20	11060	11060	11060

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	900
Medianas	10
Valor Optimo	9423

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	9423	9423	9423
10	9423	9423	9423
15	9423	9423	9423
20	9423	9423	9423

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0	0	0
10	0	0	0
15	0	0	0
20	0	0	0

Vértices	900
Medianas	90
Valor Optimo	5128

Promedio de las dos corridas para cada combinación			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	5133	5134	5132
10	5134	5131	5131
15	5132	5130	5130
20	5131	5131	5129

Desviación respecto al valor optimo = (Heurística - Valor Optimo)/Valor Optimo			
Iteraciones	Tamaño del Enjambre		
	3	6	9
5	0.000975	0.001073	0.000683
10	0.001073	0.000488	0.000488
15	0.000683	0.00039	0.000293
20	0.000488	0.000585	0.000195

2. Tabla de Datos utilizado para el Análisis de Varianza (ANOVA)

A continuación se muestra los datos de la métrica creada con los cuatro parámetros descritos en el documento principal para cada uno de los cuatro problemas analizados.

2.1. Problema pmed33

pmed33	Variable de Respuesta (Desviación del Optimo)				Valor QPSO			
Valor Optimo:	Replicas			Medias	Replicas			Medias
4700	1	2	3		1	2	3	
5 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00085	0.00128	0	0.00213	4704	4706	4700	4703.33
alto/bajo/alto	0.00021	0.00043	0.00021	0.00085	4701	4702	4701	4701.33
alto/bajo/bajo	0	0.00043	0.00043	0.00085	4700	4702	4702	4701.33
bajo/alto/alto	0.00085	0.00085	0.00021	0.00191	4704	4704	4701	4703
bajo/alto/bajo	0.00085	0.00021	0	0.00106	4704	4701	4700	4701.67
bajo/bajo/alto	0	0	0.00021	0.00021	4700	4700	4701	4700.33
0.5-0.5								
alto/alto/bajo	0.00021	0	0.00064	0.00085	4701	4700	4703	4701.33
alto/bajo/alto	0.00106	0.00043	0.00064	0.00213	4705	4702	4703	4703.33
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0.00043	0.00021	0.00106	0.0017	4702	4701	4705	4702.67
bajo/alto/bajo	0.00064	0	0	0.00064	4703	4700	4700	4701
bajo/bajo/alto	0	0	0.00021	0.00021	4700	4700	4701	4700.33
0.75-0.25								
alto/alto/bajo	0.00064	0.00085	0	0.00149	4703	4704	4700	4702.33
alto/bajo/alto	0.00064	0.00106	0.00021	0.00191	4703	4705	4701	4703
alto/bajo/bajo	0.00085	0	0	0.00085	4704	4700	4700	4701.33
bajo/alto/alto	0.00043	0.00106	0.00085	0.00234	4702	4705	4704	4703.67
bajo/alto/bajo	0.00064	0.00043	0.00106	0.00213	4703	4702	4705	4703.33
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
6 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0.00021	0	0.00021	4700	4701	4700	4700.33
alto/bajo/bajo	0.00043	0.00021	0.00021	0.00085	4702	4701	4701	4701.33
bajo/alto/alto	0.00043	0	0	0.00043	4702	4700	4700	4700.67
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0.00021	0	0.00021	0.00043	4701	4700	4701	4700.67
0.5-0.5								
alto/alto/bajo	0.00064	0	0	0.00064	4703	4700	4700	4701
alto/bajo/alto	0	0.00021	0	0.00021	4700	4701	4700	4700.33
alto/bajo/bajo	0.00043	0.00064	0	0.00106	4702	4703	4700	4701.67
bajo/alto/alto	0	0.00021	0	0.00021	4700	4701	4700	4700.33
bajo/alto/bajo	0.00021	0	0.00043	0.00064	4701	4700	4702	4701
bajo/bajo/alto	0	0	0.00043	0.00043	4700	4700	4702	4700.67
0.75-0.25								
alto/alto/bajo	0	0.00021	0.00064	0.00085	4700	4701	4703	4701.33
alto/bajo/alto	0.00043	0.00043	0.00021	0.00106	4702	4702	4701	4701.67
alto/bajo/bajo	0.00064	0	0	0.00064	4703	4700	4700	4701
bajo/alto/alto	0.00085	0	0	0.00085	4704	4700	4700	4701.33
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0.00043	0	0.00043	0.00085	4702	4700	4702	4701.33

9 partículas								
0.25-0.75								
alto/alto/bajo	0.00043	0.00021	0	0.00064	4702	4701	4700	4701
alto/bajo/alto	0	0.00021	0	0.00021	4700	4701	4700	4700.33
alto/bajo/bajo	0	0.00021	0	0.00021	4700	4701	4700	4700.33
bajo/alto/alto	0.00043	0	0	0.00043	4702	4700	4700	4700.67
bajo/alto/bajo	0	0.00021	0	0.00021	4700	4701	4700	4700.33
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
0.5-0.5								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0.00021	0	0.00021	4700	4701	4700	4700.33
alto/bajo/bajo	0	0.00021	0.00021	0.00043	4700	4701	4701	4700.67
bajo/alto/alto	0	0	0.00021	0.00021	4700	4700	4701	4700.33
bajo/alto/bajo	0	0.00064	0.00021	0.00085	4700	4703	4701	4701.33
bajo/bajo/alto	0.00021	0	0.00021	0.00043	4701	4700	4701	4700.67
0.75-0.25								
alto/alto/bajo	0.00021	0.00021	0	0.00043	4701	4701	4700	4700.67
alto/bajo/alto	0	0	0.00043	0.00043	4700	4700	4702	4700.67
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0.00021	0.00021	4700	4700	4701	4700.33
bajo/alto/bajo	0	0	0.00021	0.00021	4700	4700	4701	4700.33
bajo/bajo/alto	0	0.00043	0.00021	0.00064	4700	4702	4701	4701
10 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00085	0	0.00021	0.00106	4704	4700	4701	4701.67
alto/bajo/alto	0	0	0.00021	0.00021	4700	4700	4701	4700.33
alto/bajo/bajo	0.00064	0	0.00085	0.00149	4703	4700	4704	4702.33
bajo/alto/alto	0	0.00043	0.00021	0.00064	4700	4702	4701	4701
bajo/alto/bajo	0.00043	0.00043	0	0.00085	4702	4702	4700	4701.33
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
0.5-0.5								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0.00043	0	0.00043	0.00085	4702	4700	4702	4701.33
alto/bajo/bajo	0	0.00021	0	0.00021	4700	4701	4700	4700.33
bajo/alto/alto	0	0.00064	0.00021	0.00085	4700	4703	4701	4701.33
bajo/alto/bajo	0	0.00106	0	0.00106	4700	4705	4700	4701.67
bajo/bajo/alto	0.00021	0	0	0.00021	4701	4700	4700	4700.33
0.75-0.25								
alto/alto/bajo	0.00043	0	0	0.00043	4702	4700	4700	4700.67
alto/bajo/alto	0.00021	0.00021	0.00043	0.00085	4701	4701	4702	4701.33
alto/bajo/bajo	0.00021	0	0.00085	0.00106	4701	4700	4704	4701.67
bajo/alto/alto	0.00043	0.00021	0.00043	0.00106	4702	4701	4702	4701.67
bajo/alto/bajo	0.00106	0	0.00064	0.0017	4705	4700	4703	4702.67
bajo/bajo/alto	0	0.00064	0.00021	0.00085	4700	4703	4701	4701.33
6 partículas								
0.25-0.75								
alto/alto/bajo	0	0.00043	0.00043	0.00085	4700	4702	4702	4701.33
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0.00021	0.00021	4700	4700	4701	4700.33
bajo/alto/alto	0	0.00021	0.00043	0.00064	4700	4701	4702	4701
bajo/alto/bajo	0	0	0.00043	0.00043	4700	4700	4702	4700.67
bajo/bajo/alto	0	0.00021	0.00021	0.00043	4700	4701	4701	4700.67

0.5-0.5								
alto/alto/bajo	0	0	0.00021	0.00021	4700	4700	4701	4700.33
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0.00085	0.00021	0.00106	4700	4704	4701	4701.67
bajo/alto/bajo	0	0	0.00043	0.00043	4700	4700	4702	4700.67
bajo/bajo/alto	0	0.00021	0.00043	0.00064	4700	4701	4702	4701
0.75-0.25								
alto/alto/bajo	0	0.00021	0.00043	0.00064	4700	4701	4702	4701
alto/bajo/alto	0.00021	0.00021	0.00021	0.00064	4701	4701	4701	4701
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0.00021	0.00021	4700	4700	4701	4700.33
bajo/bajo/alto	0	0.00043	0	0.00043	4700	4702	4700	4700.67
9 partículas								
0.25-0.75								
alto/alto/bajo	0.00021	0	0	0.00021	4701	4700	4700	4700.33
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
0.5-0.5								
alto/alto/bajo	0	0.00021	0.00043	0.00064	4700	4701	4702	4701
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0.00043	0.00043	4700	4700	4702	4700.67
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
0.75-0.25								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0.00021	0.00021	4700	4700	4701	4700.33
alto/bajo/bajo	0.00021	0	0	0.00021	4701	4700	4700	4700.33
bajo/alto/alto	0.00021	0	0	0.00021	4701	4700	4700	4700.33
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0	0.00043	0.00043	4700	4700	4702	4700.67
15 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0.00021	0.00021	4700	4700	4701	4700.33
alto/bajo/alto	0.00021	0	0.00021	0.00043	4701	4700	4701	4700.67
alto/bajo/bajo	0.00021	0.00021	0.00021	0.00064	4701	4701	4701	4701
bajo/alto/alto	0.00085	0.00021	0.00043	0.00149	4704	4701	4702	4702.33
bajo/alto/bajo	0	0.00064	0	0.00064	4700	4703	4700	4701
bajo/bajo/alto	0.00064	0	0	0.00064	4703	4700	4700	4701
0.5-0.5								
alto/alto/bajo	0	0.00021	0	0.00021	4700	4701	4700	4700.33
alto/bajo/alto	0.00043	0.00021	0	0.00064	4702	4701	4700	4701
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0.00021	0.00106	0.00128	4700	4701	4705	4702
bajo/bajo/alto	0.00021	0.00085	0	0.00106	4701	4704	4700	4701.67

0.75-0.25								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0.00064	0.00064	4700	4700	4703	4701
alto/bajo/bajo	0.00064	0	0.00043	0.00106	4703	4700	4702	4701.67
bajo/alto/alto	0.00021	0	0.00021	0.00043	4701	4700	4701	4700.67
bajo/alto/bajo	0.00021	0	0.00021	0.00043	4701	4700	4701	4700.67
bajo/bajo/alto	0	0.00043	0.00064	0.00106	4700	4702	4703	4701.67
6 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0.00021	0.00021	4700	4700	4701	4700.33
bajo/alto/alto	0	0	0.00043	0.00043	4700	4700	4702	4700.67
bajo/alto/bajo	0.00021	0	0	0.00021	4701	4700	4700	4700.33
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
0.5-0.5								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0.00021	0	0	0.00021	4701	4700	4700	4700.33
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0.00043	0	0.00043	4700	4702	4700	4700.67
0.75-0.25								
alto/alto/bajo	0.00021	0	0	0.00021	4701	4700	4700	4700.33
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0.00021	0	0.00021	0.00043	4701	4700	4701	4700.67
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0.00021	0.00021	4700	4700	4701	4700.33
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
9 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0.00021	0	0	0.00021	4701	4700	4700	4700.33
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0.00021	0	0.00021	4700	4701	4700	4700.33
0.5-0.5								
alto/alto/bajo	0.00021	0.00021	0	0.00043	4701	4701	4700	4700.67
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0.00021	0	0	0.00021	4701	4700	4700	4700.33
0.75-0.25								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0.00021	0	0	0.00021	4701	4700	4700	4700.33

20 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0	0.00043	0	0.00043	4700	4702	4700	4700.67
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0	0.00021	0.00021	4700	4700	4701	4700.33
0.5-0.5								
alto/alto/bajo	0	0	0.00021	0.00021	4700	4700	4701	4700.33
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0.00064	0.00021	0	0.00085	4703	4701	4700	4701.33
bajo/alto/alto	0.00021	0	0.00021	0.00043	4701	4700	4701	4700.67
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0.00021	0.00021	0	0.00043	4701	4701	4700	4700.67
0.75-0.25								
alto/alto/bajo	0.00085	0	0	0.00085	4704	4700	4700	4701.33
alto/bajo/alto	0	0.00043	0	0.00043	4700	4702	4700	4700.67
alto/bajo/bajo	0.00021	0.00021	0.00021	0.00064	4701	4701	4701	4701
bajo/alto/alto	0	0.00021	0.00021	0.00043	4700	4701	4701	4700.67
bajo/alto/bajo	0.00064	0	0	0.00064	4703	4700	4700	4701
bajo/bajo/alto	0	0.00021	0	0.00021	4700	4701	4700	4700.33
6 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0.00021	0.00021	0.00043	4700	4701	4701	4700.67
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
0.5-0.5								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0.00021	0	0	0.00021	4701	4700	4700	4700.33
0.75-0.25								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0.00043	0	0.00043	4700	4702	4700	4700.67
bajo/alto/alto	0	0	0.00021	0.00021	4700	4700	4701	4700.33
bajo/alto/bajo	0.00021	0.00021	0	0.00043	4701	4701	4700	4700.67
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
9 partículas								
0.25-0.75								
alto/alto/bajo	0.00021	0	0	0.00021	4701	4700	4700	4700.33
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700

0.5-0.5								
alto/alto/bajo	0	0.00021	0	0.00021	4700	4701	4700	4700.33
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0	0	0	0	4700	4700	4700	4700
bajo/alto/alto	0	0	0	0	4700	4700	4700	4700
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700
0.75-0.25								
alto/alto/bajo	0	0	0	0	4700	4700	4700	4700
alto/bajo/alto	0	0	0	0	4700	4700	4700	4700
alto/bajo/bajo	0.00021	0	0	0.00021	4701	4700	4700	4700.33
bajo/alto/alto	0.00021	0	0	0.00021	4701	4700	4700	4700.33
bajo/alto/bajo	0	0	0	0	4700	4700	4700	4700
bajo/bajo/alto	0	0	0	0	4700	4700	4700	4700

2.2. Problema pmed34

pmed34	Variable de Respuesta (Desviación del Optimo)				Valor QPSO			
Valor Optimo:	Replicas			Medias	Replicas			Medias
3013	1	2	3		1	2	3	
5 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00066	0.00133	0.00033	0.00077	3015	3017	3014	3015.3
alto/bajo/alto	0.001	0.00066	0.00066	0.00077	3016	3015	3015	3015.3
alto/bajo/bajo	0.00166	0.00166	0.00033	0.00122	3018	3018	3014	3016.7
bajo/alto/alto	0.00133	0.00232	0.001	0.00155	3017	3020	3016	3017.7
bajo/alto/bajo	0.00066	0.00199	0.001	0.00122	3015	3019	3016	3016.7
bajo/bajo/alto	0.00166	0.00266	0.001	0.00177	3018	3021	3016	3018.3
0.5-0.5								
alto/alto/bajo	0.00166	0.001	0.00133	0.00133	3018	3016	3017	3017.0
alto/bajo/alto	0.00066	0.00266	0.00166	0.00166	3015	3021	3018	3018.0
alto/bajo/bajo	0.001	0.00199	0.001	0.00133	3016	3019	3016	3017.0
bajo/alto/alto	0.00166	0.001	0.00232	0.00166	3018	3016	3020	3018.0
bajo/alto/bajo	0.001	0.00266	0.00232	0.00199	3016	3021	3020	3019.0
bajo/bajo/alto	0.00166	0.00066	0.00066	0.001	3018	3015	3015	3016.0
0.75-0.25								
alto/alto/bajo	0.00066	0.00232	0.00066	0.00122	3015	3020	3015	3016.7
alto/bajo/alto	0.00232	0.00199	0.00266	0.00232	3020	3019	3021	3020.0
alto/bajo/bajo	0.00066	0.00133	0.00199	0.00133	3015	3017	3019	3017.0
bajo/alto/alto	0.00199	0.001	0.00033	0.00111	3019	3016	3014	3016.3
bajo/alto/bajo	0.00066	0.00166	0.00166	0.00133	3015	3018	3018	3017.0
bajo/bajo/alto	0.00166	0.00232	0.00232	0.0021	3018	3020	3020	3019.3
6 partículas								
0.25-0.75								
alto/alto/bajo	0.001	0.00133	0.00199	0.00144	3016	3017	3019	3017.3
alto/bajo/alto	0.00166	0.001	0.00066	0.00111	3018	3016	3015	3016.3
alto/bajo/bajo	0.00066	0.00133	0.001	0.001	3015	3017	3016	3016.0
bajo/alto/alto	0.00133	0	0.00133	0.00089	3017	3013	3017	3015.7
bajo/alto/bajo	0.001	0.00166	0.00066	0.00111	3016	3018	3015	3016.3
bajo/bajo/alto	0.00066	0.00066	0.00133	0.00089	3015	3015	3017	3015.7

0.5-0.5								
alto/alto/bajo	0.00066	0.00133	0.001	0.001	3015	3017	3016	3016.0
alto/bajo/alto	0.00066	0.00133	0.001	0.001	3015	3017	3016	3016.0
alto/bajo/bajo	0.001	0	0.001	0.00066	3016	3013	3016	3015.0
bajo/alto/alto	0.00066	0.001	0.00133	0.001	3015	3016	3017	3016.0
bajo/alto/bajo	0	0.001	0.00199	0.001	3013	3016	3019	3016.0
bajo/bajo/alto	0.00199	0.001	0.00199	0.00166	3019	3016	3019	3018.0
0.75-0.25								
alto/alto/bajo	0.001	0.00066	0.00166	0.00111	3016	3015	3018	3016.3
alto/bajo/alto	0.00066	0.00232	0.00066	0.00122	3015	3020	3015	3016.7
alto/bajo/bajo	0	0.001	0.00066	0.00055	3013	3016	3015	3014.7
bajo/alto/alto	0.00166	0.00166	0.00033	0.00122	3018	3018	3014	3016.7
bajo/alto/bajo	0.001	0.00199	0.001	0.00133	3016	3019	3016	3017.0
bajo/bajo/alto	0.00033	0.00133	0.00199	0.00122	3014	3017	3019	3016.7
9 partículas								
0.25-0.75								
alto/alto/bajo	0.00066	0.001	0.001	0.00089	3015	3016	3016	3015.7
alto/bajo/alto	0.00033	0.00133	0.00066	0.00077	3014	3017	3015	3015.3
alto/bajo/bajo	0.00133	0.00199	0.00199	0.00177	3017	3019	3019	3018.3
bajo/alto/alto	0.00133	0.00033	0.00033	0.00066	3017	3014	3014	3015.0
bajo/alto/bajo	0.00066	0.00033	0.00033	0.00044	3015	3014	3014	3014.3
bajo/bajo/alto	0.00033	0.001	0.00066	0.00066	3014	3016	3015	3015.0
0.5-0.5								
alto/alto/bajo	0	0.00066	0.00133	0.00066	3013	3015	3017	3015.0
alto/bajo/alto	0.00133	0.00033	0.00133	0.001	3017	3014	3017	3016.0
alto/bajo/bajo	0.00066	0.00066	0.001	0.00077	3015	3015	3016	3015.3
bajo/alto/alto	0.00133	0.00066	0.00133	0.00111	3017	3015	3017	3016.3
bajo/alto/bajo	0.00066	0.00166	0.001	0.00111	3015	3018	3016	3016.3
bajo/bajo/alto	0.00033	0.00133	0.00133	0.001	3014	3017	3017	3016.0
0.75-0.25								
alto/alto/bajo	0.00066	0.00133	0.00166	0.00122	3015	3017	3018	3016.7
alto/bajo/alto	0	0.00066	0.00166	0.00077	3013	3015	3018	3015.3
alto/bajo/bajo	0.00199	0.00066	0.00033	0.001	3019	3015	3014	3016.0
bajo/alto/alto	0	0.001	0.00066	0.00055	3013	3016	3015	3014.7
bajo/alto/bajo	0.00033	0.00199	0.00066	0.001	3014	3019	3015	3016.0
bajo/bajo/alto	0.001	0.00066	0.00066	0.00077	3016	3015	3015	3015.3
10 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00033	0.001	0.001	0.00077	3014	3016	3016	3015.3
alto/bajo/alto	0.001	0.001	0.00066	0.00089	3016	3016	3015	3015.7
alto/bajo/bajo	0.001	0.00133	0.001	0.00111	3016	3017	3016	3016.3
bajo/alto/alto	0.00033	0.001	0.00033	0.00055	3014	3016	3014	3014.7
bajo/alto/bajo	0.00199	0.00199	0.00066	0.00155	3019	3019	3015	3017.7
bajo/bajo/alto	0.00066	0.001	0.00033	0.00066	3015	3016	3014	3015.0
0.5-0.5								
alto/alto/bajo	0.00133	0.001	0.00033	0.00089	3017	3016	3014	3015.7
alto/bajo/alto	0.00166	0.001	0.00033	0.001	3018	3016	3014	3016.0
alto/bajo/bajo	0.001	0.001	0.00166	0.00122	3016	3016	3018	3016.7
bajo/alto/alto	0.00199	0.00133	0.00066	0.00133	3019	3017	3015	3017.0
bajo/alto/bajo	0.00033	0.001	0.00199	0.00111	3014	3016	3019	3016.3
bajo/bajo/alto	0.00133	0	0	0.00044	3017	3013	3013	3014.3

0.75-0.25								
alto/alto/bajo	0.00133	0.00066	0.00299	0.00166	3017	3015	3022	3018.0
alto/bajo/alto	0.00133	0.00166	0.001	0.00133	3017	3018	3016	3017.0
alto/bajo/bajo	0.00199	0.00066	0.00199	0.00155	3019	3015	3019	3017.7
bajo/alto/alto	0.00199	0	0.00033	0.00077	3019	3013	3014	3015.3
bajo/alto/bajo	0.001	0.00033	0.00166	0.001	3016	3014	3018	3016.0
bajo/bajo/alto	0.00166	0.00232	0.00166	0.00188	3018	3020	3018	3018.7
6 partículas								
0.25-0.75								
alto/alto/bajo	0.00133	0.001	0.00166	0.00133	3017	3016	3018	3017.0
alto/bajo/alto	0.001	0	0.001	0.00066	3016	3013	3016	3015.0
alto/bajo/bajo	0.001	0	0.00133	0.00077	3016	3013	3017	3015.3
bajo/alto/alto	0.001	0.00066	0.00033	0.00066	3016	3015	3014	3015.0
bajo/alto/bajo	0.00033	0.00033	0	0.00022	3014	3014	3013	3013.7
bajo/bajo/alto	0.00066	0.00066	0.00133	0.00089	3015	3015	3017	3015.7
0.5-0.5								
alto/alto/bajo	0.001	0.00166	0.001	0.00122	3016	3018	3016	3016.7
alto/bajo/alto	0.001	0.001	0.00066	0.00089	3016	3016	3015	3015.7
alto/bajo/bajo	0.00066	0.00066	0.00133	0.00089	3015	3015	3017	3015.7
bajo/alto/alto	0.001	0.00033	0	0.00044	3016	3014	3013	3014.3
bajo/alto/bajo	0.00166	0.00066	0.00066	0.001	3018	3015	3015	3016.0
bajo/bajo/alto	0.00066	0.00066	0.00066	0.00066	3015	3015	3015	3015.0
0.75-0.25								
alto/alto/bajo	0.00033	0.00133	0.00166	0.00111	3014	3017	3018	3016.3
alto/bajo/alto	0.001	0.001	0.00033	0.00077	3016	3016	3014	3015.3
alto/bajo/bajo	0.001	0.00066	0.00066	0.00077	3016	3015	3015	3015.3
bajo/alto/alto	0.001	0.00133	0.001	0.00111	3016	3017	3016	3016.3
bajo/alto/bajo	0.00066	0.00066	0.00066	0.00066	3015	3015	3015	3015.0
bajo/bajo/alto	0.00133	0.00199	0.00133	0.00155	3017	3019	3017	3017.7
9 partículas								
0.25-0.75								
alto/alto/bajo	0.00066	0.001	0.001	0.00089	3015	3016	3016	3015.7
alto/bajo/alto	0.00133	0	0.00066	0.00066	3017	3013	3015	3015.0
alto/bajo/bajo	0.00033	0.00066	0	0.00033	3014	3015	3013	3014.0
bajo/alto/alto	0.00066	0.00066	0.00066	0.00066	3015	3015	3015	3015.0
bajo/alto/bajo	0.00066	0.00033	0	0.00033	3015	3014	3013	3014.0
bajo/bajo/alto	0.001	0.00066	0.00066	0.00077	3016	3015	3015	3015.3
0.5-0.5								
alto/alto/bajo	0.00033	0.00066	0.001	0.00066	3014	3015	3016	3015.0
alto/bajo/alto	0.00033	0	0.00033	0.00022	3014	3013	3014	3013.7
alto/bajo/bajo	0.00033	0.00033	0.001	0.00055	3014	3014	3016	3014.7
bajo/alto/alto	0.00066	0.00166	0.00066	0.001	3015	3018	3015	3016.0
bajo/alto/bajo	0.00066	0.00066	0.001	0.00077	3015	3015	3016	3015.3
bajo/bajo/alto	0.00166	0.00033	0.001	0.001	3018	3014	3016	3016.0
0.75-0.25								
alto/alto/bajo	0	0.001	0.00133	0.00077	3013	3016	3017	3015.3
alto/bajo/alto	0.00133	0.00066	0.001	0.001	3017	3015	3016	3016.0
alto/bajo/bajo	0.00066	0.001	0.00033	0.00066	3015	3016	3014	3015.0
bajo/alto/alto	0.001	0.00166	0.001	0.00122	3016	3018	3016	3016.7
bajo/alto/bajo	0.001	0.00033	0.00033	0.00055	3016	3014	3014	3014.7
bajo/bajo/alto	0.00133	0	0.00033	0.00055	3017	3013	3014	3014.7

15 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00166	0.00066	0.00066	0.001	3018	3015	3015	3016.0
alto/bajo/alto	0.00166	0.001	0.00066	0.00111	3018	3016	3015	3016.3
alto/bajo/bajo	0.001	0.00166	0.001	0.00122	3016	3018	3016	3016.7
bajo/alto/alto	0.00133	0.00066	0.001	0.001	3017	3015	3016	3016.0
bajo/alto/bajo	0.00066	0.00166	0.001	0.00111	3015	3018	3016	3016.3
bajo/bajo/alto	0.001	0.00133	0.00166	0.00133	3016	3017	3018	3017.0
0.5-0.5								
alto/alto/bajo	0.001	0.00133	0.00133	0.00122	3016	3017	3017	3016.7
alto/bajo/alto	0.00066	0.00033	0.00133	0.00077	3015	3014	3017	3015.3
alto/bajo/bajo	0.00033	0.00133	0.00133	0.001	3014	3017	3017	3016.0
bajo/alto/alto	0.00066	0.00166	0.00066	0.001	3015	3018	3015	3016.0
bajo/alto/bajo	0.00133	0.001	0.001	0.00111	3017	3016	3016	3016.3
bajo/bajo/alto	0.00066	0.00066	0.00066	0.00066	3015	3015	3015	3015.0
0.75-0.25								
alto/alto/bajo	0.00066	0.00066	0.00133	0.00089	3015	3015	3017	3015.7
alto/bajo/alto	0.00133	0.00033	0.00033	0.00066	3017	3014	3014	3015.0
alto/bajo/bajo	0.001	0.00033	0.00166	0.001	3016	3014	3018	3016.0
bajo/alto/alto	0.001	0.001	0.00166	0.00122	3016	3016	3018	3016.7
bajo/alto/bajo	0.00166	0.00066	0.00133	0.00122	3018	3015	3017	3016.7
bajo/bajo/alto	0.001	0.001	0.00066	0.00089	3016	3016	3015	3015.7
6 partículas								
0.25-0.75								
alto/alto/bajo	0.00033	0.00033	0.00066	0.00044	3014	3014	3015	3014.3
alto/bajo/alto	0.00033	0.00066	0.00033	0.00044	3014	3015	3014	3014.3
alto/bajo/bajo	0.00133	0.00166	0.001	0.00133	3017	3018	3016	3017.0
bajo/alto/alto	0.00199	0.001	0.00033	0.00111	3019	3016	3014	3016.3
bajo/alto/bajo	0.00033	0.00033	0.00033	0.00033	3014	3014	3014	3014.0
bajo/bajo/alto	0.00033	0.00033	0.00066	0.00044	3014	3014	3015	3014.3
0.5-0.5								
alto/alto/bajo	0.001	0.00066	0.00066	0.00077	3016	3015	3015	3015.3
alto/bajo/alto	0.00033	0.001	0.00033	0.00055	3014	3016	3014	3014.7
alto/bajo/bajo	0.00133	0.00066	0.00033	0.00077	3017	3015	3014	3015.3
bajo/alto/alto	0.001	0.001	0.00066	0.00089	3016	3016	3015	3015.7
bajo/alto/bajo	0	0.00033	0.001	0.00044	3013	3014	3016	3014.3
bajo/bajo/alto	0.00066	0.00066	0	0.00044	3015	3015	3013	3014.3
0.75-0.25								
alto/alto/bajo	0.00066	0.001	0.00066	0.00077	3015	3016	3015	3015.3
alto/bajo/alto	0.00033	0.00066	0.00066	0.00055	3014	3015	3015	3014.7
alto/bajo/bajo	0.00166	0.00033	0	0.00066	3018	3014	3013	3015.0
bajo/alto/alto	0.00066	0.00033	0.00066	0.00055	3015	3014	3015	3014.7
bajo/alto/bajo	0.00133	0	0.00133	0.00089	3017	3013	3017	3015.7
bajo/bajo/alto	0.001	0.00066	0	0.00055	3016	3015	3013	3014.7
9 partículas								
0.25-0.75								
alto/alto/bajo	0	0.00033	0.00066	0.00033	3013	3014	3015	3014.0
alto/bajo/alto	0.00033	0.00066	0	0.00033	3014	3015	3013	3014.0
alto/bajo/bajo	0.00066	0.00066	0	0.00044	3015	3015	3013	3014.3
bajo/alto/alto	0	0.00066	0.00033	0.00033	3013	3015	3014	3014.0
bajo/alto/bajo	0.00066	0.00066	0.00066	0.00066	3015	3015	3015	3015.0
bajo/bajo/alto	0	0.00033	0.00033	0.00022	3013	3014	3014	3013.7

0.5-0.5								
alto/alto/bajo	0.00033	0.00033	0.00033	0.00033	3014	3014	3014	3014.0
alto/bajo/alto	0.00066	0.00066	0.00166	0.001	3015	3015	3018	3016.0
alto/bajo/bajo	0.00033	0.00133	0.00066	0.00077	3014	3017	3015	3015.3
bajo/alto/alto	0.001	0.00066	0.00066	0.00077	3016	3015	3015	3015.3
bajo/alto/bajo	0	0	0.00033	0.00011	3013	3013	3014	3013.3
bajo/bajo/alto	0.00066	0.00033	0.00033	0.00044	3015	3014	3014	3014.3
0.75-0.25								
alto/alto/bajo	0.001	0.001	0	0.00066	3016	3016	3013	3015.0
alto/bajo/alto	0.00033	0.00066	0.00033	0.00044	3014	3015	3014	3014.3
alto/bajo/bajo	0.00033	0.00033	0.00066	0.00044	3014	3014	3015	3014.3
bajo/alto/alto	0.00033	0.00066	0.00033	0.00044	3014	3015	3014	3014.3
bajo/alto/bajo	0.00033	0.00033	0.00033	0.00033	3014	3014	3014	3014.0
bajo/bajo/alto	0.00066	0.00133	0.00033	0.00077	3015	3017	3014	3015.3
20 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00033	0.00033	0.00066	0.00044	3014	3014	3015	3014.3
alto/bajo/alto	0.00033	0.001	0.00133	0.00089	3014	3016	3017	3015.7
alto/bajo/bajo	0	0.00033	0.00066	0.00033	3013	3014	3015	3014.0
bajo/alto/alto	0.001	0.001	0.00133	0.00111	3016	3016	3017	3016.3
bajo/alto/bajo	0.00066	0.00066	0.00133	0.00089	3015	3015	3017	3015.7
bajo/bajo/alto	0.001	0.00066	0.00066	0.00077	3016	3015	3015	3015.3
0.5-0.5								
alto/alto/bajo	0.00033	0	0.001	0.00044	3014	3013	3016	3014.3
alto/bajo/alto	0	0.001	0.001	0.00066	3013	3016	3016	3015.0
alto/bajo/bajo	0.001	0.00166	0.001	0.00122	3016	3018	3016	3016.7
bajo/alto/alto	0.001	0.00066	0.00033	0.00066	3016	3015	3014	3015.0
bajo/alto/bajo	0.001	0.00066	0.001	0.00089	3016	3015	3016	3015.7
bajo/bajo/alto	0	0.001	0.00166	0.00089	3013	3016	3018	3015.7
0.75-0.25								
alto/alto/bajo	0.00066	0.00066	0.00133	0.00089	3015	3015	3017	3015.7
alto/bajo/alto	0.00066	0.00033	0.00033	0.00044	3015	3014	3014	3014.3
alto/bajo/bajo	0.00133	0.00033	0.00066	0.00077	3017	3014	3015	3015.3
bajo/alto/alto	0.00066	0.00066	0.001	0.00077	3015	3015	3016	3015.3
bajo/alto/bajo	0.00133	0.00066	0.00033	0.00077	3017	3015	3014	3015.3
bajo/bajo/alto	0.001	0.00033	0.001	0.00077	3016	3014	3016	3015.3
6 partículas								
0.25-0.75								
alto/alto/bajo	0.00033	0.00066	0.00033	0.00044	3014	3015	3014	3014.3
alto/bajo/alto	0.00033	0	0.00066	0.00033	3014	3013	3015	3014.0
alto/bajo/bajo	0	0.001	0.00033	0.00044	3013	3016	3014	3014.3
bajo/alto/alto	0.00033	0.00033	0.00033	0.00033	3014	3014	3014	3014.0
bajo/alto/bajo	0	0.00066	0.00066	0.00044	3013	3015	3015	3014.3
bajo/bajo/alto	0.00033	0.00166	0.00033	0.00077	3014	3018	3014	3015.3
0.5-0.5								
alto/alto/bajo	0.00033	0.001	0.00066	0.00066	3014	3016	3015	3015.0
alto/bajo/alto	0.00066	0.001	0.001	0.00089	3015	3016	3016	3015.7
alto/bajo/bajo	0.001	0.001	0.00066	0.00089	3016	3016	3015	3015.7
bajo/alto/alto	0.00033	0.00066	0	0.00033	3014	3015	3013	3014.0
bajo/alto/bajo	0.001	0.00033	0.001	0.00077	3016	3014	3016	3015.3
bajo/bajo/alto	0.00033	0.00066	0.00133	0.00077	3014	3015	3017	3015.3

0.75-0.25								
alto/alto/bajo	0.00033	0.00066	0.00066	0.00055	3014	3015	3015	3014.7
alto/bajo/alto	0.00033	0.001	0.00066	0.00066	3014	3016	3015	3015.0
alto/bajo/bajo	0.00033	0.00066	0.00066	0.00055	3014	3015	3015	3014.7
bajo/alto/alto	0.00066	0.00066	0.00033	0.00055	3015	3015	3014	3014.7
bajo/alto/bajo	0	0.001	0.00033	0.00044	3013	3016	3014	3014.3
bajo/bajo/alto	0.00066	0.00033	0.00066	0.00055	3015	3014	3015	3014.7
9 partículas								
0.25-0.75								
alto/alto/bajo	0.00033	0.00033	0.001	0.00055	3014	3014	3016	3014.7
alto/bajo/alto	0.00033	0.00066	0.00066	0.00055	3014	3015	3015	3014.7
alto/bajo/bajo	0	0.00033	0.00066	0.00033	3013	3014	3015	3014.0
bajo/alto/alto	0	0.00066	0.00033	0.00033	3013	3015	3014	3014.0
bajo/alto/bajo	0.00066	0.00033	0.00066	0.00055	3015	3014	3015	3014.7
bajo/bajo/alto	0.00033	0.001	0.00066	0.00066	3014	3016	3015	3015.0
0.5-0.5								
alto/alto/bajo	0.001	0.00033	0.00066	0.00066	3016	3014	3015	3015.0
alto/bajo/alto	0	0.00066	0.00033	0.00033	3013	3015	3014	3014.0
alto/bajo/bajo	0.00133	0.00033	0.00033	0.00066	3017	3014	3014	3015.0
bajo/alto/alto	0.00066	0.00033	0.00066	0.00055	3015	3014	3015	3014.7
bajo/alto/bajo	0.001	0.00033	0.00033	0.00055	3016	3014	3014	3014.7
bajo/bajo/alto	0	0.00066	0.00066	0.00044	3013	3015	3015	3014.3
0.75-0.25								
alto/alto/bajo	0.00033	0.001	0.00033	0.00055	3014	3016	3014	3014.7
alto/bajo/alto	0.00033	0.00066	0.00066	0.00055	3014	3015	3015	3014.7
alto/bajo/bajo	0.00066	0.00033	0.00033	0.00044	3015	3014	3014	3014.3
bajo/alto/alto	0.001	0.00133	0.00033	0.00089	3016	3017	3014	3015.7
bajo/alto/bajo	0.00066	0.001	0.00066	0.00077	3015	3016	3015	3015.3
bajo/bajo/alto	0.00133	0.00033	0.00066	0.00077	3017	3014	3015	3015.3

2.3. Problema pmed37

pmed37	Variable de Respuesta (Desviación del Optimo)				Valor QPSO			
Valor Optimo: 5057	Replicas			Medias	Replicas			Medias
	1	2	3		1	2	3	
5 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.0002	0.0002	0.0004	0.00079	5058	5058	5059	5058.3
alto/bajo/alto	0.0004	0.0004	0.0004	0.00119	5059	5059	5059	5059.0
alto/bajo/bajo	0.0002	0.0004	0.00079	0.00138	5058	5059	5061	5059.3
bajo/alto/alto	0.0002	0.0004	0	0.00059	5058	5059	5057	5058.0
bajo/alto/bajo	0.0002	0.0002	0.00099	0.00138	5058	5058	5062	5059.3
bajo/bajo/alto	0.00059	0.00059	0.00099	0.00218	5060	5060	5062	5060.7
0.5-0.5								
alto/alto/bajo	0.0004	0	0	0.0004	5059	5057	5057	5057.7
alto/bajo/alto	0.0002	0.0002	0.0004	0.00079	5058	5058	5059	5058.3
alto/bajo/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
bajo/alto/alto	0.00079	0.00079	0.0004	0.00198	5061	5061	5059	5060.3
bajo/alto/bajo	0.00079	0	0	0.00079	5061	5057	5057	5058.3
bajo/bajo/alto	0.0002	0.0002	0.00079	0.00119	5058	5058	5061	5059.0

0.75-0.25								
alto/alto/bajo	0.0002	0.0002	0.00079	0.00119	5058	5058	5061	5059.0
alto/bajo/alto	0.0004	0.0002	0.00079	0.00138	5059	5058	5061	5059.3
alto/bajo/bajo	0.0004	0.0002	0.0002	0.00079	5059	5058	5058	5058.3
bajo/alto/alto	0.0004	0	0.0002	0.00059	5059	5057	5058	5058.0
bajo/alto/bajo	0.0002	0	0.00059	0.00079	5058	5057	5060	5058.3
bajo/bajo/alto	0.00079	0.0002	0.0002	0.00119	5061	5058	5058	5059.0
6 partículas								
0.25-0.75								
alto/alto/bajo	0.0002	0.0002	0.0004	0.00079	5058	5058	5059	5058.3
alto/bajo/alto	0.0004	0.0002	0.0002	0.00079	5059	5058	5058	5058.3
alto/bajo/bajo	0.0002	0.0002	0.0004	0.00079	5058	5058	5059	5058.3
bajo/alto/alto	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/alto/bajo	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
bajo/bajo/alto	0	0.0004	0.0002	0.00059	5057	5059	5058	5058.0
0.5-0.5								
alto/alto/bajo	0	0.0004	0.0002	0.00059	5057	5059	5058	5058.0
alto/bajo/alto	0.0002	0	0.0004	0.00059	5058	5057	5059	5058.0
alto/bajo/bajo	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/alto/alto	0.0004	0.0002	0	0.00059	5059	5058	5057	5058.0
bajo/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/bajo/alto	0	0	0	0	5057	5057	5057	5057.0
0.75-0.25								
alto/alto/bajo	0.0002	0.0002	0.00059	0.00099	5058	5058	5060	5058.7
alto/bajo/alto	0.0002	0.0004	0.00059	0.00119	5058	5059	5060	5059.0
alto/bajo/bajo	0.0004	0.0004	0.0002	0.00099	5059	5059	5058	5058.7
bajo/alto/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
bajo/alto/bajo	0.0004	0.0002	0.0002	0.00079	5059	5058	5058	5058.3
bajo/bajo/alto	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
9 partículas								
0.25-0.75								
alto/alto/bajo	0.0002	0.0004	0	0.00059	5058	5059	5057	5058.0
alto/bajo/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
alto/bajo/bajo	0.0004	0.0002	0.0004	0.00099	5059	5058	5059	5058.7
bajo/alto/alto	0.0002	0	0.0004	0.00059	5058	5057	5059	5058.0
bajo/alto/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
bajo/bajo/alto	0	0.0002	0.00059	0.00079	5057	5058	5060	5058.3
0.5-0.5								
alto/alto/bajo	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
alto/bajo/alto	0.0002	0	0.00059	0.00079	5058	5057	5060	5058.3
alto/bajo/bajo	0	0.0004	0.0002	0.00059	5057	5059	5058	5058.0
bajo/alto/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
bajo/alto/bajo	0	0	0.0002	0.0002	5057	5057	5058	5057.3
bajo/bajo/alto	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
0.75-0.25								
alto/alto/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
alto/bajo/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
alto/bajo/bajo	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
bajo/alto/alto	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/alto/bajo	0.0002	0.0004	0	0.00059	5058	5059	5057	5058.0
bajo/bajo/alto	0.0002	0.0002	0.0004	0.00079	5058	5058	5059	5058.3

10 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0	0.0004	0.0002	0.00059	5057	5059	5058	5058.0
alto/bajo/alto	0.0004	0	0.0002	0.00059	5059	5057	5058	5058.0
alto/bajo/bajo	0.0002	0	0.0004	0.00059	5058	5057	5059	5058.0
bajo/alto/alto	0.0004	0.0004	0.0002	0.00099	5059	5059	5058	5058.7
bajo/alto/bajo	0	0.0004	0.0004	0.00079	5057	5059	5059	5058.3
bajo/bajo/alto	0	0.0002	0	0.0002	5057	5058	5057	5057.3
0.5-0.5								
alto/alto/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
alto/bajo/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
alto/bajo/bajo	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
bajo/alto/alto	0	0.00079	0.0002	0.00099	5057	5061	5058	5058.7
bajo/alto/bajo	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
bajo/bajo/alto	0	0.0002	0.0004	0.00059	5057	5058	5059	5058.0
0.75-0.25								
alto/alto/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
alto/bajo/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
alto/bajo/bajo	0.00059	0.00079	0.0002	0.00158	5060	5061	5058	5059.7
bajo/alto/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
bajo/alto/bajo	0.0004	0.0002	0.0002	0.00079	5059	5058	5058	5058.3
bajo/bajo/alto	0	0.0002	0.0004	0.00059	5057	5058	5059	5058.0
6 partículas								
0.25-0.75								
alto/alto/bajo	0	0.0004	0.0002	0.00059	5057	5059	5058	5058.0
alto/bajo/alto	0	0.0002	0	0.0002	5057	5058	5057	5057.3
alto/bajo/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
bajo/alto/alto	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
bajo/alto/bajo	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
bajo/bajo/alto	0	0.0002	0.0004	0.00059	5057	5058	5059	5058.0
0.5-0.5								
alto/alto/bajo	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
alto/bajo/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
alto/bajo/bajo	0.0004	0.0002	0	0.00059	5059	5058	5057	5058.0
bajo/alto/alto	0	0	0.0002	0.0002	5057	5057	5058	5057.3
bajo/alto/bajo	0	0.0004	0	0.0004	5057	5059	5057	5057.7
bajo/bajo/alto	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
0.75-0.25								
alto/alto/bajo	0.00059	0	0.0002	0.00079	5060	5057	5058	5058.3
alto/bajo/alto	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
alto/bajo/bajo	0.0002	0	0.0004	0.00059	5058	5057	5059	5058.0
bajo/alto/alto	0	0	0.0002	0.0002	5057	5057	5058	5057.3
bajo/alto/bajo	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/bajo/alto	0	0.0002	0	0.0002	5057	5058	5057	5057.3
9 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
alto/bajo/alto	0	0.0002	0	0.0002	5057	5058	5057	5057.3
alto/bajo/bajo	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/alto/alto	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/alto/bajo	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/bajo/alto	0	0	0	0	5057	5057	5057	5057.0

0.5-0.5								
alto/alto/bajo	0	0	0.0002	0.0002	5057	5057	5058	5057.3
alto/bajo/alto	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
alto/bajo/bajo	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/alto/alto	0	0	0.0002	0.0002	5057	5057	5058	5057.3
bajo/alto/bajo	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/bajo/alto	0	0	0.0002	0.0002	5057	5057	5058	5057.3
0.75-0.25								
alto/alto/bajo	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
alto/bajo/alto	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
alto/bajo/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/alto	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/bajo	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
bajo/bajo/alto	0	0	0	0	5057	5057	5057	5057.0
15 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
alto/bajo/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
alto/bajo/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
bajo/alto/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
bajo/alto/bajo	0.00059	0.0002	0.0002	0.00099	5060	5058	5058	5058.7
bajo/bajo/alto	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
0.5-0.5								
alto/alto/bajo	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
alto/bajo/alto	0.0004	0.0002	0.0002	0.00079	5059	5058	5058	5058.3
alto/bajo/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/alto/alto	0.0002	0.0004	0.0002	0.00079	5058	5059	5058	5058.3
bajo/alto/bajo	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
bajo/bajo/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
0.75-0.25								
alto/alto/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
alto/bajo/alto	0	0.0004	0.0004	0.00079	5057	5059	5059	5058.3
alto/bajo/bajo	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
bajo/alto/alto	0	0.00059	0	0.00059	5057	5060	5057	5058.0
bajo/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/bajo/alto	0.0002	0	0	0.0002	5058	5057	5057	5057.3
6 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0.0002	0.0002	5057	5057	5058	5057.3
alto/bajo/alto	0.0002	0.0004	0.0002	0.00079	5058	5059	5058	5058.3
alto/bajo/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/alto	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/alto/bajo	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
bajo/bajo/alto	0.0002	0	0	0.0002	5058	5057	5057	5057.3
0.5-0.5								
alto/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
alto/bajo/alto	0.0004	0.0002	0	0.00059	5059	5058	5057	5058.0
alto/bajo/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/alto/alto	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/bajo/alto	0	0	0.0004	0.0004	5057	5057	5059	5057.7

0.75-0.25								
alto/alto/bajo	0	0.0004	0.0002	0.00059	5057	5059	5058	5058.0
alto/bajo/alto	0	0.0002	0.0004	0.00059	5057	5058	5059	5058.0
alto/bajo/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/alto	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
bajo/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/bajo/alto	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
9 partículas								
0.25-0.75								
alto/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
alto/bajo/alto	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
alto/bajo/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/alto/alto	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/bajo/alto	0	0	0	0	5057	5057	5057	5057.0
0.5-0.5								
alto/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
alto/bajo/alto	0	0	0	0	5057	5057	5057	5057.0
alto/bajo/bajo	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
bajo/alto/alto	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/bajo	0.0004	0.0002	0.0002	0.00079	5059	5058	5058	5058.3
bajo/bajo/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
0.75-0.25								
alto/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
alto/bajo/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
alto/bajo/bajo	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/alto/alto	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/bajo	0	0	0.0002	0.0002	5057	5057	5058	5057.3
bajo/bajo/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
20 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.0002	0.0002	0.0004	0.00079	5058	5058	5059	5058.3
alto/bajo/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
alto/bajo/bajo	0	0	0.0002	0.0002	5057	5057	5058	5057.3
bajo/alto/alto	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/alto/bajo	0.0002	0.0002	0.0004	0.00079	5058	5058	5059	5058.3
bajo/bajo/alto	0.0002	0.0002	0.0004	0.00079	5058	5058	5059	5058.3
0.5-0.5								
alto/alto/bajo	0	0	0.0002	0.0002	5057	5057	5058	5057.3
alto/bajo/alto	0.0004	0	0.0004	0.00079	5059	5057	5059	5058.3
alto/bajo/bajo	0.00059	0	0	0.00059	5060	5057	5057	5058.0
bajo/alto/alto	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
bajo/alto/bajo	0	0.0002	0	0.0002	5057	5058	5057	5057.3
bajo/bajo/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
0.75-0.25								
alto/alto/bajo	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
alto/bajo/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
alto/bajo/bajo	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
bajo/alto/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
bajo/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/bajo/alto	0.0002	0	0	0.0002	5058	5057	5057	5057.3

6 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
alto/bajo/alto	0	0	0	0	5057	5057	5057	5057.0
alto/bajo/bajo	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
bajo/alto/alto	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/bajo/alto	0	0	0.0002	0.0002	5057	5057	5058	5057.3
0.5-0.5								
alto/alto/bajo	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
alto/bajo/alto	0.0002	0.0002	0.0002	0.00059	5058	5058	5058	5058.0
alto/bajo/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/alto	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
bajo/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/bajo/alto	0.0002	0	0	0.0002	5058	5057	5057	5057.3
0.75-0.25								
alto/alto/bajo	0	0.0002	0.0002	0.0004	5057	5058	5058	5057.7
alto/bajo/alto	0.0002	0	0	0.0002	5058	5057	5057	5057.3
alto/bajo/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/alto	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/bajo/alto	0.0002	0	0.0002	0.0004	5058	5057	5058	5057.7
9 partículas								
0.25-0.75								
alto/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
alto/bajo/alto	0	0	0	0	5057	5057	5057	5057.0
alto/bajo/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/alto	0	0	0.0002	0.0002	5057	5057	5058	5057.3
bajo/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/bajo/alto	0.0002	0.0002	0	0.0004	5058	5058	5057	5057.7
0.5-0.5								
alto/alto/bajo	0	0	0.0002	0.0002	5057	5057	5058	5057.3
alto/bajo/alto	0	0	0.0002	0.0002	5057	5057	5058	5057.3
alto/bajo/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
bajo/alto/alto	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/bajo	0	0	0.0002	0.0002	5057	5057	5058	5057.3
bajo/bajo/alto	0	0	0	0	5057	5057	5057	5057.0
0.75-0.25								
alto/alto/bajo	0.0002	0	0	0.0002	5058	5057	5057	5057.3
alto/bajo/alto	0	0	0.0002	0.0002	5057	5057	5058	5057.3
alto/bajo/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/alto	0	0	0	0	5057	5057	5057	5057.0
bajo/alto/bajo	0	0	0	0	5057	5057	5057	5057.0
bajo/bajo/alto	0	0	0	0	5057	5057	5057	5057.0

2.4. Problema pmed40

pmed40	Variable de Respuesta (Desviación del Optimo)				Valor QPSO			
Valor Optimo:	Replicas			Medias	Replicas			Medias
5128	1	2	3		1	2	3	
5 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00117	0.00059	0.00059	0.00234	5134	5131	5131	5132.0
alto/bajo/alto	0.00039	0.00059	0.00098	0.00195	5130	5131	5133	5131.3
alto/bajo/bajo	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
bajo/alto/alto	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
bajo/alto/bajo	0.001365	0.00059	0.00078	0.00273	5135	5131	5132	5132.7
bajo/bajo/alto	0.00117	0.00039	0.00098	0.002535	5134	5130	5133	5132.3
0.5-0.5								
alto/alto/bajo	0.000585	0.00078	0	0.001365	5131	5132	5128	5130.3
alto/bajo/alto	0.00078	0.00059	0.00078	0.002145	5132	5131	5132	5131.7
alto/bajo/bajo	0.00078	0.00078	0.00078	0.00234	5132	5132	5132	5132.0
bajo/alto/alto	0.00156	0.00117	0.00117	0.0039	5136	5134	5134	5134.7
bajo/alto/bajo	0.000975	0.00195	0.00117	0.004095	5133	5138	5134	5135.0
bajo/bajo/alto	0.000585	0.00078	0.00059	0.00195	5131	5132	5131	5131.3
0.75-0.25								
alto/alto/bajo	0.00078	0.00039	0.00098	0.002145	5132	5130	5133	5131.7
alto/bajo/alto	0.00117	0.00098	0.00059	0.00273	5134	5133	5131	5132.7
alto/bajo/bajo	0.00078	0.00156	0.00039	0.00273	5132	5136	5130	5132.7
bajo/alto/alto	0.001755	0.00137	0.00059	0.003705	5137	5135	5131	5134.3
bajo/alto/bajo	0.001365	0.00156	0.00039	0.003315	5135	5136	5130	5133.7
bajo/bajo/alto	0.000585	0.00078	0.00039	0.001755	5131	5132	5130	5131.0
6 partículas								
0.25-0.75								
alto/alto/bajo	0.00078	0.0002	0.00117	0.002145	5132	5129	5134	5131.7
alto/bajo/alto	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
alto/bajo/bajo	0.000195	0.00039	0.00039	0.000975	5129	5130	5130	5129.7
bajo/alto/alto	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
bajo/alto/bajo	0.000975	0.00039	0.00078	0.002145	5133	5130	5132	5131.7
bajo/bajo/alto	0.00039	0.00098	0.00098	0.00234	5130	5133	5133	5132.0
0.5-0.5								
alto/alto/bajo	0.00078	0.0002	0.00098	0.00195	5132	5129	5133	5131.3
alto/bajo/alto	0.00039	0.00039	0.0002	0.000975	5130	5130	5129	5129.7
alto/bajo/bajo	0.000585	0.00059	0.00059	0.001755	5131	5131	5131	5131.0
bajo/alto/alto	0.000975	0.00117	0.00117	0.003315	5133	5134	5134	5133.7
bajo/alto/bajo	0.00117	0.00117	0.00059	0.002925	5134	5134	5131	5133.0
bajo/bajo/alto	0.00039	0.00059	0	0.000975	5130	5131	5128	5129.7
0.75-0.25								
alto/alto/bajo	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
alto/bajo/alto	0.00078	0.00059	0.0002	0.00156	5132	5131	5129	5130.7
alto/bajo/bajo	0.000585	0.00039	0.00117	0.002145	5131	5130	5134	5131.7
bajo/alto/alto	0.000585	0.0002	0.0002	0.000975	5131	5129	5129	5129.7
bajo/alto/bajo	0.000585	0.00039	0.00098	0.00195	5131	5130	5133	5131.3
bajo/bajo/alto	0.00039	0.00039	0.0002	0.000975	5130	5130	5129	5129.7

9 partículas								
0.25-0.75								
alto/alto/bajo	0.00078	0.00059	0.00039	0.001755	5132	5131	5130	5131.0
alto/bajo/alto	0.000195	0.00039	0.00039	0.000975	5129	5130	5130	5129.7
alto/bajo/bajo	0.00039	0	0.00059	0.000975	5130	5128	5131	5129.7
bajo/alto/alto	0.00039	0.00078	0.0002	0.001365	5130	5132	5129	5130.3
bajo/alto/bajo	0.000585	0.0002	0.00078	0.00156	5131	5129	5132	5130.7
bajo/bajo/alto	0.00078	0.00059	0.0002	0.00156	5132	5131	5129	5130.7
0.5-0.5								
alto/alto/bajo	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
alto/bajo/alto	0.00039	0.00059	0.00059	0.00156	5130	5131	5131	5130.7
alto/bajo/bajo	0.000585	0.00039	0.00039	0.001365	5131	5130	5130	5130.3
bajo/alto/alto	0.000585	0.0002	0.00039	0.00117	5131	5129	5130	5130.0
bajo/alto/bajo	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
bajo/bajo/alto	0.00039	0.00039	0.00078	0.00156	5130	5130	5132	5130.7
0.75-0.25								
alto/alto/bajo	0.000195	0.00098	0.00039	0.00156	5129	5133	5130	5130.7
alto/bajo/alto	0.000975	0.00078	0.00039	0.002145	5133	5132	5130	5131.7
alto/bajo/bajo	0.00039	0.00059	0.00078	0.001755	5130	5131	5132	5131.0
bajo/alto/alto	0.00039	0.00098	0.00098	0.00234	5130	5133	5133	5132.0
bajo/alto/bajo	0.00078	0.00039	0.00039	0.00156	5132	5130	5130	5130.7
bajo/bajo/alto	0.00039	0.00078	0.00059	0.001755	5130	5132	5131	5131.0
10 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.000195	0.00098	0.00098	0.002145	5129	5133	5133	5131.7
alto/bajo/alto	0.00039	0.00059	0.00078	0.001755	5130	5131	5132	5131.0
alto/bajo/bajo	0.000585	0.00078	0.0002	0.00156	5131	5132	5129	5130.7
bajo/alto/alto	0.000585	0.00039	0.00098	0.00195	5131	5130	5133	5131.3
bajo/alto/bajo	0.000585	0.00098	0.00059	0.002145	5131	5133	5131	5131.7
bajo/bajo/alto	0.00039	0.00078	0.00039	0.00156	5130	5132	5130	5130.7
0.5-0.5								
alto/alto/bajo	0.000585	0.00078	0.00039	0.001755	5131	5132	5130	5131.0
alto/bajo/alto	0.000585	0.00078	0.0002	0.00156	5131	5132	5129	5130.7
alto/bajo/bajo	0.000585	0.00059	0.00078	0.00195	5131	5131	5132	5131.3
bajo/alto/alto	0.00039	0.00059	0.00059	0.00156	5130	5131	5131	5130.7
bajo/alto/bajo	0.000585	0.00059	0.00078	0.00195	5131	5131	5132	5131.3
bajo/bajo/alto	0.000585	0.00078	0.00098	0.00234	5131	5132	5133	5132.0
0.75-0.25								
alto/alto/bajo	0.000585	0.00117	0.00039	0.002145	5131	5134	5130	5131.7
alto/bajo/alto	0.00117	0	0.00039	0.00156	5134	5128	5130	5130.7
alto/bajo/bajo	0.00078	0.00059	0.00059	0.00195	5132	5131	5131	5131.3
bajo/alto/alto	0.00117	0.00039	0.00059	0.002145	5134	5130	5131	5131.7
bajo/alto/bajo	0.000975	0.00078	0.00039	0.002145	5133	5132	5130	5131.7
bajo/bajo/alto	0.00078	0.00117	0.00117	0.00312	5132	5134	5134	5133.3
6 partículas								
0.25-0.75								
alto/alto/bajo	0.000585	0.00039	0.00059	0.00156	5131	5130	5131	5130.7
alto/bajo/alto	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
alto/bajo/bajo	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
bajo/alto/alto	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
bajo/alto/bajo	0.000195	0.00059	0.00039	0.00117	5129	5131	5130	5130.0
bajo/bajo/alto	0.000585	0.00059	0.00059	0.001755	5131	5131	5131	5131.0

0.5-0.5								
alto/alto/bajo	-0.01365	0.00078	0.00059	-0.01229	5058	5132	5131	5107.0
alto/bajo/alto	0.000975	0.00078	0.00039	0.002145	5133	5132	5130	5131.7
alto/bajo/bajo	0.000585	0.00039	0.00059	0.00156	5131	5130	5131	5130.7
bajo/alto/alto	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
bajo/alto/bajo	0.000585	0.00059	0.00078	0.00195	5131	5131	5132	5131.3
bajo/bajo/alto	0.000585	0.00039	0.00078	0.001755	5131	5130	5132	5131.0
0.75-0.25								
alto/alto/bajo	0.00078	0.00039	0.0002	0.001365	5132	5130	5129	5130.3
alto/bajo/alto	0.000585	0.00039	0.00078	0.001755	5131	5130	5132	5131.0
alto/bajo/bajo	0.000585	0.00039	0.00039	0.001365	5131	5130	5130	5130.3
bajo/alto/alto	0.00039	0.00039	0.0002	0.000975	5130	5130	5129	5129.7
bajo/alto/bajo	0.00039	0.00078	0.00039	0.00156	5130	5132	5130	5130.7
bajo/bajo/alto	0.00078	0.00039	0.00059	0.001755	5132	5130	5131	5131.0
9 partículas								
0.25-0.75								
alto/alto/bajo	0.00039	0.00059	0.0002	0.00117	5130	5131	5129	5130.0
alto/bajo/alto	0.00039	0.00039	0	0.00078	5130	5130	5128	5129.3
alto/bajo/bajo	0.000195	0.00059	0.00059	0.001365	5129	5131	5131	5130.3
bajo/alto/alto	0.00039	0.00039	0.0002	0.000975	5130	5130	5129	5129.7
bajo/alto/bajo	0.000195	0.0002	0.00039	0.00078	5129	5129	5130	5129.3
bajo/bajo/alto	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
0.5-0.5								
alto/alto/bajo	0.00078	0.00039	0.00059	0.001755	5132	5130	5131	5131.0
alto/bajo/alto	0.00039	0.00059	0.0002	0.00117	5130	5131	5129	5130.0
alto/bajo/bajo	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
bajo/alto/alto	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
bajo/alto/bajo	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
bajo/bajo/alto	0.00039	0.00039	0.00078	0.00156	5130	5130	5132	5130.7
0.75-0.25								
alto/alto/bajo	0.000585	0.00059	0.00059	0.001755	5131	5131	5131	5131.0
alto/bajo/alto	0.000585	0.00059	0.00059	0.001755	5131	5131	5131	5131.0
alto/bajo/bajo	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
bajo/alto/alto	0.000585	0.0002	0.0002	0.000975	5131	5129	5129	5129.7
bajo/alto/bajo	0.000195	0.00078	0.00059	0.00156	5129	5132	5131	5130.7
bajo/bajo/alto	0.000585	0.00039	0.00039	0.001365	5131	5130	5130	5130.3
15 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00078	0.0002	0.00039	0.001365	5132	5129	5130	5130.3
alto/bajo/alto	0.00039	0.0002	0.0002	0.00078	5130	5129	5129	5129.3
alto/bajo/bajo	0.000975	0.00039	0.00098	0.00234	5133	5130	5133	5132.0
bajo/alto/alto	0.000585	0.00039	0.00059	0.00156	5131	5130	5131	5130.7
bajo/alto/bajo	0.000975	0.00059	0.00059	0.002145	5133	5131	5131	5131.7
bajo/bajo/alto	0.00039	0.00059	0.00059	0.00156	5130	5131	5131	5130.7
0.5-0.5								
alto/alto/bajo	0.00039	0.00059	0.0002	0.00117	5130	5131	5129	5130.0
alto/bajo/alto	0.000975	0.00176	0.00078	0.00351	5133	5137	5132	5134.0
alto/bajo/bajo	0.00039	0.00039	0.00078	0.00156	5130	5130	5132	5130.7
bajo/alto/alto	0.000975	0.00059	0.00059	0.002145	5133	5131	5131	5131.7
bajo/alto/bajo	0.000585	0.0002	0.00039	0.00117	5131	5129	5130	5130.0
bajo/bajo/alto	0.00117	0.00059	0.00059	0.00234	5134	5131	5131	5132.0

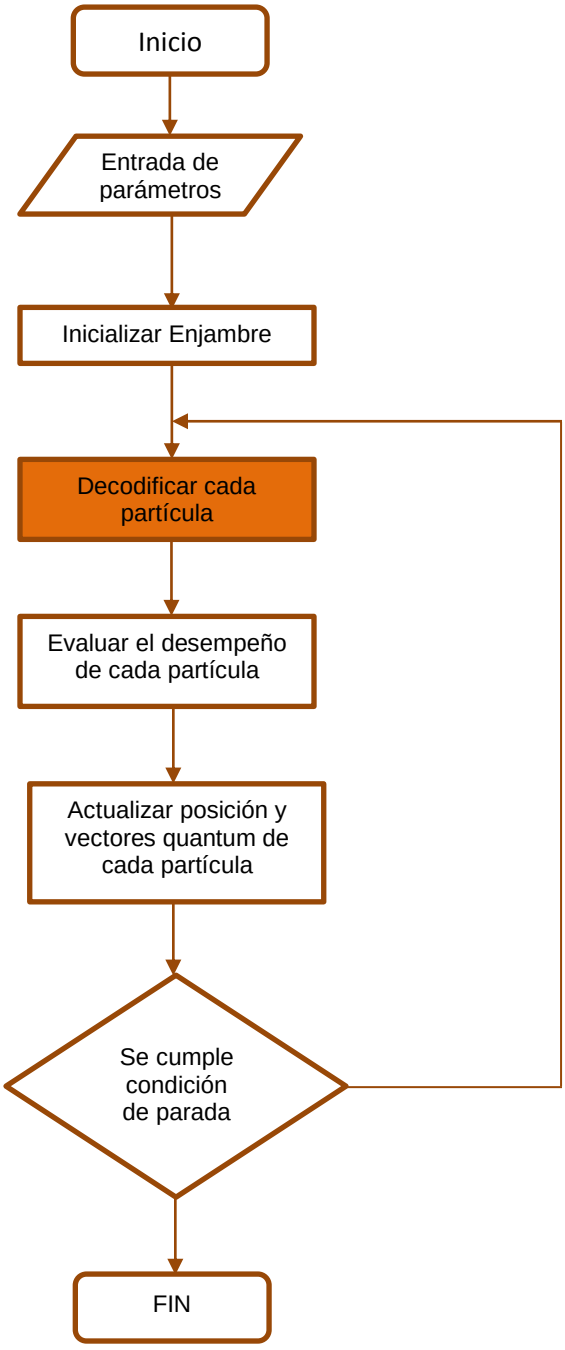
0.75-0.25								
alto/alto/bajo	0.000585	0.00059	0.00078	0.00195	5131	5131	5132	5131.3
alto/bajo/alto	0.000975	0.00078	0.00059	0.00234	5133	5132	5131	5132.0
alto/bajo/bajo	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
bajo/alto/alto	0.000585	0.00117	0.00039	0.002145	5131	5134	5130	5131.7
bajo/alto/bajo	0.000975	0.00059	0.00059	0.002145	5133	5131	5131	5131.7
bajo/bajo/alto	0.00078	0.00078	0.00059	0.002145	5132	5132	5131	5131.7
6 partículas								
0.25-0.75								
alto/alto/bajo	0.000195	0.00039	0.00059	0.00117	5129	5130	5131	5130.0
alto/bajo/alto	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
alto/bajo/bajo	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
bajo/alto/alto	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
bajo/alto/bajo	0.000975	0.00059	0.0002	0.001755	5133	5131	5129	5131.0
bajo/bajo/alto	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
0.5-0.5								
alto/alto/bajo	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
alto/bajo/alto	0.000195	0.00059	0.0002	0.000975	5129	5131	5129	5129.7
alto/bajo/bajo	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
bajo/alto/alto	0.00039	0.0002	0.0002	0.00078	5130	5129	5129	5129.3
bajo/alto/bajo	0.000585	0.00039	0.0002	0.00117	5131	5130	5129	5130.0
bajo/bajo/alto	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
0.75-0.25								
alto/alto/bajo	0.000585	0.00098	0.00059	0.002145	5131	5133	5131	5131.7
alto/bajo/alto	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
alto/bajo/bajo	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
bajo/alto/alto	0.000585	0.00059	0.00059	0.001755	5131	5131	5131	5131.0
bajo/alto/bajo	0.00039	0.00059	0.00059	0.00156	5130	5131	5131	5130.7
bajo/bajo/alto	0.000585	0.00078	0.00059	0.00195	5131	5132	5131	5131.3
9 partículas								
0.25-0.75								
alto/alto/bajo	0	0.00059	0.00059	0.00117	5128	5131	5131	5130.0
alto/bajo/alto	0.000195	0.00059	0.00039	0.00117	5129	5131	5130	5130.0
alto/bajo/bajo	0.000585	0.00039	0.00039	0.001365	5131	5130	5130	5130.3
bajo/alto/alto	0.00039	0.00098	0	0.001365	5130	5133	5128	5130.3
bajo/alto/bajo	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
bajo/bajo/alto	0.00039	0.0002	0.0002	0.00078	5130	5129	5129	5129.3
0.5-0.5								
alto/alto/bajo	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
alto/bajo/alto	0.000585	0.0002	0.00039	0.00117	5131	5129	5130	5130.0
alto/bajo/bajo	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
bajo/alto/alto	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
bajo/alto/bajo	0.000195	0.00039	0.0002	0.00078	5129	5130	5129	5129.3
bajo/bajo/alto	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
0.75-0.25								
alto/alto/bajo	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
alto/bajo/alto	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
alto/bajo/bajo	0.00039	0.0002	0.00039	0.000975	5130	5129	5130	5129.7
bajo/alto/alto	0.000585	0.00059	0.00059	0.001755	5131	5131	5131	5131.0
bajo/alto/bajo	0.00039	0.0002	0.00039	0.000975	5130	5129	5130	5129.7
bajo/bajo/alto	0.00039	0.00039	0.0002	0.000975	5130	5130	5129	5129.7

20 iteraciones								
3 partículas								
0.25-0.75								
alto/alto/bajo	0.00078	0.0002	0.00059	0.00156	5132	5129	5131	5130.7
alto/bajo/alto	0.000195	0.00059	0.0002	0.000975	5129	5131	5129	5129.7
alto/bajo/bajo	0.000585	0.0002	0.00039	0.00117	5131	5129	5130	5130.0
bajo/alto/alto	0.000195	0.00039	0.00078	0.001365	5129	5130	5132	5130.3
bajo/alto/bajo	0.000195	0.00039	0.0002	0.00078	5129	5130	5129	5129.3
bajo/bajo/alto	0.000585	0.00059	0.0002	0.001365	5131	5131	5129	5130.3
0.5-0.5								
alto/alto/bajo	0.000195	0.00039	0.00039	0.000975	5129	5130	5130	5129.7
alto/bajo/alto	0.000585	0.00059	0.0002	0.001365	5131	5131	5129	5130.3
alto/bajo/bajo	0.00078	0.00039	0.00059	0.001755	5132	5130	5131	5131.0
bajo/alto/alto	0.00078	0.00039	0.0002	0.001365	5132	5130	5129	5130.3
bajo/alto/bajo	0.000585	0.0002	0.00059	0.001365	5131	5129	5131	5130.3
bajo/bajo/alto	0.00078	0.00039	0.00039	0.00156	5132	5130	5130	5130.7
0.75-0.25								
alto/alto/bajo	0.00039	0.00059	0.00078	0.001755	5130	5131	5132	5131.0
alto/bajo/alto	0.00039	0.00078	0.00078	0.00195	5130	5132	5132	5131.3
alto/bajo/bajo	0.00039	0.00039	0.00078	0.00156	5130	5130	5132	5130.7
bajo/alto/alto	0.00039	0.00078	0.00078	0.00195	5130	5132	5132	5131.3
bajo/alto/bajo	0.00039	0.00059	0.0002	0.00117	5130	5131	5129	5130.0
bajo/bajo/alto	0.00039	0.00039	0.00117	0.00195	5130	5130	5134	5131.3
6 partículas								
0.25-0.75								
alto/alto/bajo	0.000195	0.0002	0.00039	0.00078	5129	5129	5130	5129.3
alto/bajo/alto	0.000585	0.00039	0.00039	0.001365	5131	5130	5130	5130.3
alto/bajo/bajo	0.00039	0.0002	0.00039	0.000975	5130	5129	5130	5129.7
bajo/alto/alto	0.00039	0.00039	0.0002	0.000975	5130	5130	5129	5129.7
bajo/alto/bajo	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
bajo/bajo/alto	0.000195	0.00039	0.0002	0.00078	5129	5130	5129	5129.3
0.5-0.5								
alto/alto/bajo	0.000195	0.0002	0.00039	0.00078	5129	5129	5130	5129.3
alto/bajo/alto	0.00039	0.00078	0.00039	0.00156	5130	5132	5130	5130.7
alto/bajo/bajo	0.000195	0.00039	0.0002	0.00078	5129	5130	5129	5129.3
bajo/alto/alto	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
bajo/alto/bajo	0.00039	0.00039	0.00059	0.001365	5130	5130	5131	5130.3
bajo/bajo/alto	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
0.75-0.25								
alto/alto/bajo	0.000585	0.00039	0.00039	0.001365	5131	5130	5130	5130.3
alto/bajo/alto	0.000585	0.00039	0.00059	0.00156	5131	5130	5131	5130.7
alto/bajo/bajo	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
bajo/alto/alto	0.000195	0.00059	0.00039	0.00117	5129	5131	5130	5130.0
bajo/alto/bajo	0.00039	0.0002	0.00059	0.00117	5130	5129	5131	5130.0
bajo/bajo/alto	0.00039	0.0002	0.00039	0.000975	5130	5129	5130	5129.7
9 partículas								
0.25-0.75								
alto/alto/bajo	0.00039	0	0.00039	0.00078	5130	5128	5130	5129.3
alto/bajo/alto	0.00039	0.0002	0.00039	0.000975	5130	5129	5130	5129.7
alto/bajo/bajo	0.000195	0.00039	0.00059	0.00117	5129	5130	5131	5130.0
bajo/alto/alto	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
bajo/alto/bajo	0.00039	0.00039	0.0002	0.000975	5130	5130	5129	5129.7
bajo/bajo/alto	0.00039	0.00039	0.0002	0.000975	5130	5130	5129	5129.7

0.5-0.5								
alto/alto/bajo	0.000195	0.00039	0.00039	0.000975	5129	5130	5130	5129.7
alto/bajo/alto	0.00039	0.0002	0.00059	0.00117	5130	5129	5131	5130.0
alto/bajo/bajo	0.000585	0.00059	0.00039	0.00156	5131	5131	5130	5130.7
bajo/alto/alto	0.00039	0.00039	0.00039	0.00117	5130	5130	5130	5130.0
bajo/alto/bajo	0.000585	0.00039	0.00039	0.001365	5131	5130	5130	5130.3
bajo/bajo/alto	0	0.00039	0.00059	0.000975	5128	5130	5131	5129.7
0.75-0.25								
alto/alto/bajo	0.00039	0.00059	0.00039	0.001365	5130	5131	5130	5130.3
alto/bajo/alto	0.000195	0.0002	0.00059	0.000975	5129	5129	5131	5129.7
alto/bajo/bajo	0	0.00059	0.00039	0.000975	5128	5131	5130	5129.7
bajo/alto/alto	0.00039	0.0002	0.00039	0.000975	5130	5129	5130	5129.7
bajo/alto/bajo	0.000195	0.00039	0.00039	0.000975	5129	5130	5130	5129.7
bajo/bajo/alto	0.00039	0.0002	0.00039	0.000975	5130	5129	5130	5129.7

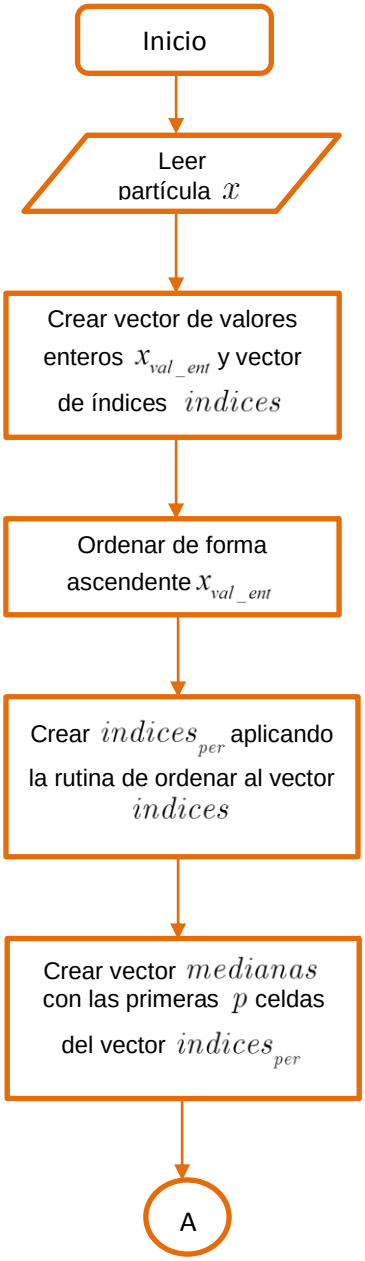
Apéndice E. Diagramas de Flujo

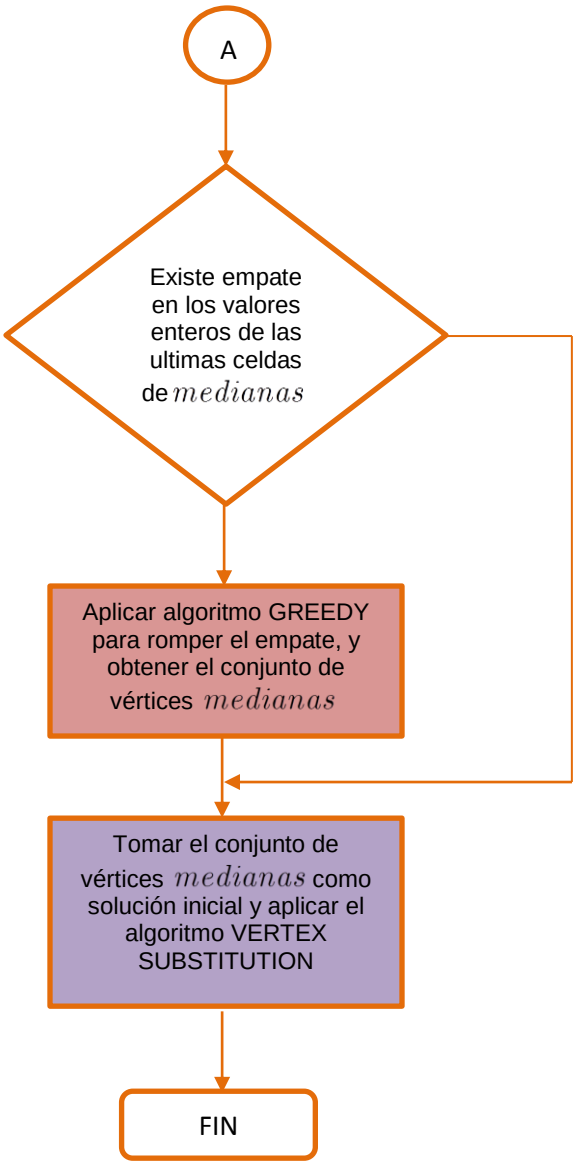
1. Diagrama de Flujo de Quantum Particle Swarm Optimization



Fuente: Autor

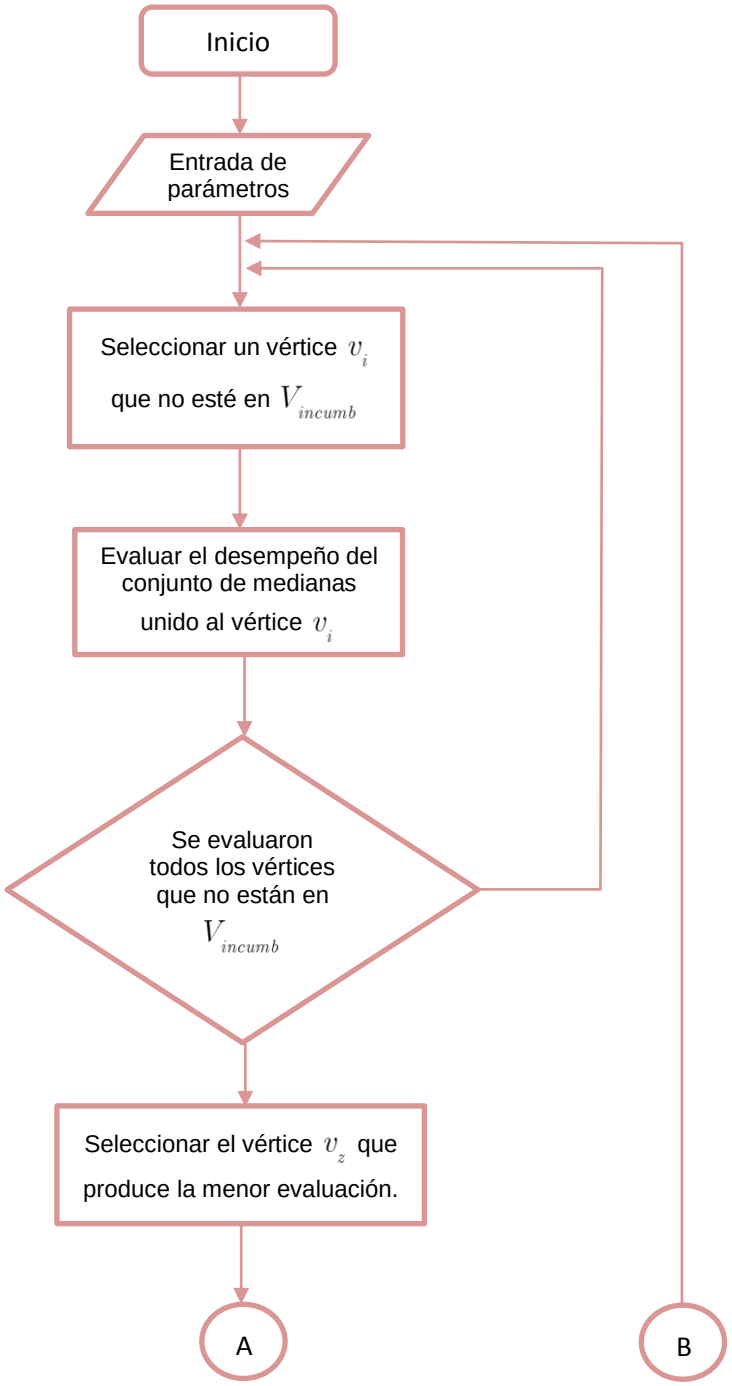
2. Diagrama de Flujo del Método de Decodificación

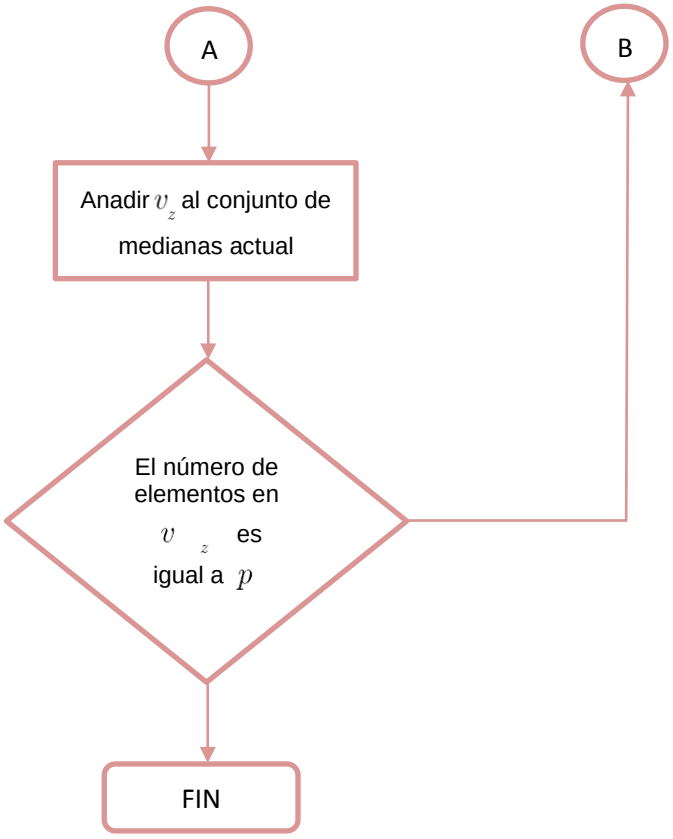




Fuente: Autor

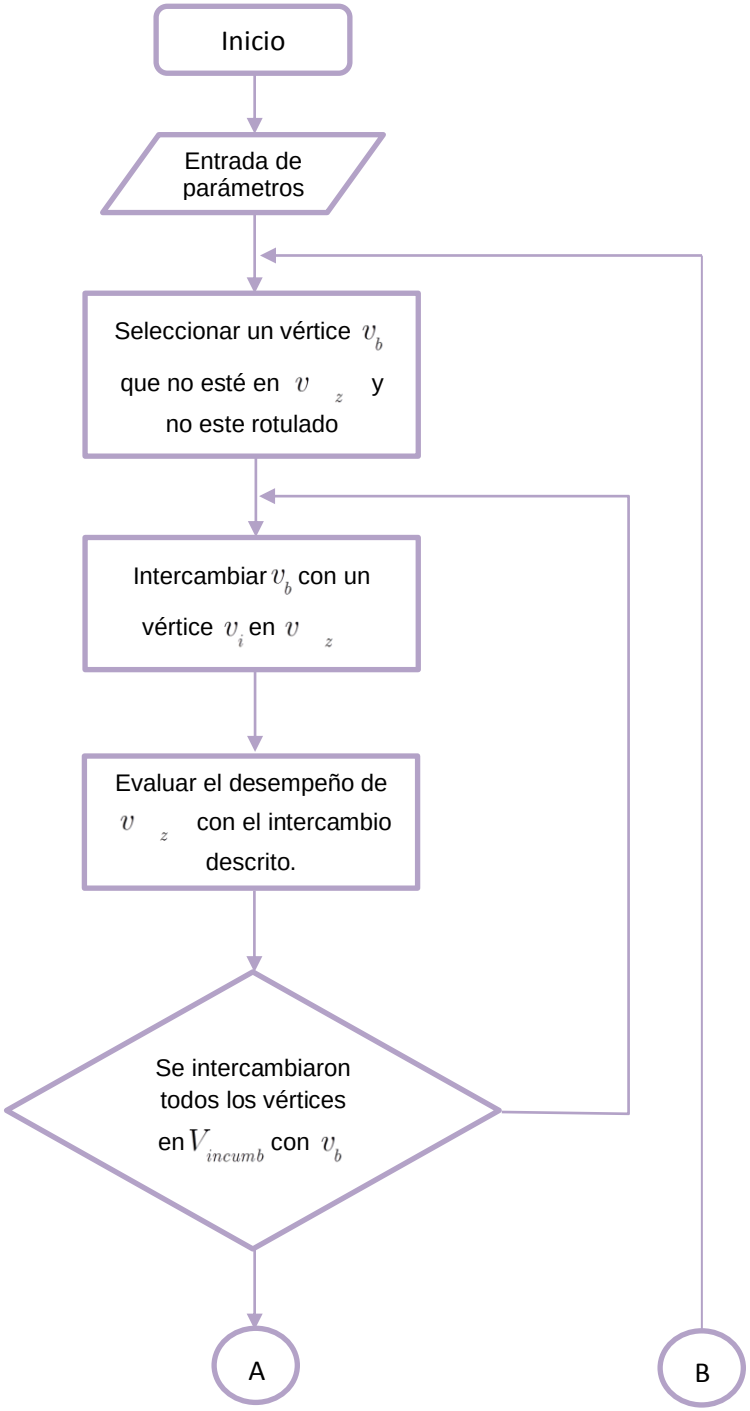
3. Diagrama de Flujo del Algoritmo Greedy

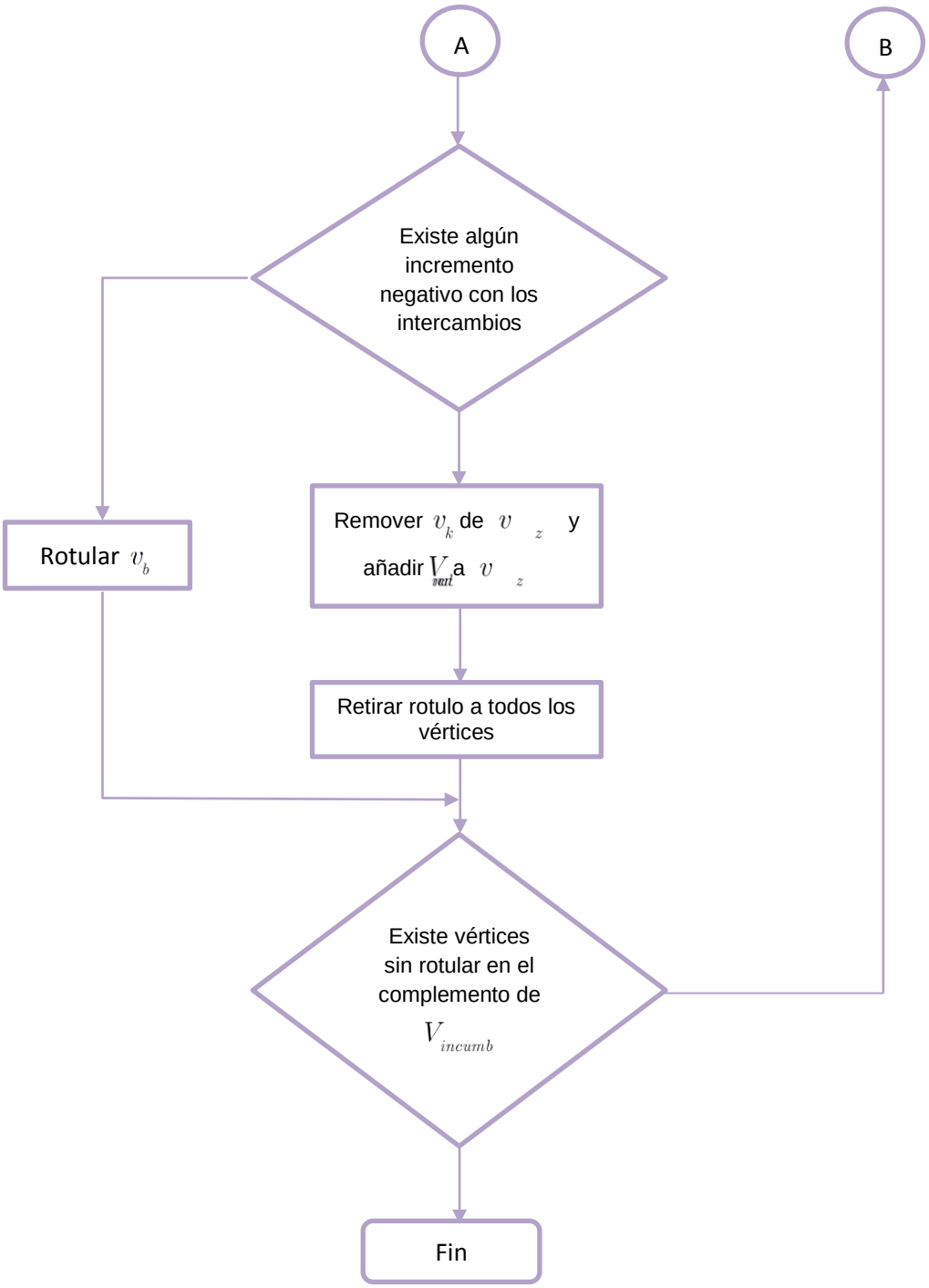




Fuente: Autor

4. Diagrama de Flujo del Algoritmo Vertex Substitution





Fuente: Autor