Uso de algoritmo secuencial de Colonia de Hormigas y Diferenciación Evolutiva para optimización de la distribución de materiales en almacén

T E S I S

Maestría en Ciencias en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Por

Ing. José Ivanhoe Vélez Herrera

Diciembre 2009

Uso de algoritmo secuencial de Colonia de Hormigas y Diferenciación Evolutiva para optimización de la distribución de materiales en almacén

TESIS

Maestría en Ciencias en

Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Por

Ing. José Ivanhoe Vélez Herrera

Diciembre 2009

Uso de algoritmo secuencial de Colonia de Hormigas y Diferenciación Evolutiva para optimización de la distribución de materiales en almacén

Por

Ing. José Ivanhoe Vélez Herrera

TESIS

Presentada a la División de Mecatrónica y Tecnologías de Información

Este trabajo es requisito parcial para obtener el grado académico de Maestro en Ciencias en Sistemas Inteligentes

Instituto Tecnológico y de Estudios Superiores de Monterrey

Campus Monterrey

Monterrey, N.L. Diciembre de 2009

Dedicado a quienes me esperaron.

Reconocimientos

José Ivanhoe Vélez Herrera

Instituto Tecnológico y de Estudios Superiores de Monterrey
Diciembre 2009

Uso de algoritmo secuencial de Colonia de Hormigas y Diferenciación Evolutiva para optimización de la distribución de materiales en almacén

José Ivanhoe Vélez Herrera, M.C.
Instituto Tecnológico y de Estudios Superiores de Monterrey, 2009

Asesor de la tesis: Dr. Horacio Martínez Alfaro

Un almacén eficiente es valioso para casi cualquier contexto industrial actual, donde cualquier ventaja competitiva en tiempo de entrega o en costos debe ser aprovechada. Uno de los problemas importantes para un almacén eficiente, es el acomodar los materiales de manera que la mayor parte del tiempo sean rápidamente accesibles y que sean localizados con facilidad. Esta tesis trata del problema de distribución de materiales de un almacén (WLP por sus siglas en inglés) que consiste en encontrar un plan de acomodo adecuado, de modo que la forma en que se introducen y sacan los materiales minimice el costo de transportación; esto se hará aplicando las técnicas de optimización de Colonias de Hormigas y Diferenciación Evolutiva.

Se utilizó como base para la técnica de optimización por Colonia de Hormigas la Optimización Max-Min, que dado que el modelo maneja cierto grado de aleatoriedad en todo momento, fue adaptada al problema agregándole una variable de holgura, que permite que el algoritmo siga trabajando una cantidad arbitraria de tiempo aunque aparentemente se haya alcanzado una convergencia.

La técnica de Diferenciación Evolutiva se utiliza como una extensión de la Colonia de Hormigas, donde la representación de esta se utiliza como población de la primera. Se manejan diversas formas de efectuar la selección para la Diferenciación Evolutiva. Así mismo se hacen pruebas con un modelo híbrido.

Durante la investigación se encontró que la técnica de Colonia de Hormigas funcionó más eficientemente que la Diferenciación Evolutiva durante las pruebas. El modelo híbrido funcionó tan bien cómo la Colonia de Hormigas, pero convergiendo en menor número de generaciones.

Índice general

Índice de tablas

Índice de figuras

Lista de Algoritmos

Capítulo 1

Introducción

La optimización de almacén es un problema que se ha atacado desde diversas perspectivas y con diferentes objetivos. Ha sido estudiado por la Investigación de Operaciones y está dividido en muchas partes, desde la cantidad de invetario que se debe tener para minimizar costos hasta el volumen ocupado por sus productos. La forma en que se define un almacén óptimo varía dependiendo del giro y cultura de la empresa, puede ser una propuesta humana de mantener los materiales clasificados y en orden, también se ha permitido que un software se encargue de dar seguimiento a la ubicación de materiales sin que tengan que estar ubicados de maneras obvias, incluso se ha propuesto mantener sólo un almacén de producto en proceso, con el famoso sistema *Just-in Time*. Una de las formas más difundidas de organizar el almacén es el sistema de Costeo Basado en Actividad, en la que este documento se basará.

Esta tesis trata del problema de distribución de materiales de un almacén (WLP por sus siglas en inglés) que consiste en encontrar un plan de acomodo adecuado, de modo que la forma en que se introducen y sacan los materiales minimice el costo de transportación. Otra forma de ver el problema es minimizar la suma de las distancias desde dos nodos de un grafo hacia el resto de los nodos cambiando el peso de cada nodo mientras se satisfacen determinadas restricciones. Esto se hará aplicando a un modelo de WLP las técnicas de optimización de Colonias de Hormigas y Diferenciación Evolutiva.

Un algoritmo de optimización por colonia de hormigas es una meta heurística inspirada en el comportamiento de hormigas reales, que encuentran la ruta más corta hacia su alimento basada en una comunicación indirecta entre ellas por medio de rastros de olor llamados caminos de feromonas.

El algoritmo de Diferenciación Evolutiva (DE) puede ser visto como un algoritmo genético de estado estable (es decir, que parte de la generación actual se mantiene en la generación siguiente), donde cada nuevo individuo puede ser parte o no de la siguiente generación. Básicamente, DE suma la diferencia ponderada entre dos vectores de una población a un tercer vector diferente de esa misma generación, de este modo, no se necesita utilizar una distribución de probabilidades separada, lo que lo convierte en un esquema completamente auto-organizable.

1.1. Definición del Problema

En la actualidad, el acomodo de materiales aún se realiza de manera artesanal o con heurísticas sencillas, y sigue habiendo problemas causados por la dificultad de alcanzar la optimalidad en la organización del almacén. Un aspecto importante del diseño de un almacén es su distribución de materiales. Una buena distribución de materiales de un almacén requiere tomar muchas decisiones de diseño con problemas combinatorios difíciles de resolver óptimamente. También se deben considerar operaciones como la obtención de material, el transporte dentro del propio almacén, servicios de valor agregado, así como factores como la demanda, las características físicas de los materiales y cargas, el servicio a mercados globales, el manejo de material con su costo, impacto de viaje por seguir filosofías *Just-in-Time* (JIT), entre otros. Todas estas operaciones y factores interactúan, y estas interacciones deben ser consideradas para el diseño [16]. Para este análisis teórico se asumirán las siguientes características:

- La base de cada material tiene un tamaño fijo (tarima).
- Habrá un límite de materiales apilados o estiba máxima independiente del tipo de material.
- El acomodo será de tipo tradicional (sin estantes), con pasillos de anchura fija para uso de montacargas, o de haberlos, no es área que no se intentará optimizar.
- Se considera el acomodo de áreas especiales con restricciones propias para proponer también áreas que no son de almacén, como las oficinas o áreas de inspección.
- Se maneja un sistema de Primeras Entradas, Primeras Salidas (PEPS) para el movimiento de material.
- El almacén tiene un sólo piso, una entrada y una salida (que pueden o no estar en el mismo sitio).

1.2. Motivación

Cada parte de este proyecto fue motivado como unión entre mi carrera e intereses independientes. El problema de optimización de almacén fue elegido para investigar perspectivas teóricas alternativas a las estudiadas durante la carrera de Ingeniería Industrial y de Sistemas. El manejo correcto de almacén es un problema con muchos modelos de solución diferentes, algunos de ellos buscan utilidad cualitativa, donde es más importante mantener una organización fácil de seguir que disminuir los costos de mantener el almacén. Otros modelos dan preferencia a permitir la salida rápida del

material mientras se respeta un sistema de movimiento de Primeras Entradas, Primeras Salidas. En este caso, el acomodo de materiales se maneja como un problema de combinatoria usando como función objetivo una representación matemática de lo que se busca con algunos modelos utilizados de manera artesanal.

La técnica que se decidió utilizar para atacar el problema fue un algoritmo de Optimización por Colonia de Hormigas (ACO por sus siglas en inglés). El hecho que la técnica implica en su base el recorrido y minimización de distancias, fue a primera vista el método que parecía más adecuado para el tipo de problema. Una vez que se investigó más de la técnica, demostró ser una metaheurística robusta y con una aplicación relativamente directa con una representación gráfica del problema. Durante la búsqueda de más técnicas para resolver problemas de optimización, la manera en que fue visualizada la Diferenciación Evolutiva para ser aplicada al problema fue utilizando una extensión de la forma en que éste es representado para algoritmos de Colonia de Hormigas, y por lo tanto, podría ser utilizado en conjunto con éste, en especial, como alternativa al reinicio del algoritmo de Sistema de Hormigas Max-Min como es propuesto en la literatura.

1.3. Objetivos

Los objetivos principales del modelo serán minimizar movimiento de montacargas y satisfacer restricciones de adyacencia. Para esta tesis se modelará una situación propuesta por la literatura y se trabajará con dos sistemas de optimización tanto de manera conjunta como independiente para resolverla. La primer meta-heurística que se utilizará será de un Algoritmo de Optimización por Colonia de Hormigas (ACO por sus siglas en inglés), específicamente el Sistema de Hormigas Max-Min (ACO-MMAS o símplemente MMAS por sus siglas en inglés); la segunda meta-heurística será la Diferenciación Evolutiva (DE), que se encargará de generar variedad para mejorar el rendimiento de la primera.

Un algoritmo de optimización por colonia de hormigas es una meta heurística basada en un modelo de probabilidades parametrizado a través del espacio de solución llamado modelo de feromonas. Las soluciones construidas, y posiblemente las soluciones que fueron construidas en iteraciones anteriores son usadas para modificar los valores de las feromonas de manera que genera la tendencia de llevar futuras muestras hacia soluciones de alta calidad. Esto se hace dándole más probabilidades a que se construyan partes de solución que en el pasado resultaron exitosas. El concepto de que la representación del camino de feromonas tenga la oportunidad de abarcar cualquier solución del espacio de búsqueda en cada iteración, lo vuelve un algoritmo que mantiene la exploración en todo momento y que dada una cantidad ilimitada de tiempo, garantiza encontrar el óptimo de un problema.

La forma en que la colonia de hormigas focaliza la búsqueda es aumentando las probabilidades de que partes de la solución se asemejen cada vez más a buenas soluciones encontradas en el pasado. La forma en que esta representación de la solución se va modificando a través del tiempo ha sido mejorada desde su concepción original, creando diversas técnicas de colonias de hormigas como lo es el *Ant System* o el *Max-Min Ant System*, sin embargo todas tienen en común una representación de feromonas que se va modificando gradualmente conforme el algoritmo avanza. En el presente documento se propone una nueva forma en la que se modifica el camino de feromonas utilizando una población de rastros de feromona en vez de uno solo, usando un algoritmo evolutivo para valores continuos. Dada la naturaleza requerida para esta aproximación, se tomó como punto de partida la forma básica del algoritmo de Diferencial Evolutivo (DE).

La forma en que se atacará el problema será utilizando un camino de feromonas en el caso de ACO-MMAS o una población de estos caminos cuando se trabaje con DE.

1.4. Hipótesis

Un algoritmo de Sistema de Hormigas Min-Max (MMAS) permite obtener buenas propuestas para el modelo de Optimización de Distribución de Materiales de Almacén, y alternarlo con un algoritmo de Diferenciación Evolutiva (DE) obtiene mejores resultados que el MMAS por si solo.

Se harán las siguientes preguntas de investigación:

- ¿Cuál es la forma más adecuada para representar las meta-heurísticas en un algoritmo ACO?
- ¿Es factible modelar el problema para un algoritmo ACO de manera que se obtengan buenos resultados?
- ¿Qué datos deben ser considerados por el algoritmo como restricciones o como parte de la función objetivo? ¿Cómo se representarán?
- ¿Mejora de alguna forma el secuencializar el algoritmo ACO con uno DE?

1.4.1. Justificación

El modelo propuesto si bien simplificado, sigue las reglas generales de lo que se supone debe ser un almacén con clasificación ABC, que es una clasificación que utiliza la regla de Pareto y fue propuesto en [11], además, permite flexibilidad en cuanto a la ubicación de materiales del mismo tipo aunque permite que estos estén separados. El uso de la Diferenciación Evolutiva no tiene aplicación directa apreciable a este modelo,

ya que al usar como referencia una matriz de ubicación donde cada valor es de tipo cualitativo, las operaciones aritméticas para las mutaciones, que son para variables cuantitativas continuas, no tienen sentido. Sin embargo, el modelo generado por el trayecto de feromonas si es aplicable a la Diferenciación Evolutiva, lo que permite agregar una dimensión más al trabajo que realiza un algoritmo de colonia de hormigas.

1.5. Contribución

Esta investigación propone una forma de combinar modelos de optimización utilizando el esquema de Optimización por Colonia de Hormigas como población de un esquema de Diferenciación Evolutiva para el problema de Optimización de Acomodo de Materiales de Almacén. Con esto se hacen las siguientes aportaciones:

- Se analiza el comportamiento de un algoritmo de Diferenciación Evolutiva donde la evaluación de cada individuo de la población es estadística, por lo que cada individuo requiere evaluarse varias veces para reducir un engaño producido por la aleatoriedad.
- Propone una variación al algoritmo de Sistema de Hormigas Max-Min para permitir reinicios más adecuados cuando la generación de propuestas de solución no cumple las probabilidades del camino de feromona para mantener la factibilidad.
- Propone un sistema secuencial para aplicar los algoritmos de Diferenciación Evolutiva y Colonia de Hormigas de manera secuencial. Para el modelo utilizado de Optimización de Acomodo de Materiales de Almacén funcionó mejor que los algoritmos por separado.

1.6. Organización de la Tesis

A continuación se presenta una descripción general de la organización de la presente tesis:

En el capítulo 2 se dará una breve reseña teórica acerca de los diferentes modelos del almacén y lo que se espera de un buen almacén, se expondrá parte de la teoría necesaria para seguir la codificación de un algoritmo de Optimización por Colonia de Hormigas, en especial, el Sistema de Hormigas Max-Min en el modelo Hipercúbico. Finalmente se hará una breve reseña de lo que es la Diferenciación Evolutiva y sus parámetros.

En el capítulo 3, se expondrá detalladamente la forma en que el problema fue codificado y sus soluciones evaluadas. En seguida, se explicará con su respectivo pseudocódigo, la forma en que el modelo de Colonia de Hormigas y Diferenciación Evolutiva generan las soluciones con base en las variables dadas.

En el capítulo 4 se expondrán los experimentos hechos para comparar las técnicas y describir su comportamiento.

Capítulo 2

Antecedentes

En este Capítulo se presenta la teoría que sustenta este trabajo de tesis. Los temas presentados en este Capítulo son necesarios para la comprensión de la investigación y sus implicaciones. Primero se describe lo que es la optimización. En seguida se presenta una reseña sobre los algoritmos de colonias de hormigas, especialmente el Sistema de Hormigas Max-Min en un modelo Hipercúbico. Posteriormente se describirán las bases de la Diferenciación Evolutiva.

2.1. Optimización

Los problemas de optimización global a través de espacios continuos, son una tarea que consiste en encontrar el mínimo (o máximo) a ciertas propiedades de un sistema eligiendo un conjunto de parámetros, que comúnmente es representado como un vector [17]. La forma en que se atacan estos problemas normalmente consta de una función objetivo que permite modelar el problema mientras se le incorporan restricciones. Típicamente esta función objetivo consiste en la minimización de una suma ponderada de los valores que pueden obtener los vectores:

$$z(\vec{x}) = \sum_{m=0}^{p} w_m f_m(\vec{x}) \tag{2.1}$$

donde f_m es una de las p propiedades de valor real del sistema representadas en forma de vectores $\mathbf{x}$, y w es el peso de la propiedad con respecto a la evaluación el sistema [17].

Cuando la función objetivo es no lineal y no diferenciable, la forma en que se intenta resolver es por métodos de búsqueda, y siendo entre los más utilizados, algoritmos evolutivos. La principal característica de los métodos de búsqueda directa es generar variaciones a los vectores parámetro. Una vez que se genera la variación, se debe decidir si se acepta o no el parámetro modificado utilizando un criterio avaricioso (*greedy*), es decir, sólo se acepta el nuevo parámetro si de alguna manera éste mejora el valor anterior de acuerdo a la función objetivo; sin embargo, un método que se comporte de una manera demasiado *greedy* corre el riesgo de llegar a una convergencia demasiado

rápido y quedar atrapado en un mínimo local. Las búsquedas inherentemente paralelas, como aquellas que utilicen población, pueden ayudar a evitar una convergencia prematura; utilizando varios vectores simultaneamente, se puede ayudar a encontrar configuraciones de parámetros mejores que ayuden a escapar de mínimos locales.

Lo que se espera de las técnicas de optimización es que se encuentre el óptimo independientemente de los parámetros iniciales, que lo haga en tiempo aceptable y, que tenga pocos parámetros de control, por lo que sea fácil de usar. En este trabajo se utilizan dos técnicas de optimización: la Diferenciación Evolutiva (DE) y la Optimización por Colonia de Hormigas MAX-MIN (ACO-MMAS). El primero es una variación de los algoritmos genéticos aplicados a valores continuos que surgió como una mejora al recocido simulado utilizando una población de vectores, y que ha demostrado tener buenas propiedades de convergencia a pesar de requerir pocas variables y ser bastante simple. El algoritmo ACO-MMAS utiliza una población de agentes muy simples que por medio de una comunicación indirecta modifica una guía estadística para generar soluciones, siendo el hecho de que las soluciones se proponen de manera estocástica lo que ayuda a evitar que el algoritmo se estanque en un mínimo local. Durante las siguientes secciones se hablará a profundidad sobre estas técnicas.

2.2. Optimización por Colonias de Hormigas

La optimización por Colonia de Hormigas (ACO) es una metaheurística en la que una colonia de hormigas artificiales cooperan para encontrar buenas soluciones a problemas difíciles de optimización discreta. De este modo, asignando recursos a un grupo de agentes simples que se comunican indirectamente por medio del ambiente, surgen buenas soluciones por su interacción cooperativa [20]. Esta meta-heurística emula el comportamiento de las colonias de hormigas que buscan la ruta más corta entre su nido y la fuente de comida. Los principales iniciadores de esta técnica fueron Dorigo y DiCaro [13], y se ha conseguido aplicar a variadas situaciones, aunque su aplicación más típica es al problema del vendedor viajero.

Las hormigas artificiales en un algoritmo ACO son procedimientos de construcción estocástica que incrementalmente construyen una solución agregando componentes de solución a una solución parcial. Lo que principalmente compone a un algoritmo ACO es la *Construcción de Soluciones*, la *Actualización de Feromonas* y las *Acciones Independientes*:

- **Construcción de Soluciones**: maneja una colonia de hormigas que asincrónicamente visitan estados adyacentes del problema moviéndose hacia nodos vecinos del grafo de la construcción del problema. Se mueven aplicando una política de decisión local estocástica usando el camino de feromonas e información heurística. Con esto, las hormigas incrementalmente construyen la solución al problema

de optimización. Una vez que una hormiga ha construido una solución, la hormiga evalúa la solución que será usada para que el procedimiento *Actualización de Feromonas* decida cuanta feromona se depositará.

- **Actualización de Feromonas**: es el proceso mediante el cual los caminos de feromona se modifican. Los valores de los caminos se pueden incrementar conforme las hormigas depositan la feromona, o pueden disminuir por su evaporación. El depósito de nueva feromona aumenta la probabilidad que los componentes que fueron usados por muchas hormigas, o por una sola con una muy buena solución, vuelvan a ser usados otra vez por hormigas en el futuro; mientras que la evaporación es una forma útil de olvidar, es decir, de permitir que soluciones que ya no han visitado pierdan gradualmente probabilidades de que vuelvan a ser utilizadas, de este modo, se evita una convergencia demasiado rápida hacia regiones subóptimas, y por lo tanto, favorece la exploración de nuevas áreas en el espacio de búsqueda.

- **Acciones Independientes**: son procedimientos utilizados para implementar acciones centralizadas que no pueden ser hechas por una sola hormiga. Un ejemplo de acción independiente será la aplicación de un algoritmo de Diferenciación Evolutiva una vez que el algoritmo se haya quedado estancado.

El primer algoritmo de ACO fue el Sistema de Hormigas (AS), que fue aplicado para pequeñas instancias del problema de agente viajero con resultados satisfactorios. Desde entonces, se han propuesto varias mejoras para el AS. En [13] se propuso una estrategia elitista, que está basada en agregar refuerzo adicional al camino de feromona producido por el mejor viaje encontrado, mediante la generación de hormigas elitistas que sólo siguen las rutas del mejor viaje hasta el momento. Otra mejora es el Sistema de Colonia de Hormigas (ACS) [21], que se diferencia del AS principalmente por la regla de elección para la siguiente ciudad, pues tiene un esquema de actualización local en la que sólo la hormiga con la mejor evaluación tiene permitido actualizar su nivel de feromona. También está el Sistema de Hormigas basado en rangos (ASrank) [20], en la que, aparte de las hormigas elitistas, otras hormigas pueden dejar rastro de feromona dependiendo del rango que tienen basado en qué tan cercana estuvo su evaluación con respecto al mejor encontrado. En general, las mejoras hechas al AS consisten en formas prometedoras de agregarle avaricia al algoritmo, sin embargo, se debe tener cuidado con estancamientos tempranos en óptimos locales. De manera interesante, las mejores soluciones son encontradas cuando el algoritmo está muy cerca de estar estancado, pero al estarlo, deja de haber exploración, eliminando la posibilidad de mejorar aún más. Una mejora al AS que está hecha para ser avaricioso al mismo tiempo que promueve la exploración en todo momento es el Sistema de Hormigas MAX-MIN (MMAS), que será visto con mayor profundidad a continuación.

2.2.1. Sistema de hormigas MAX-MIN

El Sistema de Hormigas MAX-MIN (MMAS) [26] introduce cuatro modificaciones importantes con respecto al Sistema de Hormigas básico:

- Usa estrategia elitista, es decir, le permite sólo depositar feromona a la mejor hormiga de la iteración (i_{bs}) o al que ha encontrado la mejor solución (bs). La desventaja de hacer esto es que esta estrategia puede llevar a producir estancamiento por un crecimiento excesivo del camino de feromona; para contrarrestar este efecto, surgieron las siguientes modificaciones.

- La cantidad de feromona que puede tener un camino está limitada por el intervalo $[\tau_{MIN},\tau_{MAX}]$.

- Se reinician los caminos de feromona cada vez que el algoritmo se aproxima a una convergencia o no se han obtenido mejores soluciones por un cierto número de iteraciones consecutivas.

- En el algoritmo MMAS original, todos los caminos de feromonas son inicializados en τ_{MAX}, para que de este modo, aunado a una evaporación lenta, toda la fase inicial del algoritmo esté fuertemente enfocado a la exploración. Esto fue modificado para el modelo hipercúbico del algoritmo, y por lo tanto, para la aplicación de este proyecto como será visto en la siguiente sección.

Una vez que todas las hormigas han construido todo el viaje, se actualizan las feromonas aplicando la evaporación como en Sistema de Hormigas básico, para que después, el bs o el i_{bs} dejen su feromona; en general, en las implementaciones de MMAS ambas formas de dejar feromona son utilizadas de manera alternada, mientras más veces sea dejada por el bs, el algoritmo se enfoca más rápidamente sobre una solución, mientras que permitiendo que la feromona sea depositada por el i_{bs}, es la exploración la que tiene la preferencia. Típicamente, la actualización de feromonas se hace utlizando la siguiente fórmula:

$$\tau_{ij}(t) = \rho\tau_{ij}(t-1) + \Delta\tau_{ij}^{best} \tag{2.2}$$

donde $\Delta\tau_{ij}^{best}$ es la cantidad de feromona en base a su evaluación que agregará la hormiga encargada de actualizar el rastro y ρ es la evaporación del rastro de feromona. Los límites τ_{MAX} y τ_{MIN} son impuestos para evitar que el algoritmo se estanque, estos tienen el efecto de limitar la probabilidad ρ_{ij} de seleccionar el nodo j cuando una hormiga está en el nodo i al intervalo $[\rho_{MIN}, \rho_{MAX}]$, de este modo se asegura que todos los caminos tendrán siempre una probabilidad significativamente mayor que cero para que una hormiga pase por ellos, y del mismo modo, que todos los caminos tengan una probabilidad menor a uno de que sean considerados para la construcción de la solución. El MMAS ha sido un algoritmo ACO muy estudiado y al que se le han

realizado diversas mejoras, la variante que será utilizada en este documento será la del Modelo Hipercúbico.

2.2.2. Modelo Hipercúbico de Optimización por Colonia de Hormigas

El modelo Hipercúbico de Optimización por Colonia de Hormigas [6] (HCF por sus siglas en inglés) está basado en el cambio de la regla de actualización de feromonas utilizada en los algoritmos ACO, de manera que el rango de valores del camino de feromonas esté limitado al intervalo $[0, 1]$. Este modelo fue concebido como una forma de abolir la propiedad no deseable de los algoritmos ACO de cambiar su rendimiento dependiendo de la escala del problema. Adicionalmente, el HCF vuelve algunos aspectos del análisis teórico de algoritmos ACO más fácil y los vuelve más robustos. El camino de feromonas puede ser visto como un vector τ $|C|$-dimensional. Aplicando una actualización de feromona al final de cada actualización se cambia el vector, que se mueve en un hiperespacio $|C|$-dimensional definido por los límites mínimos y máximos del rango de valores que los parámetros del camino de feromonas puede asumir. A este espacio se le denomina $\mathcal{H}\tau$. La diferencia entre esta actualización de feromona y la estándar es la multiplicación de la cantidad agregada de feromona por ρ y la normalización de la feromona agregada.

Para dar una interpretación gráfica a la regla de actualización en el HCF, consideramos una solución s al problema desde otra perspectiva. Con respecto a la solución $s \in S$, particionamos el conjunto de componentes de solución C en dos subconjuntos, el conjunto C_{in} que contiene todos los componentes de solución $c_{i,x_i} \in s$, y $C_{out} = C/C_{in}$. De este modo, podemos asociar a una solución s un vector binario $\mathbf{s}$ de dimensión $|C|$, donde la posición correspondiente al componente de solución c_i,x_i es 1 si $c_{i,c_i,c_i} \in C_{in}$, y 0 de lo contrario. Esto significa que podemos considerar una solución s como una esquina de un hipercubo $|C|$-dimensional. Por lo tanto, el conjunto de soluciones factibles S puede ser considerado como un subconjunto de esquinas del cubo $|C|$-dimensional. De modo que

$$\overrightarrow{\tau} \leftarrow (1 - \rho)\tau + \rho\mathbf{m} \tag{2.3}$$

donde $\mathbf{m}$ es un vector $|C|$-dimensional con

$$\mathbf{m} = \sum_{s \in S_{upd}} \gamma_s \mathbf{s} \tag{2.4}$$

y

$$\gamma_s = \frac{F\ (s)}{\sum_{s' \in S_{upd}} F\ (s')} \tag{2.5}$$

El vector $\mathbf{m}$ es un vector en s, la envolvente convexa de s, es

$$\sum_{s \in S_{upd}} \gamma_s = 1$$

y

$$0 \leq \lambda_s \leq 1 \forall s \in S_{upd}.$$

Mientras mayor sea la calidad $F(s)$ de una solución s, más alta será la influencia del vector $\mathbf{m}$. La regla de actualización de feromonas para MMAS en el HFS es:

$$\tau \leftarrow \tau + \rho(\mathbf{s_{upd}} - \tau) \tag{2.6}$$

Los algoritmos ACO tienen la propiedad de que los valores de feromonas pueden ser interpretados como probabilidades durante la corrida del algoritmo si son probabilidades desde el comienzo. El modelo hipercúbico aplicado a un algoritmo MMAS tiene sus propias ventajas, pues al ser el nivel de feromona similar a la probabilidad de construcción de solución, facilita la selección del τ_{MAX} y τ_{MIN}, volviéndose valores complementarios en vez de independientes; para esto, sólo se necesita elegir un valor menor pero cercano a 1 como máximo para evitar convergencia y se mantenga exploración cercana a la solución con más feromona, este valor será τ_{MAX}; para obtener τ_{MIN} sólo se reparte el complemento de τ_{MAX} entre las demás soluciones posibles. Otra ventaja de usar el HCF para un algoritmo MMAS es que en lugar de turnar el rastro de feromona dejado entre la mejor iteración y el mejor valor del reinicio, se puede hacer una interpolación lineal de modo que cada uno de ellos tenga un porcentaje conveniente de ambos tipos de evaluación.

2.3. Diferenciación Evolutiva

La diferenciación evolutiva es una rama de los algoritmos evolutivos desarrollados por Rainer Storn y Kenneth Prince para problemas de optimización sobre dominios continuos [22]. En un algoritmo de Diferenciación Evolutiva (DE), cada valor de una variable en el cromosoma es representada por un número real. La aproximación trabaja creando una población inicial aleatoria de posibles soluciones factibles. Un individuo es seleccionado para un remplazo aleatorio y tres individuos diferentes son seleccionados como padres. Uno de estos individuos es seleccionado como el padre principal. Con cierta probabilidad, cada variable en el padre principal es cambiada. El cambio es tomado agregando a cada valor de la variable la diferencia entre los dos valores de esta variable en los otros dos padres. En escencia, una fracción del vector padre principal es perturbado por los otros dos vectores. Desde el primer artículo que Storn publicón sobre la Diferenciación Evolutiva [22], a la selección de parámetros para la mutación se le relacionó con operación de cruce de los algoritmos genéticos [15], término que

será utilizado en este documento para indicar las operaciones que definen las partes del vector que serán mutadas. Si el valor resultante es mejor que el elegido para el remplazo, éste es remplazado, de lo contrario, el original se mantiene en la población.

Un algoritmo DE es un método de búsqueda en paralelo que utiliza p vectores de parámetros como población por cada generación G. La población inicial es elegida aleatoreamente y debería cubrir el espacio de búsqueda de manera uniforme, aunque para este proyecto se utilizaron métodos de inicialización alternativos para que funcione en conjunto con la técnica MMAS. Por cada vector $\mathbf{x}_{i,G}$, donde $i = 0, 1, 2, ..., p-1$, se genera un vector hijo $\mathbf{v}_{i,G+1}$ de acuerdo a $r_1, r_2, r_3 \in [0, p-1]$, que son enteros, diferentes entre ellos y una $F > 0$. Los enteros r son elegidos aleatoreamente, F es un factor constante $\in [0, 2]$ que controla la amplificación de la variación $(\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G})$. De este modo, por cada vector $\mathbf{x}_{r_i,G}$, un vector hijo es generado de acuerdo a

$$\mathbf{v} = \mathbf{x}_{r_i,G} + (\mathbf{x}_{r_2,G} - \mathbf{x}_{r_3,G}) \tag{2.7}$$

Para incrementar la diversidad de los vectores parámetros perturbados, la forma en que se elige las partes del vector que serán modificadas, se utiliza un proceso similar al del cruce de los algoritmos genéticos. Para esto, se crea el vector:

$$\mu_{i,G+1} = (\mu_{0i,G+1}, \mu_{1i,G+1}, ..., \mu_{(D-1)i,G+1}) \tag{2.8}$$

con

$$\overrightarrow{\mu}_{ji,G+1} = \begin{cases} \mathbf{v}_{ji,G+1}, & j = \langle n \rangle_D, \langle n+1 \rangle_D, ..., \langle n+L-1 \rangle_D \\ x_{ji,G} & delocontrario \end{cases} \tag{2.9}$$

donde los paréntesis del tipo $\langle\ \rangle_D$ son una función módulus con módulo D; n es un entero elegido de manera aleatorea en el intervalo $[0, D-1]$; el entero L es el número de parámetros que será cambiado, y es elegido del intervalo $[1, D]$.

Para decidir si el nuevo vector $\mathbf{u}$ se volverá miembro de la población en la generación $G+1$, se comparará con $\mathbf{x_{i,G}}$. Si el vector $\mathbf{u}$ es mejor que $\mathbf{x_{i,G}}$, entonces $\mathbf{x_{i,G+1}}$ será $\mathbf{u}$, de lo contrario, se mantendrá $\mathbf{x_{i,G}}$.

2.4. Modelos de almacén

Para que un almacén se le pueda considerar "óptimo" se las características que más comunmente se toman en cuenta son las siguientes [4]:

- Mínimos recorridos: La metodología más usada para clasificar el inventario se le llama ABC (Activity Based Costing) y consiste primeramente en obtener aquellos que forman el 80 % de la venta, a los que se les clasificará como artículos "A" al 20 % restante de los artículos se les puede reclasificar obteniendo de nuevo el 80 % para a estos clasificarlos como artículos "B" los cuáles representan el 16 %

de la venta (20 % * 80 %), y a los artículos restantes se les clasifica como artículos "C", los cuáles bajo este esquema representan el 4 % de la venta (20 % * 20 %) .

- Facilidad de acceso al stock: Modulación de ubicaciones.
- Facilidad de recuento.
- Minimizar: espacio, tráfico, movimientos, riesgos.
- La estructura e implantación deben ser flexibles para adaptarse a futuras necesidades.
- Aprovechamiento del espacio: Superficie y volumetría.
- Mínima manipulación: unidades de almacenaje, unidades de manipulación.
- Flexibilidad en la colocación: Espacios libres, previsión de espacios.
- Las cantidades almacenadas se deben calcular para que los costos que originen sean mínimos, siempre que se mantengan los niveles de servicio.

Los pasos generales para el diseño de un almacén [16] se enumeran en la tabla 2.1. Para el proyecto, se tomarán los resultados de los primeros cinco pasos como datos dados, así como restricciones acerca de los demás que servirán que el modelo pueda ofrecer propuestas factibles para el diseño del área de almacén. Dada la variedad de maneras de evaluar el correcto acomodo de materiales, así como el hecho de que aunque el espacio sea constante, el sistema de materiales es intrínsecamente dinámico, obtener el acomodo óptimo es un problema en el que si no se usan heurísticas, su complejidad crece en tiempo no polinomial. Los ingenieros industriales y los administradores de las fábricas muchas veces confían en la experiencia obtenida por prueba y error para tomar decisiones en la distribución de materiales de la planta y en los sistemas de manejo de materiales. Aunque la experiencia es valiosa, determinar los costos y beneficios de distribución de materiales alternativos cuya modificación es costosa requieren análisis matemático [1]. Algunos problemas parciales como la volumetría y el planteamiento de la metodología de ABC pueden recibir ayuda de software [19]. Por medio de meta-heurísticas se podrían aplicar en conjunto los métodos actuales para obtener mejores resultados de manera robusta y combinar soluciones de problemas parciales para obtener una aproximación al óptimo del problema completo.

Para esta tesis, no se considerará como factor a evaluar la volumetría o espacio ocupado por los materiales en el almacén. Si se quisiera modelar el problema de volumetría con un algoritmo computacional, no se le debe considerar sólo como un problema de empacado en tres dimensiones, dado que el área de los materiales es generalmente fija (igual al tamaño de la tarima) y que la altura está limitada por el peso o la estiba

1. Especificar el tipo y propósito del almacén.
2. Análisis y pronóstico de la demanda esperada.
3. Establecer políticas de operación.
4. Determinar niveles de inventario.
5. Clasificar.
6. Diseño de área de departamentos y distribución de materiales general.
7. Partición de almacén.
8. Diseño de manejo de material y acomodo.
9. Diseño de pasillos.
10. Determinación de requerimientos de espacio.
11. Determinación del número y localización de puntos de entrada y de salida.
12. Determinación del número y localización de áreas de descarga.
13. Arreglo del almacén.
14. Formación de zonas.

Tabla 2.1: Pasos generales para el diseño de un almacén

máxima. Además, si el objetivo principal de la distribución de materiales fuera minimizar el volumen del espacio ocupado por los productos, posiblemente el problema esté en la mala planeación del nivel de inventario o de las dimensiones del almacén y no en el acomodo de los materiales.

2.4.1. Problema de Distribución de Materiales de Almacén

El Problema de Distribución de Materiales de Almacén (WLP por sus siglas en inglés) ha sido atacado desde diversas perspectivas, tanto estocásticas o difusas como determinísticas [29]. Lo que se espera al resolver el problema de la distribución de materiales de almacén es un plan de acomodo adecuado para ubicar los diferentes ítems en un almacén de modo que el costo total de transporte sea minimizado. Otra manera de ver el problema es: minimizar la suma de las distancias desde dos nodos de un grafo al resto de los nodos en el grafo cambiando el peso de cada nodo siguiendo un grupo de restricciones. Al hacer la distribución de materiales se intentará reducir el costo de transporte. No se considerará la restricción de capacidad de los camiones de carga.

Para esta versión del problema, el almacén tiene un sólo piso, una entrada y una salida (que pueden o no estar en el mismo lugar). Para experimentar, se consideró la posibilidad de que el algoritmo busque también la configuración adecuada de pasillos, los resultados para esto fueron interesantes, proponiendo a veces pasillos en diagonales y generando propuestas muy diferentes a las tradicionales. Dado que las diagonales normalmente no son adecuadas en los almacenes y para mantener al mínimo la necesidad

de heurísticas para generar las soluciones, se optó por fijar previamente los pasillos para mantener las propuestas de modo que puedan ser mejor visualizadas.

Cada material tiene un valor ponderado basado en la demanda esperada, es un tipo de medida de movimiento del material. La demanda esperada, y por lo tanto la prioridad dada por el valor de peso, es independiente del área ocupada en el almacén. Por ejemplo, puede haber un material que por razones de seguridad debe estar en el almacén incluso si está caducado, o por otro lado, los perecederos pueden tener mucho movimiento aunque se manejen pocas cantidades.

Para representar el almacén, se utilizó una matriz de $m \times n$ como mapa del lugar, usando esta matriz se genera un grafo donde los nodos adyacentes son las celdas adyacentes en el mapa. La entrada y la salida son parámetros dados, que pueden ser el mismo, y lo que se minimiza es la distancia ponderada de cada celda a la entrada y la salida. Para medir la distancia, se utilizó un algoritmo Dijkstra del camino más corto entre dos nodos de un grafo [12]

Cada casilla puede pertenecer a una de tres clasificaciones:

- *Restringidas*: tienen peso infinito, esto representa áreas no relacionadas con el movimiento del material, por ejemplo, oficinas. Estas casillas no conectan en el grafo con las adyacentes y no serán modificadas en ningún momento por el algoritmo.

- *Materiales*: tienen peso finito, representan el material que será movido. Los materiales tienen como parámetro la clasificación de importancia. Mientras mayor sea este valor significará que el material tiene mayor movimiento y debe tener fácil acceso. Al momento de evaluar, se multiplica el peso dado de los materiales por la importancia específica del material evaluando.

- *Pasillos*: tienen peso finito, aunque menor que el de los materiales, representan el área por el que se trasportará el material. Este valor es opcional, se permite que el almacén se represente como completamente lleno. Los pasillos también tienen un parámetro clasificador de importancia que es mayor al de cualquiera de los materiales, éste parámetro dirige da preferencia a soluciones donde todos los pasillos estén completamente conectados.

Además de lo anterior, en un almacén es deseable que los materiales de características similares estén juntos. En esta simplificación del problema, esto último está siendo considerado como una restricción suave, y para evaluarlo, se miden las áreas donde se encuentra adyacente un mismo tipo de material, se eleva cada valor en un factor cercano a uno y el resultado se le resta a la evaluación obtenida por recorridos, esto se verá a detalle en el capítulo 3.1.

Una forma de aproximar la complejidad del problema es viéndolo como un problema de permutación con repeticiones. De este modo, si tenemos un conjunto de n

materiales de modo que del primer tipo hay n_1, del segundo hay n_2 y del k-ésimo hay n_k, el total de permutaciones son:

$$PR = \frac{n!}{\prod_{i=1}^{k} n_i!}$$

.

Los algoritmos utilizados tienen una complejidad por evaluación aproximada a las dimensiones de la matriz ($A = m \times n$) de croquis por la cantidad de materiales Mat más la complejidad del algoritmo Dijkstra para caminos más cortos en un grafo ($4n \log A$) [24], es decir, $m \times n \times \text{Mat} + 4n \log n$. En el caso del algoritmo de colonia de hormigas, cada iteración se hace h veces (una evaluación por hormiga), y en el DE este valor además se multiplica por la población.

2.5. Trabajo previo

El problema de distribución del almacén pertenece al tipo de problemas de diseño de almacén, que tienen diversas perspectivas y modelos independientes que se dividen en problemas estratégicos, tácticos y operacionales [3].

La importancia relativa de un criterio en particular varía con los tipos de almacenes, se distinguen dos tipos de almacenes: los de distribución y los de producción. Un almacén de producción tiene como función almacenar materias primas, trabajo en proceso y productos terminados, estos productos están almacenados por largos periodos de tiempo, estos almacenes buscan maximizar la capacidad de almacenamiento, minimizando los costos operacionales y de inversión. La función de un almacén de distribución es almacenar productos para satisfacer consumidores externos, y en general se busca la eficiencia en costo de la salida de material, el objetivo de los modelos para diseñar este tipo de almacén es maximizar el flujo de materiales manteniendo al mínimo la inversión [3].

Un diseño de almacén en el que se utilice una aproximación puramente analítica o puramente de simulación no llevará en general a un método de diseño práctico, sin embargo, una combinación de estos puede llevar a un buen método de diseño [2].

Dependiendo de las políticas de almacenamiento, se asignan los lugares de los maerial. Estos pueden ser asignados aleatoreamente o agrupados en la misma área, basados en el orden o volumen en que son recogidos. Una política de almacenamiento basado en el volumen acomoda los materiales con mucho movimiento cerca de área de salida para disminuir el viaje en que el material es recogido [14]. En el presente trabajo, se busca minimizar la distancia recorrida para tomar los materiales modificando el acomodo de los mismos, considerando las políticas de acomodo por volumen.

Otra forma en que se ha buscado minimizar la distancia recorrida para la salida de material es optimizando el orden en que se recogen los lotes [8], para esto se tradi-

cionalmente se usan heurísticas de Primeras Entradas, Primeras Salidas, y se intenta mejorar estos resultados con metaheurísticas como Algoritmos Genéticos [9].

La técnica de Diferenciación Evolutiva ha sido aplicada para optimización indirecta del almacén como parte del flujo de materiales en la cadena de suministro [23]. Bajo esta perspectiva, lo que se busca minimizar no es la distancia del material a la salida, sino la cantidad de material que será almacenado.

El modelo utilizado en el presente documento se basó principalmente en el propuesto en [28], eliminando las características difusas y permitiendo que un mismo material ocupe más de un espacio de la representación del almacén. Al permitir que más de una sección tenga la misma clave que otra permite reducir la complejidad, de este modo, mientras más materiales del mismo tipo haya habrá menos combinaciones posibles, sin embargo, se puede manejar todos los materiales distintos sin más cambio que el aumento en la complejidad.

Capítulo 3

Codificación

En este capítulo se explicará la forma en que el problema fue codificado, evaluado, así como la forma en que fue se aplicaron las técnicas de optimización. La primer sección es común a todas las técnicas utilizadas. La segunda sección es la forma en que fue aplicado el algoritmo de Colonia de Hormigas. Puesto que existen características en común entre la forma en que se aplicó la Diferenciación Evolutiva y el algoritmo de Colonia de Hormigas, las secciones siguientes se explican los cambios hechos para aplicar la técnica de Diferenciación Evolutiva y el sistema secuencial. Cuando los algoritmos son llamados como funciones por otros algoritmos, se incluirá entre paréntesis el nombre de la función.

3.1. Codificación del problema de WLP y su evaluación

Para codificar el WLP, se genera una matriz donde cada casilla l representa un grupo de tarimas, un pasillo o un área prohibida; esta matriz puede ser vista como un croquis, donde los números enteros representan áreas de tarima, los ceros representan áreas de pasillo y los símbolos de infinito (∞) representan áreas restringidas, donde no se puede agregar material y no son pasillo (Algoritmo 1), este algoritmo incluye una función llamada *pool* para limita los materiales a aquellos para los que haya cupo, o agrega espacios vacíos como materiales en caso que haya espacio extra. Tanto el área prohibida como la de pasillos, dada la definición propia del mapa, en este modelo se le considerará que es múltiplo de cada unidad de área de tarima. En la figura 3.2 se muestra un ejemplo de una matriz de resultado, con base en esta matriz será la distribución de las matrices de ejemplo de este capítulo.

Las diversas matrices que representan el croquis se convierten en grafos dirigidos considerando las casillas adyacentes como sus uniones, y los valores en la matriz, como los pesos que unen los diversos puntos. La excepción a esto son las casillas con material prohibido, que no tienen unión con ninguna casilla. Es sobre estos grafos que el algoritmo de optimización trabaja, sin embargo, en este documento se mostrarán sólo las

Algoritmo 1: Inicializar parámetros y generar representación de almacén (*Inicializar*)

```
1 mapabase ← GenerarMapa(x, y);
2 mapaviajes ← GenerarMapaViajes(mapabase, prohibidos, pasillos) ;
3 constantesMMAS ← GenerarConstantesMMAS ;
4 constantesDE ← GenerarConstantesDE ;
5 M ← GenerarPool(Materiales, mapaviajes)
```

matrices que representan el croquis por ser más fáciles de asimilar visualmente (Algoritmo 2). En este algoritmo, *casillas* es la lista de valores consecutivos del tamaño de *mapaviajes*; *eleccion* asigna un valor aleatorio de la lista *casillas*, y *restringir* elimina de la lista de opciones los materiales agotados.

Algoritmo 2: Generación de soluciones a partir del τ y el Mapa de Viajes (*ConstruirSolucion*)

```
1 casillas ← [1, 2..., m × n]  for i ∈ mapaviajes_i do
2     eleccion ← random casillas(eleccion) ←NULL ;
3     if mapaviajes(lugar) = areamateriales then
4         opciones ←restringir(lugar, mapaviajes, materiales)  for j ∈ opciones
          do
5             probabilidades(j) ← τ_{i,opciones(j)}
6         MatrizSolucion(i) ← Asignar(probabilidades, opciones)
```

El objetivo del modelo es reducir el movimiento de salida y entrada del material al almacén. En vez de utilizar una simulación iterativa para hacer la evaluación, esta se hace utilizando la probabilidad de manejo de un material como un valor que da un peso al material, de este modo, mientras mayor sea la probabilidad, más valor tendrá que el producto sea de fácil acceso. Para calcular la distancia, se considera que cada vez que se toma un material, el almacén se encontrará lleno, es decir, que para tomar el material en la casilla $l_{x,y}$, se necesita hacer a un lado el material de la casilla $l_{w,z}$, aunque éste en la práctica, tiene probabilidades de no encontrarse en ese punto por haber sido tomado antes.

Para evaluar el promedio de viajes necesarios para obtener o depositar material, se necesita un valor específico para las celdas de los pasillos y otro para las de materiales. El valor de los pasillos representa el viaje del montacargas a través del almacén, así que debe ser una medida de distancia; por conveniencia, usamos la unidad como valor para las celdas de pasillo. El valor de las celdas de los materiales representa el esfuerzo del montacargas de atravesar el material poniéndolos a un lado y regresándolos a su lugar

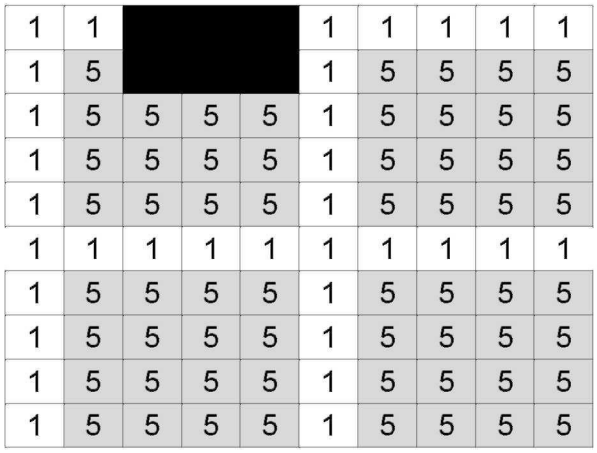

Figura 3.1: Matriz de viajes

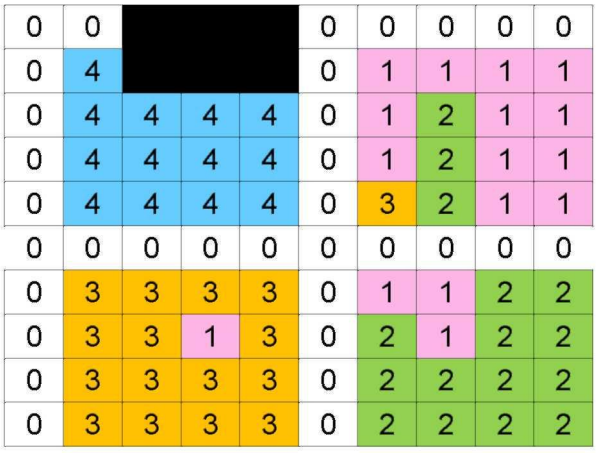

Figura 3.2: Matriz de solución

después de obtener el material que se quiere obtener. No hay una convención acerca de cuál debería ser este valor, no debe ser tan pequeño que el algoritmo prefiera mover un material para obtener el que está detrás que rodear un camino corto, pero tampoco debe ser tan grande que vuelva el viaje por los pasillos completamente irrelevante. Para calcular la distancia entre un material y la entrada d_{in} o la distancia entre un material y la salida la salida d_{out}, se utilizó un algoritmo Dijkstra para encontrar el camino más corto entre dos nodos de un grafo [12] [27], donde cada casilla de pasillo tiene un peso W de 1 y cada casilla de material tiene un peso finito mayor al del pasillo, las pruebas se hicieron con un peso W de 5 ó 10 como se muestra en la figura 3.1.

Lo que se espera minimizar es la distancia recorrida por el montacargas, sin embargo, al estar facilitando la obtención del material con mayor movimiento, la velocidad para obtener o acomodar material también está disminuyendo. La importancia de acomodar rápidamente los materiales en el almacén no es equivalente a que estos salgan

	a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*	j
1	1	∞	∞	∞	1	1	1	1	1	1
2	1	∞	∞	∞	0.1	1	0.4	0.4	0.4	0.4
3	1	0.1	0.1	0.1	0.1	1	0.4	0.3	0.4	0.4
4	1	0.1	0.1	0.1	0.1	1	0.4	0.3	0.4	0.4
5	1	0.1	0.1	0.1	0.1	1	0.2	0.3	0.4	0.4
6	1	1	1	1	1	1	1	1	1	1
7	1	0.2	0.2	0.2	0.2	1	0.4	0.4	0.3	0.3
8	1	0.2	0.2	0.4	0.2	1	0.3	0.4	0.3	0.3
9	1	0.2	0.2	0.2	0.2	1	0.3	0.3	0.3	0.3
10	1	0.2	0.2	0.2	0.2	1	0.3	0.3	0.3	0.3

Figura 3.3: Matriz de ponderaciones asignadas a cada material

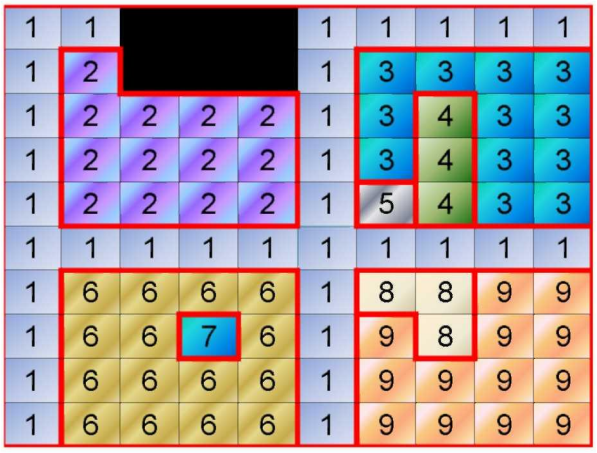

Figura 3.4: Matriz de agrupamiento

Figura 3.5: Matriz de conteo de grupos

rápido, siendo esto último por lo general más importante, por lo que el valor de la distancia de entrada puede no ser simético con el de salida. En el Algoritmo 3 esto es representado por una ponderación a_{in} que se multiplica por la suma de las distancias hacia la entrada, ó a_{out} si se dirigen hacia la salida .

Algoritmo 3: Evaluación de distancias (*EvaluaDistancias*)

1 $[d_{in}, caminoentrada] \leftarrow \omega_{s_i}\text{Dijkstra}(mapagrafo, entrada)$;
2 $[d_{out}, caminosalida] \leftarrow \omega_{s_i}\text{Dijkstra}(mapagrafo, salida)$;
3 $ed \leftarrow d_{in}a_{in} + d_{out}a_{out}$;

Cada material tiene un peso específico ω entre 0 y 1 que representa el porcentaje de la demanda que el material tiene con respecto a la demanda total. Este peso se multiplica por la distancia del material a la entrada o salida obtenida por la Matriz de Viaje para llegar a una matriz como la tabla 3.3. Por lo tanto, los materiales con mayor movimiento tendrán un valor más grande que otros con menor movimiento en una celda l con la misma distancia respecto a la entrada o salida. El peso específico de los pasillos también es representado en esta matriz, pero el valor que se le asigne debe diferir dependiendo de las características del modelo. En el ejemplo que estamos siguiendo, el peso de los pasillos debería ser de uno o mayor, para que si hubiera menos material que espacio para material, el modelo no busque dejar espacios vacíos rodeados de materiales, por ejemplo, si el peso de los pasillos fuera 0, y hubiera menos de cualquier tipo de material, dejar un espacio de pasillo rodeado de material, por ejemplo en la casilla $d4$, le parecería al algoritmo una mejor opción que asignarlo adyacente a un pasillo, por ejemplo en la casilla $d5$.

La forma en que está planteada la evaluación asume que el almacén siempre estará lleno cuando se requiera un material, es decir, es el peor caso posible; esta perspectiva tiene un problema: tiende a poner los materiales de mayor movimiento en las partes exteriores de las áreas de materiales, lo cual no es deseable si es poca la diferencia con los materiales que hay detrás, pues esto requeriría hacer movimientos del material con mayor movimiento siempre que se requiera obtener el que está detrás. Lo que tradicionalmente se hace en los almacenes es agregar material del mismo tipo con profundidad, es decir, hacer hileras de material similar uno detras de otro. Para promover en la evaluación este comportamiento, se utiliza un cambio en la evaluación de la distancia, de modo que en la ruta más corta del material hacia la entrada o salida, cuando se atraviese un material del mismo tipo a la casilla origen, el valor de distancia de pasar por ahí será menor al de pasar por un material de diferente tipo. Una forma física de interpretar el valor disminuido de la distancia de materiales del mismo tipo es considerando el tipo de manejo del almacén, es de Primeras Entradas, Primeras Salidas (PEPS). Al dirigirse hacia la salida no hay diferencia si el material que se debe tomar es del mismo tipo que los que están delante, se debe hacer a un lado

los materiales delante de éste para tomar el primero en llegar; de lo contrario, nunca se tomará un material que esté detrás de otro igual, sino que se tomará el primero en encontrarse. En este escenario, es conveniente que cuando la ruta más corta hacia la entrada o salida atraviese materiales del mismo tipo, este camino tenga un costo menor que si se tratara de distintos materiales. Bajo esta perspectiva, este cambio en la evaluación de distancia puede hacerse sólo al acomodo de material y no a su obtención cuando es PEPS, y hacerse de manera inversa cuando es Últimas Entradas, Primeras Salidas. Cuando esto es implementado, el tiempo de ejecución del algoritmo aumenta considerablemente, pues esto implica una búsqueda a posteriori del camino elegido para restar el costo de los materiales del mismo tipo (Algoritmo 5).

En los almacenes es recomendable que los materiales del mismo tipo se encuentren adyacentes o cercanos. Hasta el momento, la distancia total ponderada que el algoritmo está intentando minimizar no toma en cuenta que es deseable que los materiales del mismo tipo se encuentren juntos. Hacer que sólo las soluciones que satisfacen las restricciones de adyacencia formen parte de la búsqueda no fue considerado porque hubiese provocado que la construcción de las soluciones fueran demasiado dependientes de las primeras celdas generadas y no permitirían que un algoritmo de optimización se comportara con las suficiente flexibilidad. Para este modelo, se usó una función que reste a la distancia ponderada un valor dependiendo de los materiales adyacentes del mismo tipo (Algoritmo 6).

Para contar la cantidad de celdas de materiales juntas, se utilizó una función recurrente que busca los valores adyacentes a cada celda. Si la celda adyacente no ha sido visitada y tiene un valor igual a la celda origen, se le asigna un número que representa el orden o en que los grupos van siendo encontrados, aumenta un contador r_o asociado al orden, y por recurrencia, explora alrededor de esta celda. Si la celda adyacente ya ha sido explorada o es diferente, la ignora, y cuando ya no hay adyacencias del mismo valor, de manera consecutiva explora las siguientes casillas hasta que encuentre una que no ha sido visitada y vuelve a empezar la recursión, aumentando en uno el orden o (figura 3.4). Con ayuda de esta matriz, se genera una lista O con el peso ω del material M_o y su contador r_o asociado (figura 3.5), que representan los grupos que se formaron y su tamaño. Para obtener la bonificación de adyacencia V, cada tamaño de grupo es elevado a una potencia con un valor de prioridad o que debe ser ligeramente mayor a uno, este valor es multiplicado por el peso específico del material correspondiente. Finalmente, la suma se le resta a la distancia ponderada.

$$V = \sum_{O_o \in O} (r_o \omega_o)^v \tag{3.1}$$

El factor v es un auxiliar de prioridad que intenta dar más importancia a la restricción de adyacencia conforme el algoritmo hace su trabajo, provocando que en generaciones avanzadas, el material del mismo tipo en una buena posición se mantenga

junto. Si hay muchos materiales del mismo tipo, v debe ser más cercano a uno que de lo contrario, para que de este modo, el algoritmo no deje el material estático demasiado pronto. Si los pasillos se consideran fijos, no hay diferencia si estos reciben el mismo factor v que los materiales. En resumen, la forma en que trabaja el factor v es la siguiente: si v es igual a 1, no hay restricción de adyacencia, si v es menor a 1, el algoritmo separará los materiales del mismo tipo, si v es mayor a 1, mientras mayor sea el valor la restricción de adyacencia se volverá más dura en un momento más temprano del algoritmo. Si en vez de material, en la casilla l_i hay pasillo, ω_{o_i} debe sustituirse por el peso del pasillo.

Dado los múltiples niveles de datos que se necesitan para evaluar el modelo, es difícil escribir una fórmula de función objetivo algebraica, una representación de alto nivel de la operación de minimizado es:

$$Min\, E = a_{in} \sum_{i=1}^{m} \sum_{j=1}^{n} \omega_k s_{in_{i,,j}} + a_{out} \sum_{i=1}^{m} \sum_{j=1}^{n} \omega_k s_{out_{i,,j}} - V - EST \tag{3.2}$$

donde a_{in} y a_{out} son respectivamente la ponderación por la importancia de acomodar y obtener material respectivamente; $\omega_k s_{in_{i,,j}}$ es la ponderación del material que fue asignado a la casilla que se está evaluando; V es la bonificación por tener materiales del mismo tipo adyacentes (Algoritmo 6) y EST es la bonificación por tener materiales del mismo tipo alineados (Algoritmo 5). Cada variable de la evaluación es llamada por el Algoritmo 4, en donde primero el Mapa de Viajes es convertido en un grafo para que se puedan calcular las distancias entre las casillas.

Algoritmo 4: Evaluación en alto nivel (*Evalua*)

```
1 mapagrafo ← conectar(s)
2 for s_i ∈ s do
3     [fd, caminoentrada, caminosalida] ←EvaluaDistancia(mapagrafo);
4     fa ←EvaluaAdyacencia(s, caminoentrada, caminosalida);
5     ft ←EvaluaSimilaresEnTrayectoria(s, caminos);
6     evaluacion ← evaluacion + fd − fa − ft;
7 if evaluacion < s_ib then
8     s_ib ← s;
9     fs_ib ← evaluacion
```

El algoritmo 6 incluye las funciones *Adyacentes* y *EvaluacionAdyacencia*, siendo la primera una función recursiva que identifica las casillas similares alrededor de la casilla actual y devuelve una tabla de conteo de adyacentes como la figura 3.5; *EvaluacionAdyacencias* recibe como parámetros M_i, que es el material que se está analizando y r_i su

Algoritmo 5: Cálculo que se hace para identificar y evaluar materiales similares al que se está revisando en su recorrido a la entrada o salida (*EvaluaSimilaresEnTrayectoria*)

```
1 for caminoentrada_i ∈ caminoentrada do
2     if mapasolucion_i = mapasolucion_{caminoentrada_i} then
3         et ← et + bonus
```

cantidad. La variable *matrizadyacencia* es inicializada como una matriz de ceros, l es la cantidad de celdas con cantidades diferentes.

Algoritmo 6: Evaluación de adyacencia (*EvaluaAdyacencia*)

```
1 matrizadyacencia ← s0  for s_{i,j} ∈ s do
2     if matrizadyacencia_{i,j} = 0 then
3         [matrizadyacencia, O_{clasificador}] ←
          Adyacentes(i, j, matrizadyacencia, s, clasificador);
4         clasificador ← clasificador + 1
5 for i ← 1, 2..., l do
6     EvaluacionAdyacencia ← EvaluacionAdyacencia + (ω_{s_{Mi}} * r_i)^v
```

3.2. Codificación de WLP para un algoritmo de colonia de hormigas

Un algoritmo de Colonia de Hormigas utiliza la matriz de trazo de feromonas τ para generar las soluciones. Para este problema, la matriz se genera asignándole probabilidades de que cada casilla que represente un área de material pueda tener uno de los valores factibles. Por lo tanto, el tamaño de la matriz del camino de feromonas τ es igual al tamaño de la Matriz de Viajes por la cantidad de materiales (Algoritmo 7). En la figura 3.6 se muestra un diagrama con el orden en que los algoritmos son llamados de acuerdo al algoritmo de alto nivel de MMAS, *PrincipalMMAS*. Entre paréntesis está el número de algoritmo que representa cada función.

El total de feromona para cada posición en el almacén es siempre 1, de modo que la cantidad de feromona correspondiente a cada material es igual a la probabilidad de que éste sea elegido por el algoritmo al asignar ese material a la casilla. Cada material es inicializado con las mismas probabilidades de que éste sea elegido al principio del algoritmo, es decir, cada material de cada casilla tiene una probabilidad inicial

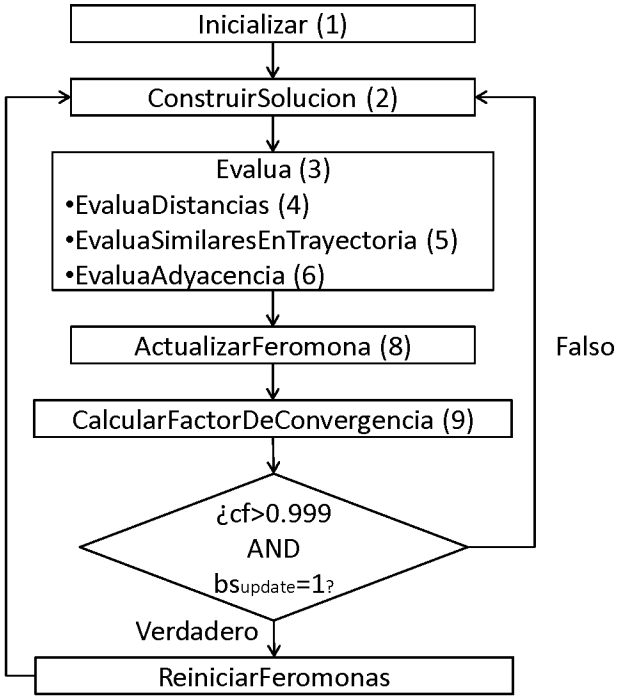

Figura 3.6: Diagrama de alto nivel del algoritmo MMAS

de $1/Mat$, donde Mat es la cantidad de materiales diferentes.

Aunque inicialmente los algoritmos ACO fueron utilizadas para resolver problemas de agente viajero, ya han sido aplicados para generar *layouts*, o acomodos, en diferentes contextos [10]. A continuación, se presenta el procedimiento utilzado para que, en base a la codificación en forma de grafo de una representación de área de almacén, se genere un camino de feromonas que permita dirigir la búsqueda local hacia propuestas de solución prometedoras.

Para crear la solución en base al camino de feromonas, se genera un contenedor de casillas y uno de materiales, el primero para evitar que se elijan casillas repetidas, el segundo, para que el algoritmo no tenga posibilidad de agregar materiales cuyo total de existencias ha sido completamente acomodado. De los valores que hay en el contenedor de casillas, se elige uno aleatorio, y se agrega el valor del material de acuerdo a las probabilidades que le corresponden según la matriz de camino de feromonas (τ). Este procedimiento es repetido hasta que no queden casillas libres para asignar material. Se prefirió elegir la siguiente casilla a la que se asignará material de manera aleatoria, por que si se eligieran de manera ordenada, las últimas casillas siempre tendrían el

Algoritmo 7: Algoritmo Principal de MMAS (*PrincipalMMAS*)

```
1  Inicializar;
2  ciclo ← ciclo + 1;
3  while Condiciones satisfechas do
4  |   for i = 1 to h do
5  |   |   s ← ConstruirSolucion(τ, mapaviajes, pool)
6  |   [s_ib, fs_ib] ←Evalua(s, pool)
7  if fs_ib < fs_rb OR s_rb = ∞ then
8  |   s_rb ← s_ib;
9  |   fs_rb ← fsib(cycle);
10 if fs_ib < fs_bs OR fs_sb = ∞ then
11 |   s_bs ← s_ib;
12 |   fs_bs ← fsib(cycle);
13 |   ReinicioUtil ← 1;
14 τ ← ActualizarFeromona(cf, bs_update, τ, s_ib, s_rb, s_bs)
15 cf ← CalcularFactordeConvergencia;
16 if cf > 0.999 then
17 |   if bs_update = TRUE then
18 |   |   ReiniciarFeromonas(τ);
19 |   |   s_rb ← NULL;
20 |   |   bs_update ← FALSE;
21 |   |   reset ← 0;
22 |   |   fs_rb ← ∞
23 |   bs_update ← TRUE
```

material sobrante del resto de las casillas en vez de obedecer a sus propias feromonas. En caso de haber material faltante, el algoritmo asignará un área de pasillo a la casilla. Si existe más material que espacio para acomodarlo, el algoritmo intentará acomodar completamente los primeros materiales con los que haya sido alimentado el algoritmo, independientemente del peso que tengan. Esto se decidió así para que el usuario decida la importancia del material que debe estar dentro del almacén; el último material que quepa en el contenedor será truncado. Dado que la generación de soluciones sí está restringida, la probabilidad real de que una casilla dada tenga un material específico no es igual al nivel de feromonas durante todo el proceso de generación de solución, esto es porque una vez que el material está agotado, este no puede ser utilizado en una nueva casilla independientemente del nivel de feromonas que esté asignado para ésta; de éste modo también aumentan las probabilidades que alguna casilla obtenga materiales con muy poco nivel de feromonas, ya que si están agotados al momento de hacer la asignación los materiales que en el pasado generaron un fuerte trazo, estos no serán considerados al generar la solución.

Para elegir la forma en que se actualizará el camino de feromonas, sólo la mejor hormiga por iteración (i_{bs}), la mejor desde el reinicio (s_{rb}) y la mejor global (s_{bs}) serán consideradas; una iteración consiste en una cantidad h de soluciones generadas utilizando el mismo camino de feromonas y sin actualizarlo. A cada iteración se le llama hormiga, mientras más hormigas se usen, mayor será la exploración que se utilice antes de intensificar con el trazo de feromona de la mejor hormiga (Algoritmo 8). Las áreas prohibidas y los pasillos forzados (representadas en el algoritmo por la variable *inmovil*) forman parte de la matriz τ, sin embargo, sus valores son ignorados por el algoritmo, por lo que su valor no cambia en ningún momento.

Una de las características del ACO MMAS es que existen un τ_{MIN} y un τ_{MAX} que impiden que el algoritmo se estanque irremediablemente en un máximo local, permitiendo siempre que exista la exploración. Conforme la concentración de feromonas va aumentando para cierto valor en cada casilla, puede llegar un momento en que la diferencia entre las probabilidades del valor más grande y los demás pueda ser demasiado grande. Para evitar esto último es que se elige un τ_{MAX}, que limita la diferencia, por ejemplo de 0.99. En caso de que el camino de feromona intente actualizarse a un valor mayor a éste, con la función *Corregir* su valor se regresará a τ_{MAX}, del mismo modo, aquellos valores de τ que debieran tener un valor menor τ_{MIN} será convertidos a τ_{MIN}, en este caso es igual a $(1 - \tau_{MAX})/(Mat - 1)$, con esto, podemos permitir que la cantidad de feromona en cada casilla pueda sumar 1 con la función *Normalizar*.

Otra de las mejoras del algoritmo de [5] sobre el ACO básico es el uso de los reinicios del algoritmo. Una vez que un parámetro de convergencia cf (Algoritmo 9) se acerca a su valor máximo (típicamente, cf es un valor entre 0 y 1, y se considera que hay convergencia cuando alcanza un valor cercano a 1, por ejemplo, 0.999), el algoritmo se reinicia recordando el s_{bs}, volviendo a darle importancia en un principio

Algoritmo 8: Actualización de feromonas (*ActualizarFeromona*)

```
if bs_update = 1 then
    k_bs ← 1;
    k_ib ← 0;
    k_rb ← 0
k_ib ← m_{k_ib} + b;
k_rb ← 1 − k_ib;
for s_i ∈ s do
    if s_i ≠ inmovil then
        τ_sib ← k_ib ωs_{ib_i} / Σω;
        τ_srb ← k_rb ωs_{rb_i} / Σω;
        τ_sbs ← k_bs ωs_{bs_i} / Σω;
        m ← τ_{s_ib} + τ_{s_rb} + τ_{s_bs};
        τ ← τρ + (1 − ρ)m;
        τ ← Normalizar (τ);
        τ ← Corregir (τ, τ_MAX, τ_MIN);
```

sólo al mejor de la iteración. Conforme el algoritmo avanza nuevamente, le da cada vez mayor importancia al mejor encontrado de el presente reinicio s_{rb}, y cuando vuelve a alcanzar el valor máximo de cf, empieza a actualizar sólo con base en el mejor encontrado de la corrida completa (sbs). Dicho de otro modo, conforme el algoritmo avanza, la importancia que se le da al mejor valor de la iteración disminuye, mientras que aumenta para el mejor encontrado de la corrida completa (s_{bs}), de modo que hacia el final del reinicio, es prácticamente irrelevante el s_{ib}, la exploración es mucho menor y se concentra en mejorar el s_{bs}. En el algoritmo MMAS propuesto en [26] propone cambios discretos de probabilidad dependiendo del cf, en lugar de eso, para esta propuesta se utilizó una interpolación lineal para generar un cambio gradual, como se muestra en las fórmulas 3.6, 3.7 y 3.8. De esta forma, se hace una exploración extra dirigiendo el τ actual hacia la mejor solución encontrada, disminuyendo de nuevo el valor de cf. En caso de que la solución encontrada sea igual a la de s_{bs}, lo que sucederá es que el cf no cambiará, volviendo a llamar al reinicio del sistema. Para este problema, el cf es calculado de la siguiente manera:

$$cf = \frac{\sum_{i=1}^{T} \text{máx}\{\tau_i\} - T\tau_{Inicial}}{A\tau_{MAX} + C\tau_{Inicial} - T\tau_{Inicial}} \tag{3.3}$$

donde A es la cantidad de casillas en las que se les puede asignar material, C es la cantidad de casillas reservadas para pasillo o prohibidas, T es la cantidad de casillas totales y $\tau_{Inicial}$ son las probabilidades con las que el algoritmo arranca para cada

casilla, que son las que permanecen inmutables para las áreas reservadas durante todo el algoritmo. Lo que se está haciendo con esta formula es dividir la cantidad actual de concentración de feromona entre la máxima concentración posible, de este modo, conforme el valor de cf se acerca a 1, la concentración de feromona total se acerca a τ_{MAX} y por lo tanto, se está cada vez más cerca del estancamiento. La razón por la que se resta el $\tau_{Inicial}$ es para que al iniciar el algoritmo parta desde 0.

Algoritmo 9: Calculo de factor de convergencia (*CalcularFactordeConvergencia*)

1 $valormaximo \leftarrow A\tau_{MAX} + C\tau_{Inicial} - T\tau_{Inicial}$
$valoractual \leftarrow (\sum_{i=1}^{T} \sum \tau_{i,j} \in \tau_i - \tau_{inicial})$;
2 $cf \leftarrow \frac{valoractual}{valormaximo}$

Para actualizar los caminos de feromona, se usa el promedio ponderado de las tres soluciones s_{ib}, s_{rb} y s_{bs}, que en el modelo hipercúbico se especifica como:

$$\tau \leftarrow \tau + \rho(\overrightarrow{m} - \tau) \tag{3.4}$$

con

$$m = k_{ib}s_{ib} + k_{rb}s_{rb} + k_{bs}s_{bs} \tag{3.5}$$

donde k_{ib}, k_{rb}, k_{bs} son los pesos de s_{ib}, s_{rb} y s_{bs} respectivamente; estos pesos deben sumar 1 y son asignados dependiendo si está activado bs_{update} o no. Si bs_{update} es igual a 0, se utilizan las siguientes fórmulas:

$$k_{ib} = \begin{cases} 1, & cf < 0.4 \\ cfm_{k_{ib}} + b, & 0.4 < cf < 0.8 \\ 0, & cf > 0.8 \end{cases} \tag{3.6}$$

$$k_{rb} = \begin{cases} 1, & cf < 0.4 \\ 1 - k_{ib} + b, & 0.4 < cf < 0.8 \\ 0, & cf > 0.8 \end{cases} \tag{3.7}$$

$$k_{bs} = 0 \tag{3.8}$$

donde $m_{k_{ib}}$ y b son la pendiente y constante de una recta que devuelve 1 cuando la abscisa está en 0, y 1 cuando la abscisa está en 1. Si bs_{update} es igual a 1, entonces, $k_{ib} = 0$, $k_{rb} = 0$ y $k_{bs} = 1$. Una vez que se aplica la regla de actualización, los valores son reiniciados según corresponda a τ_{MAX} o τ_{MIN}.

Dado que el modelo utilizado no siempre respeta la probabilidad propuesta por el τ para la generación de soluciones, se incluyó una variable adicional para permitir

que el algoritmo se mantenga con las probabilidades que llevaron a $cf \approx 1$, permitiendo de este modo explotar un poco más el camino de feromonas antes de reiniciarse. Esta variable, a la que se llamó *holgura* sucede en dos ocasiones: antes de que se reinicie el algoritmo al empezar a actualizar según k_{bs} (*holgura*), y antes de pasar de k_{rb} a k_{bs} (*holgura*2). Aunque el algoritmo ACO-MMAS utilizado se puede considerar un caso especial cuando la holgura es igual a cero, sin embargo, por cuestión de claridad, en este documento se consideraron independientes, en el Algoritmo 10 se muesta el *PrincipalMMAS* considerando la holgura.

3.3. Codificación de Diferenciación Evolutiva

Un algoritmo DE parte de una población de vectores con valores reales, la diferencia entre ellos es utilizado como perturbación para modificar vectores de acuerdo a su nivel de variabilidad. Conforme los vectores se parecen más entre ellos, las perturbaciones son menores hasta llegar a una convergencia. El Algoritmo 11 muestra el proceso DE en alto nivel, y en la figura 3.7 se muestra la forma en que los diferentes algoritmos se relacionan, incluidos los que se usaban para el método de Colonia de Hormigas y que también son aplicados para DE. Un algoritmo DE no fue aplicado directamente al modelo del almacén porque los valores que sirven para evaluar la optimización son del tipo nominales discretos, por lo que no tienen sentido las perturbaciones para valores reales propias del algoritmo. Sin embargo, la representación de los caminos de feromonas τ utilizados por un algoritmo ACO si manejan valores reales que pueden ser utilizados para una población de un algoritmo DE. Lo que se está proponiendo es usar poblaciones de rastros de feromona para generar a su vez, otros trazos de feromonas que tienen mejores oportunidades de obtener buenas soluciones.

Los individuos consisten de una matriz donde cada fila representa una celda en el mapa del almacén, mientras que cada columna representa un tipo de material. Cada celda en la matriz representa las probabilidades de que un material de la fila sea puesta en la celda de la columna en el mapa, de modo que la suma de las celdas horizontales siempre sea igual a uno. Cada individuo de la generación inicial es generado aleatoriamente; si se hará nueva generación en un momento del algoritmo que no es inicial, por ejemplo, después de una etapa de optimización por ACO, esta deberá tener implícito el conocimiento del mejor encontrado s_{bs}.

La terminología de Diferenciación Evolutiva que se utilizará a continuación es una adaptación libre al español de la terminología utilizada en la literatura [22] que es coherente con esta versión del algoritmo. Una vez que la población es generada, se crean mutantes que son comparados con la población p. Para esto, se eligen dos individuos aleatorios a los que se les llamará *padres*, se restan sus valores, y esta diferencia llamada *mutación* es agregada a un tercer vector. Finalmente, el valor es

Algoritmo 10: Algoritmo Principal de MMAS con holgura (*PrincipalMMAS*)

```
1  Inicializar;
2  ciclo ← ciclo + 1;
3  while Condiciones satisfechas do
4      for i = 1 to h do
5          s ← ConstruirSolucion(τ, mapaviajes, pool)
6      [s_ib, fs_ib] ←Evalua(s, pool)
7      if fs_ib < fs_rb OR s_rb = ∞ then
8          s_rb ← s_ib;
9          fs_rb ← fsib(cycle);
10         TRUE ← nuevomejorreset
11     if fs_ib < fs_bs OR fs_sb = ∞ then
12         s_bs ← s_ib;
13         fs_bs ← fsib(cycle);
14         ReinicioUtil ← 1;
15     τ ← ActualizarFeromona(cf, bs_update, τ, s_ib, s_rb, s_bs)
16     cf ← CalcularFactordeConvergencia;
17     if cf > 0.999 then
18         if bs_update =TRUE OR holgura >= holguramax then
19             ReiniciarFeromonas(τ);
20             s_rb ← NULL;
21             bs_update ← FALSE;
22             reset ← 0;
23             fs_rb ← ∞
24         else if holgura2 >= holgura2max then
25             bs_update ← TRUE;
26             reset ← 0;
27             holgura ← holgura + 1;
28         else
29             holgura2=holgura2+1
30     else if nuevomejorreset=TRUE then
31         holgura ← 0;
32         holgura2 ← 0
33     FALSE ← nuevomejorreset
```

Algoritmo 11: Diferenciación Evolutiva en alto nivel (*PrincipalDE*)

```
1  while Condiciones cumplidas do
2      fs_ib ← ∞;
3      s_ib ← NULL;
4      promedio ← NULL;
5      rivalespool ← poblacion hijospool ← poblacion
6      for i = 1, 2..., p do
7          mutante ←GenerarMutante;
8          [p, s_ib, fs_ib] ←ElegirSobreviviente;
9      if s_bs = NULL OR fs_ib > fs_bs then
10         s_bs ← s_ib;
11         fs_bs ← fs_ib;
```

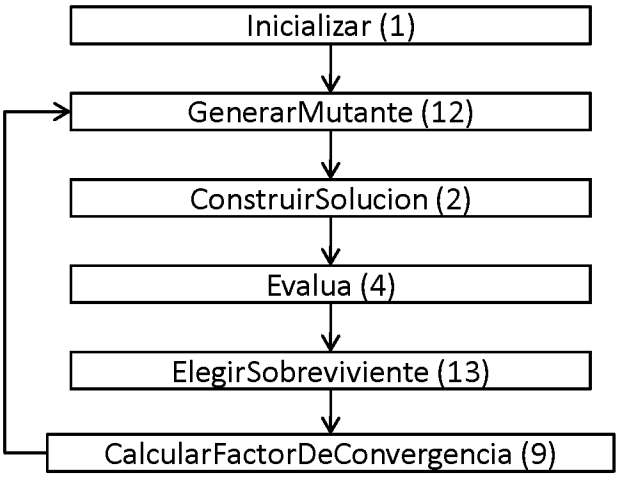

Figura 3.7: Diagrama de alto nivel de Diferenciación Evolutiva (PrincipalDE)

normalizado de modo que la suma de las columnas sea igual a 1, a este nuevo individuo se le llamará *mutante*. De acuerdo a ciertos parámetros, se selecciona un miembro de la población, al cual se le llamará *rival* y se comparará con el *mutante*, conservando en la siguiente generación aquel que tenga una mejor evaluación. Cada individuo puede ser uno de los padres del siguiente *mutante* varias veces por generación, pero dependiendo de la forma en que sean establecidas las variables, sólo pueden ser *rival* o *hijo* una vez por generación. Para evaluar cada individuo, se deben generar i soluciones que son evaluadas con la función objetivo, y entonces, la media de estas evaluaciones es tomada como la aptitud del individuo. La cantidad de las iteraciones de cada individuo puede verse análoga a la cantidad de hormigas en un algoritmo ACO.

Para generar el *mutante* de la iteración, se utiliza el algoritmo 12, que incluye las siguientes funciones: $GenerarMutagen$, que define el área a la que se aplicará la mutación; $Normalizar$, que vuelve la suma de probabilidad de materiales igual a 1 y elimina las probabilidades generadas para pasillos forzados y áreas prohibidas; *construir* genera h soluciones utilizando el algoritmo 2 y evalua tomando la mejor solucion y el promedio de soluciones.

En equivalencia al cruce de los algoritmos genéticos, se utiliza un parámetro $v_{i,j}$ para que se generen esquemas matriciales de la misma forma en que se generan esquemas según propone Golberg [15]. Para esto, al momento de realizar una mutación, se genera una matriz auxiliar *mutageno* como la de la figura 3.8 del tamaño del croquis del almacén y con un área señalada de tamaño aleatorio, esta área (que en el la figura 3.8 se representa con números 1) es el área a la que se aplicará el ruido en el vector *hijo*. También se programó un cruce inverso, es decir, el área generada será respetada, análogo al cruce de dos puntos de los algoritmos genéticos. Este cruce sólo fue probado en los casos donde una probabilidad alta de cambio es contraproducente. Para el cruce inverso se debe tomar en cuenta que esta preferentemente, esta área no debe tener altas probabilidades de ser mayor a una cuarta parte de la Matriz de Solución, o habrá áreas con menores probabilidades de cambio con respecto a las demás.

Para decidir que parámetros se utilizarán para elegir el rival y el vector a mutar la literatura presenta varias propuestas [25], para este trabajo se eligió la de rival único ($DE/rand/1$) y $DE/best/1$. Otras técnicas se han propuesto para aumentar las propiedades de avaricia del algoritmo, sin embargo, en este proyecto se experimentó con modelos propios cuyo objetivo es disminuir la avaricia en vez de aumentarla, las propuestas son las siguientes:

- *HijoÚnico*: En cada generación, se podrá modificar cada vector una sola vez. Esto disminuye el efecto de que en una misma generación, ligeras modificaciones de un vector con calificación superior al resto empiece a eliminar al resto de la población reproduciéndose rápidamente.

- *RivalÚnico* o *DE/rand/1*: En cada generación, se podrá comparar el vector hijo

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Figura 3.8: Matriz de Mutágeno.

con cada vector una sola vez. Esto se propone para disminuir el efecto que al comparar varios vectores con el más apto de la generación, se disminuya el cambio en la población. Este es el modelo que originalmente se propuso para el algoritmo (Algoritmo 12).

- *HijoRivalUnico*: Con la combinación de los modelos anteriores, se esperaría que disminuyera la velocidad de convergencia con respecto al modelo completamente aleatorio.Este método es análogo al torneo propuesto para los Algoritmos Genéticos para facilitar el escalamiento, dado que la técnica DE es escalable por si misma, no es necesario un método como éste, sin embargo, se provará durante los experimentos.

- *Libre*: No hay una forma dirigida para definir el hijo ni el rival, de modo que la única forma en que se diferencían las generaciones es por la cantidad de población que se modificó. No hay un objetivo definido con esta técnica con respecto a las otras, sin embargo, primeros experimentos mostraron un buen comportamiento y se optó por utilizarla también durante los experimentos finales.

- *AutoSuperación*: Cada vector hijo se compara sólo consigo mismo. Este método debería comportarse como una búsqueda de alpinista en paralelo, esta propuesta fue desechada porque, aunque en primeros experimentos la población parecía tender a una convergencia, esto en realidad era provocado por un efecto indeseable cuando el vector *hijo* era el mismo que uno de los padres, pues si los signos eran correctos, uno de ellos se reproducía, y si éste remplazaba al vector *rival*, la población tendía a parecerse. Una vez que se limitó al *rival* como un vector diferente a los padres, todos los esquemas se comportaron mejor.

- *DE/best/1*: El único vector que se elegirá para mutar es el que tenga mejor

evaluación de la generación, el *rival* se elige con un esquema de *RivalÚnico*. El efecto de este esquema es aumentar la avaricia del algoritmo.

Algoritmo 12: Generar propuesta de individuo (*GenerarMutante*)

```
1 while padre=madre OR hijo=padre OR hijo=madre do
2     padre ← poblacion_aleatorio madre ← poblacion_aleatorio
3 mutagen ← GenerarMutagen elegido ← aleatorio
  hijo ← poblacion_{hijospool_elegido};
4 mutacion ← (padre − madre)mutagen;
5 mutantprevio ← |hijo + (Fmutacion)|;
6 τ_mutante ← Normalizar(mutante)
7 [s_mutante, fs_mutante, f̄_mutante ← Construir(τ_mutante, M, h)
8 if hijounico = TRUE then
9     hijospool(elegido) ← NULL
```

La razón por la que se está evaluando el promedio de un grupo de soluciones y no la mejor solución encontrada durante cada corrida, es porque, a diferencia de un algoritmo ACO, la convergencia no se dirige hacia la mejor solución encontrada, sino al conjunto de probabilidades que dirigen hacia ella, por lo que si aleatoriamente, un τ lleva a una buena solución que no concuerda con las probabilidades a las que tiene tendencia, se puede generar un engaño [7] que puede ser evitado simplemente tomando en cuenta varios resultados, aunque el mejor encontrado se sigue guardando como s_{bs}. Conforme el algoritmo avanza, los individuos que generan mejores promedios son también aquellos que dan mayores probabilidades a un conjunto ganador, por lo que el nivel de feromona para los materiales que conducen a buenas calificaciones va aumentando progresivamente, y conforme esto sucede, el promedio de la evaluación se acerca cada vez más al del mejor encontrado en la iteración.

Para la selección del ganador, aunque al principio se pensó en utilizar medios artificiales para permitir que un *mutante* tomara el lugar de un *rival* mejor con cierta probabilidad para mantener a la población en movimiento, el sistema de promedios permite hacer este trabajo automáticamente, ya que siempre existe la probabilidad de que un individuo que en el pasado haya tenido soluciones superiores sea superado por otro que en promedio tiene mejores soluciones. De hecho, esta misma posibilidad hace que no se pueda utilizar una población estática como condición de paro, puesto que la población está en constante movimiento, y las pequeñas variaciones que ocurren avanzado el algoritmo al mutar no son suficientemente grandes para hacer que nuevos individuos siempre pierdan incluso ante individuos que le dan altas probabilidades al óptimo (Algoritmo 13).

Algoritmo 13: Elegir sobreviviente simplificado (*ElegirSobreviviente*)

```
1  if autosuperacion = TRUE then
2      rival ← hijo
3  elegido ← aleatorio;
4  rival ← poblacion_elegido;
5  if rivalunico = TRUE then
6      rivalpool_elegido ← NULL
7  [s_rival, fs_rival, f̄_rival]=Construir(τ_rival, M, h);
8  if f_rival < f_mutant then
9      nuevapoblacion_i ← mutante;
10     evaluacion_i ← f_mutante;
11     s ← s_mutante;
12     media ← vecf_mutant;
13 nuevapoblacion_i ← rival;
14 evaluacion_i ← f_rival;
15 s ← s_rival;
16 media ← vecf_rival
```

El sistema de evaluación por promedio implica que, aunque el algoritmo recuerda la mejor solución encontrada, no recuerda al individuo que la produjo. De hecho, recordar a este individuo no tiene verdadero sentido si el encontrar la mejor evaluación hasta el momento no fue suficiente para mantener al individuo en la población; sin embargo, esto tiene la desventaja de que el método no tiene una forma de acercarse a la mejor solución encontrada si no hay un individuo que le de alta probabilidad a soluciones similares para explotar el resultado. Esta desventaja puede ser reducida si el método se alterna con algoritmo MMAS, que al terminar la exploración, empieza a explotar la mejor solución encontrada independientemente de τ.

3.4. Aplicación de sistema secuencial de Diferenciación Evolutiva y Sistema de Hormigas Max-Min

La forma en que el problema fue codificado para ser resuelto por DE usa una población similar a la forma en que se codifica el camino de feromonas en un algoritmo ACO. Aprovechando esto, es posible tomar las variables utilizadas para una técnica y aplicarla a la otra de una manera relativamente directa. De este modo, un algoritmo ACO

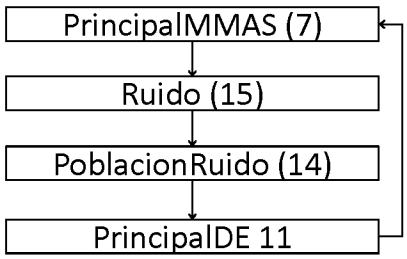

Figura 3.9: Algoritmo alternado

puede proveer a un algoritmo DE con un camino de feromonas τ cerca de converger, y con ayuda de la población de DE, explorar este camino de manera que salga del óptimo local y encuentre mejores valores con la combinación de información. Del mismo modo, un algoritmo ACO puede ayudar a explotar buenas soluciones encontradas durante el progreso del algoritmo DE que no han sido suficientemente explotadas por enfocarse en los promedios que llevan hacia buenas soluciones que hacia las mejores encontradas.

La forma que se optó por combinar los algoritmos es, simplemente, alternándolos, como se muestra en la figura 3.9. En vez de efectuar un reinicio cuando el algoritmo MMAS converja, se generará una población agregando ruido al τ final (Algoritmo 14). Una vez que se genera la población, se inicia la optimización por DE hasta llegar a un grado de convergencia mínimo o hasta un número de ciclos especificado. Por las características del tamaño de los individuos, el algoritmo DE es más lento que el MMAS. Sin embargo, su sistema de población le permite encontrar más rápidamente caminos de feromona prometedores. Por esta razón, la fase de DE no necesita llegar a una convergencia tan profunda como la fase MMAS, y le da a éste el τ con el mejor promedio, y la mejor solución encontrada, aunque esta solución no haya sido encontrada por el mejor individuo. De esta manera se le da a la fase de MMAS información, tanto para una exploración más rápida, como para ejecutar el cambio de dirección cuando se acerca a convergencia en caso de no haber mejorado el mejor valor encontrado por la fase DE. Ya se han explorado en el pasado algoritmos híbridos entre MMAS y algoritmos genéticos [18], donde uno trabaja sobre soluciones de otros como buscadores locales. Sin embargo, en este caso, no se está trabajando sobre una solución, sino sobre una representación utilizada para generar soluciones, además, los algoritmos se turnan en lugar de estar uno embebido en otro. Por estas razones, se optó por llamar a esta unión de Diferenciación Evolutiva y Colonia de Hormigas, sistema secuencial en vez de algoritmo híbrido.

Para indicar que termine la técnica DE para dar paso al algoritmo ACO, se utiliza un factor de convergencia similar al utilizado para la técnica MMAS, pero midiendo el τ con la mejor evaluación. Para mantener la mayor cantidad de parámetros similares

entre los dos algoritmos, la forma en que se calculará cf con la DE será similar que para MMAS, pero como no hay τ_{MAX} ni τ_{MIN} para DE, el valor de cf puede ser mayor a uno.

Algoritmo 14: Generar población a partir de un τ ($PoblacionRuido$)

```
1 for i ∈ Population do
2     for j ∈ s do
3         for k ∈ Mat do
4             individuo_i =Random[j,k]ruido − (0.5ruido) + τ
              individuo_i =normalizar(individuo_i)
```

Para generar el ruido que convierte el τ final del MMAS a una población para el algoritmo DE se utilizó al Algoritmo 15.

Algoritmo 15: Generar ruido a τ a partir del $cf(Ruido)$

```
1 if cf < 0.7 then
2     ruido = 0.3 else if cf < 0.95 then
3         ruido = 1 − cf else
4             ruido = 1 − cf + 0.02
```

Si durante la etapa de DE se encontró un nuevo fs_{bs}, al cambiar a MMAS se utilizará como τ al mejor individuo encontrado por DE, de lo contrario, se utilizará como τ el último valor que tuvo antes de cambiar de algoritmo, o en caso de que sea alternado por reinicio, se usará $\tau_{inicial}$.

Capítulo 4

Experimentos

En este capítulo se expondrán los experimentos finales que se utilizaron para probar el modelo y las técnicas. Primero se expondrá la estrategia, indicando lo que se quiere probar en cada tipo de experimento, seguido de la experimentación del modelo, para finalmente experimentar sobre las variables que se consideró más importante para las técnicas utilizadas.

4.1. Estrategia

La sección de experimentación se divide en cuatro partes. En la primera, el objetivo es revisar la forma en que afectan las principales variables del modelo propuesto para evaluar el WLP. Puesto que se está afectando directamente la función de evaluación, para esta sección, no se utilizará comparaciones cuantitativas entre los experimentos, sino la diferencia cualitativa entre las mejores soluciones obtenidas para cada variable modificada. En la segunda sección, se mostrarán los resultados de algunos experimentos en MMAS para analizar la forma en que se comporta con diversas variables. Una vez encontrada una combinación de variables relativamente satisfactoria, se utilizará esta combinación para los demás experimentos que impliquen aplicar la técnica de MMAS. En la tercer sección se mostrarán los resultados de algunos experimentos en DE, haciendo énfasis en las formas en que se elegirán los vectores para la etapa de selección. Las variables de los mejores encontrados se utilizarán para experimentar en la cuarta sección. En la cuarta sección se comparará un algoritmo que alterna las técnicas y se compararán con los algoritmos independientes.

4.2. Función de evaluación

Para esta sección se utilizarán dos tipos de instancias de materia (tabla 4.1). Para conocer la forma en que el Factor de Adyacencia v afecta al modelo, se diseñó un escenario con cuatro áreas de materiales de 4×4 celdas, cada una separada por pasillos y sin áreas prohibidas para revisar el comportamiento del algoritmo en problemas

Tabla 4.1: Variables en experimentos de modelo.

Número de experimento	1.1	1.2	1.3	1.4	1.5
Factor de adyacencia	1.2	2	5	2	2
Valor por espacio ocupado	25	25	25	5	5
Ahorro por similar en camino	18.75	18.75	18.75	3.75	1.25
Tipos diferentes de material	4	4	4	16	16
Costo por ocupado	10	10	10	5	5
Permutaciones posibles	6.2×10^{35}	6.2×10^{35}	6.2×10^{35}	1×10^{67}	1×10^{67}

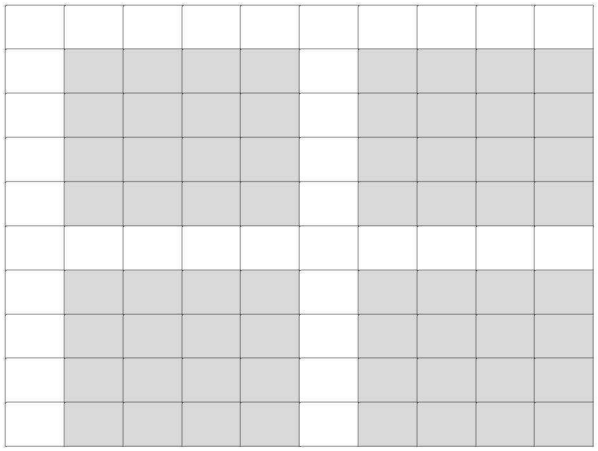

Figura 4.1: Matriz de viaje para los experimentos 1.1, 1.2 y 1.3

relativamente pequeños (figura 4.1). Hay 16 materiales de cada uno de los cuatro tipos diferentes, con ponderación decreciente. El valor de salida del material es de 0.75, mientras que el de entrada es 0.25, de modo que la respuesta correcta puede ser encontrada con simple inspección agrupando todos los materiales iguales en un área, el 1 y 2 cercanos a la salida y el resto cercanos a a la entrada. Se definió el problema de este modo porque aunque el problema dista de ser computacionalmente pequeño, por sus características, es fácil de resolver de manera visual (véase figura 4.2).

Puesto que los resultados de este experimento se utilizarían para elegir variables en lo experimentos principales posteriores, se eligió para obtener soluciones para estas instancias el algoritmo MMAS con $\rho = 0.98$ por ser el método que mejor se comportó durante experimentos previos. Todos los experimentos tienen la entrada en la sexta casilla de la primer columna, y la salida en la sexta casilla de la última columna, ambas a la altura del pasillo mostrado.

La primera variable a revisar está relacionada directamente con la propuesta para modelo de almacén: el factor de adyacencia. Al ser una variable que modifica la función de evaluación directamente, esta no puede compararse directamente con otras instancias del mismo problema, en su lugar, podemos observar la forma en que se comporta

0	0	0	0	0	0	0	0	0	0
0	4	4	4	4	0	2	2	2	2
0	4	4	4	4	0	2	2	2	2
0	4	4	4	4	0	2	2	2	2
0	4	4	4	4	0	2	2	2	2
0	0	0	0	0	0	0	0	0	0
0	3	3	3	3	0	1	1	1	1
0	3	3	3	3	0	1	1	1	1
0	3	3	3	3	0	1	1	1	1
0	3	3	3	3	0	1	1	1	1

Figura 4.2: Matriz de solución óptima

0	0	0	0	0	0	0	0	0	0
0	1	2	2	1	0	1	1	2	2
0	1	2	2	1	0	1	3	3	3
0	4	2	2	1	0	1	3	3	1
0	4	4	4	4	0	4	4	4	1
0	0	0	0	0	0	0	0	0	0
0	4	4	4	4	0	4	4	4	4
0	1	3	3	1	0	2	3	3	3
0	1	3	2	2	0	1	2	3	3
0	2	2	2	2	0	1	3	3	3

Figura 4.3: Mejor matriz de solución del experimento 1.1 ($v = 1.2$).

0	0	0	0	0	0	0	0	0	0
0	4	4	2	2	0	1	1	1	1
0	4	3	2	4	0	1	1	1	1
0	4	2	2	4	0	1	1	1	1
0	4	4	4	4	0	4	1	1	1
0	0	0	0	0	0	0	0	0	0
0	1	4	4	2	0	4	3	3	2
0	2	2	2	2	0	4	3	3	3
0	2	2	3	2	0	4	3	3	3
0	2	3	3	3	0	2	3	3	3

Figura 4.4: Mejor matriz de solución del experimento 1.2 ($v = 2$).

una función con buena evaluación utilizando diferentes factores de adyacencia v independientemente del valor de la propia evaluación. Como fue explicado en el capítulo tres, el factor de adyacencia debe ser elegido dependiendo de la cantidad de material del mismo tipo que se utilizará: mientras más material del mismo tipo exista, menor debe ser este valor. Dado que existen 16 materiales del mismo tipo, se eligió un factor de adyacencia base de 2, que será comparados con uno de 1.2 y otro de 5. Cuando se trabajó con el $v = 5$ (figura 4.5), los resultados mantuvieron unidos la mayoría de los materiales del mismo tipo, aunque el orden de éstos resultó independiente de la distancia a la entrada o a la salida. Dado que el incremento de la calificación de adyacencia aumenta conforme existen más materiales unidos, conforme avanza el algoritmo, cada incremento, en vez de disminuir como típicamente sucede en los algoritmos de optimización, aumenta drásticamente. Los resultados con $v = 1.2$ (figura 4.3) fueron en general poco favorables, pues no mantuvo unidos los materiales del mismo tipo ni acomodó a la mayoría cerca de la salida como sería el comportamiento esperado. En general, este parámetro funciona como se esperaba, pero dentro del modelo es quizás el parámetro más dificil de sintonizar de manera adecuada, pues se dificulta cuando la cantidad de materiales de cada tipo no está balanceada y puede reducir el problema de uno de optimización a uno de satisfacción de restricciones.

La siguiente variable a revisar es el ahorro por encontrarse otro material del mismo tipo en camino a la salida. Para esto se usó un mapa como el de la figura 4.6 y 4 materiales de 16 tipos diferentes; esto para reducir el efecto del factor de adyacencia v y permitir sólo una salida en línea para cada material. Para efectuar la comparación, se hicieron experimentos con 25 % y 75 % de descuento por pasar por una casilla con el mismo tipo de material al hacer el recorrido más corto. Cabe señalar que estos experimentos son más complejos que los primeros en cuanto a permutaciones posibles, pero el óptimo también es fácil de visualizar, y se muestra en la figura 4.9.

0	0	0	0	0	0	0	0	0	0
0	2	2	2	2	0	4	4	4	4
0	2	2	2	2	0	4	4	4	4
0	2	2	2	2	0	4	4	4	4
0	2	2	2	2	0	4	4	4	4
0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	3	3	3	3
0	1	1	1	1	0	3	3	3	3
0	1	1	1	1	0	3	3	3	3
0	1	3	1	1	0	3	3	1	3

Figura 4.5: Mejor matriz de solución del experimento 1.3 ($v = 5$).

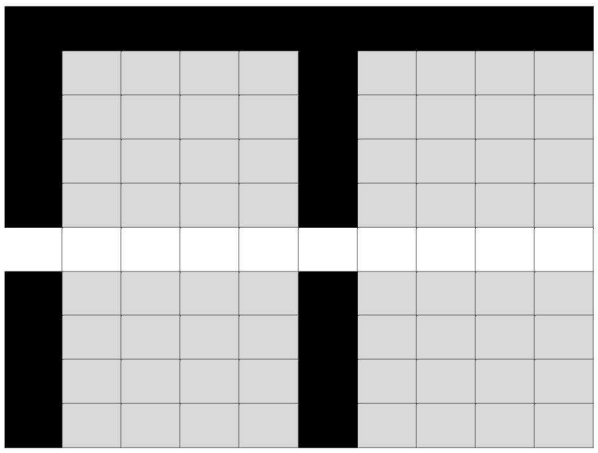

Figura 4.6: Matriz de viaje para los experimentos 1.4, 1.5.

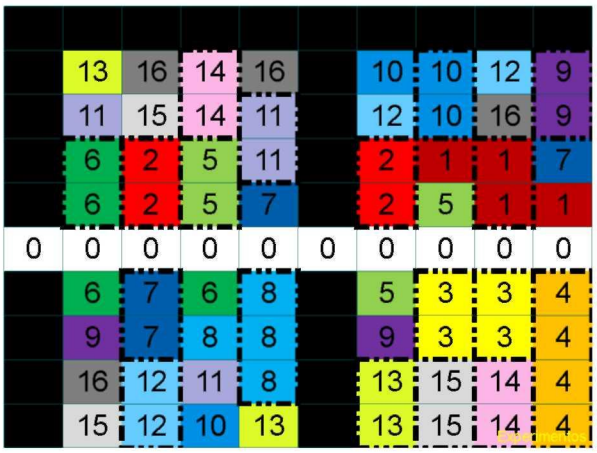

Figura 4.7: Mejor matriz de solucion del experimento 1.4 (ahorro de 3.75).

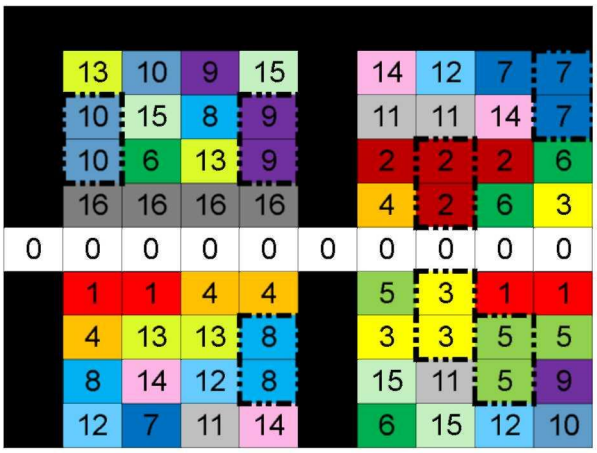

Figura 4.8: Mejor matriz de solución del experimento 1.5 (ahorro de 1.25).

Para medir la importancia de esta variable, al no poder usar la función de evaluación, se optó por contar la cantidad de pares de materiales acomodados en línea. La mejor matriz con ahorro de 25 % obtuvo 7 pares alineados, mientras que la del 75 % obtuvo 15, de los cuales uno era de tres materiales en linea y otro cuatro.

El resultado indicó que esta variable sí afecta al rendimiento, sin embargo, el algoritmo no es muy sensible a su variación. El mejor de los resultados de los experimentos con 25 % obtuvo menos de la mitad de pares que los experimentos con 75 %. Sin embargo, estos tampoco tuvieron tanto impacto como v; esto puede apreciarse en las figuras 4.8 con 13 grupos de materiales unidos y 4.7 con 17 grupos.

Figura 4.9: Óptimo de experimentos con 16 materiales.

4.3. Sistema de Hormigas Max-Min

La primer variable específica a revisar es el sistema de reinicio propuesto para evitar estancamiento [26]. La base de la idea del reinicio es que los mejores resultados se encuentran cuando el algoritmo optimizador se acerca al estancamiento, pero antes de hacerlo, y la forma en que se propone es reiniciar cuando el trazo de feromonas en todos los caminos esté muy cerca de τ_{max}, es decir, cuando $cf \approx 1$. La razón por la que es importante revisar el funcionamiento de esta función es porque la forma en que se generan las soluciones permite que existan variaciones importantes entre la solución generada y la que proponen las probabilidades.

Como primer experimento para el comportamiento del algoritmo MMAS, se comparará el método propuesto en [26] para reiniciar, el permitir durante un número largo de generaciones que el algoritmo continúe, y por último, la propuesta de permitir que el algoritmo se reinicie siempre y cuando haya pasado un determinado número de generaciones estancado. Estos experimentos se hicieron con 10 hormigas y con $\rho = 0.95$ y $\rho = 0.98$, se esperó 50 generaciones antes de volver el bs_{update} igual a 1, y 50 generaciones para permitir el reinicio una vez que el algoritmo haya convergido y bs_{update} ya sea uno.

Con esto, podemos ver que para este problema, los resultados no son significativamente sensibles a pequeños cambios en el ρ. Los mejores resultados fueron encontrados cuando se aplicó la variable de holgura, y de estos, fueron mejores los resultados con un ρ de 0.98. Esto significa que esta variable funciona y que al menos para este tipo de instancia, es mejor permitir que esté temporalmente estancado antes de efectuar el reincio al llegar a convergencia en el algoritmo.

En los experimento hechos sin holgura, cuando se tuvo mejor promedio fue cuando no hubo reinicios y con un ρ de 0.95, sin embargo, el mejor resultado (igualando al encontrado con holgura) fue utilizando con reinicios y un ρ de 0.98 en dos de las

Tabla 4.2: Comparación de promedios de resultados con Colonias de Hormigas.

Técnica	Primer reinicio	Promedio	Mejor evaluación
$\rho = 0.95$ Sin reinicio	620	620	598
$\rho = 0.95$ Reinicio básico	678	628	612
$\rho = 0.95$ Reinicio holgura	677	617	598
$\rho = 0.98$ Sin reinicio	667	667	609
$\rho = 0.98$ Reinicio básico	647	627	588
$\rho = 0.98$ Reinicio holgura	651	609	588

cinco ocasiones que se repitió la prueba (cabe señalar que para estas dos ocasiones, el primer reinicio tuvo un resultado relativamente malo). El hecho que el promedio de los experimentos hechos sin reinicio con $\rho = 0.98$ tenga como mejor encontrado una solución menor al del promedio de mejor reinicio, lo que sugiere es que cuando se tiene una velocidad de evaporación lenta, hay pocas probabilidades que se encuentre un mejor valor permitiendo que el algoritmo siga funcionando estancado. Por otro lado, el que el promedio sin reinicio cuando $\rho = 0.95$ indica que para este problema, cuando la velocidad de evaporación es alta, es mejor acercarse al punto en que el algoritmo está casi estancado y explotar estas soluciones que mantener largas exploraciones (tabla 4.2). Cómo trabajo futuro, se sugiere realizar expermentos considerando más variables.

4.4. Diferenciación Evolutiva

Los experimentos para DE fueron hechos sobre la misma instancia del problema que aquellos para MMAS. Se probaron los distintos modos de elegir el rival, los resultados de cinco corridas de cada uno se muestran en la tabla 4.3. La tercer columna se muestra el cf cuando el experimento llegó a 1500 ciclos, o bien, la cantidad de ciclos que se utilizaron para alcanzar un cf igual a 0.95. El cruze utilizado fue de entre el 50 % y el 100 % del tamaño de la matriz y también se experimentó sin cruce parcial, es decir con una probabilidade de cruce igual a 1. La razón por la que se decidió hacer experimentos sin cruce parcial fue que en pruebas preeliminares, cuando no se usaba cruce aparentaba dar mejores resultados, durante las pruebas finales, se dio a luz que cuando el cruce es completo, la convergencia es más rápida hacia óptimos locales. Cuando se le permitió al algoritmo elegir cualquier vector a modificar y ser comparado (es decir, ser *rival* o *hijo*, siempre y cuando el vector a modificar no sea de los vectores *padre*) se comportó ligéramente mejor que de la forma DE/Rand/1. Esto indica que es cuando no se está buscando que el algoritmo se comporte de manera más avariciosa, es mejor permitir que lo haga de manera libre que como se propuso originalmente en [26].

Los experimentos DE/Best/1 en los que se usó crossover funcionaron muy mal,

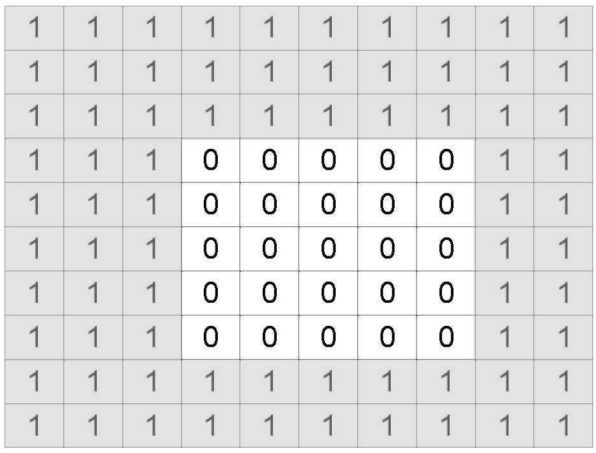

Figura 4.10: Matriz de Mutágeno Inverso.

Tabla 4.3: Comparación de tecnicas de selección de DE.

	Promedio	Mejor encontrado	$cf = 0.95$ o 2500 ciclos
Libre Cross	980	900	0.916
HijoUnico Cross	965	901	0.918
DE/Rand/1 Cross	995	912	0.88
HijoRivalUnico Cross	975	924	0.9
DE/Rand/1	1202	1129	766
HijoUnico	1218	1188	688
DE/Rand/1	1141	1115	1237
HijoRivalUnico	1179	1083	1166
DE/Best/1	1060	858	1060
DE/Best/1 Cross	1179	1150	0.24
DE/Best/1 Cross Inverso	877	793	1371

esto debido a una convergencia prematura de la población causada por una clonación rápida de óptimos locales. En [25] se recomienda que si una población llega a una convergencia prematura, se utilice una probabilidad de cruce mayor a 80 %, para permitir que cualquier punto de la matriz tenga oportunidad tanto de ser cambiado como de permanecer. Se utilizó una matriz de cruce inversa a la de la figura 3.8, que se vería parecida a la figura 4.10, y en este caso, se utilizó para cruce un complemento de 20 %.

Como se muestra en la tabla 4.3, la técnica que mejor se comporta es DE/Best/1, seguida por DE/Best/1 sin cruce y la selección *Libre* con cruce. Sin incluir DE/Best/1, que se comporta mejor con un crossover complementario, los algoritmos utilizados de DE se comportan mejor con cruce, aunque no se buscó una sintonización fina para ver cual es el más conveniente, se puede ver que cuando éste está activado, la convergencia es más lenta hacia mejores resultados.

Tabla 4.4: Algoritmo secuencial por ciclos.

	Promedio	Mejor encontrado	Iteraciones
Alternado por ciclos	610	588	5282
Alternado por reset	644	609	5686

Aún con esto, el algoritmo utilizado de MMAS funciona más rápido y con mejores resultados que cualquiera de las técnicas probadas de DE, esto porque se está utilizando como población el camino de feromonas completo, lo que genera individuos más grandes que la matriz de problema original, lo que hace que cada evaluación sea lenta en comparación a una sola generación de Colonia de Hormigas.

4.5. Algoritmos secuenciales

Se probó alternar los algoritmos DE y MMAS de dos formas: intercambiando de algoritmos cada determinada cantidad de ciclos e, intercambiando algoritmos cada vez que MMAS convergiera como alternativa al reinicio tradicional (tabla 4.5). Para el intercambio por reinicios se activó el algoritmo DE por 200 ciclos. Para generar ruido se utilizó el Algoritmo 15. Puesto que el objetivo de utilizar el DE previo al reinicio del algoritmo es explorar alrededor de la mejor solución encontrada, se eligió el mejor algoritmo DE no avaracioso que se probó, en este caso, el de selección Libre con cruce.

Durante las primeras generaciones, el algoritmo DE tiene mejores evaluaciones que el MMAS, sin embargo, éste pronto es superado. Para el algoritmo alternado por ciclos, se optó por permitir que MMAS arranque el sistema alternado, y así que el DE, que encuentra soluciones más rápido, redirija el camino de feromonas de manera distinta a como lo haría por si solo el algoritmo MMAS. En el caso del algoritmo alternado por reinicio, siempre comienza el ACO-MMAS, para el DE agregar un poco de variedad que explote-explore el camino de feromonas ya estancado.

Durante los experimentos, alternando por reinicio si llegó a permitir que la calificación encontrada mejorara un poco en algunas de las corridas, pero no significativamente con respecto a un segundo reinicio como se propone originalmente para un algoritmo MMAS. Alternando por ciclos, en cambio, si permitió una ligera mejora con respecto al algoritmo MMAS sin holgura, redirigiendo tendencias de τ cuando se estaban formando.

Capítulo 5

Conclusiones

En esta tesis se presenta un modelo para evaluar y codificar el problema de distribución de materiales de un almacén (WLP) utilizando matrices que representan diversas características de un mapa de almacén, y que son convertidas en grafos que permiten la aplicación de métodos de optimización para minimizar el transporte de material siguiendo restricciones de adayacencia y ubicación de materiales.

El modelo propuesto como función objetivo para el problema de istribución de almacén, complementado con las características propias de la forma en la que se vaya a aplicar, permite una evaluación sencilla basada en varios de los factores más comunes a tomar en cuenta en la distribución de un almacén, aunque se debe tener especial cuidado en la fuerza que se le dará a la restricción de mantener a los materiales del mismo tipo unidos, pues puede pasar de ser un problema de optimización a uno de satisfacción de restricciones. Este modelo permite modificar directamente la ponderación de diferentes variables, como la importancia relativa de los materiales, el valor de la velocidad de entrada con respecto a la salida de materiales, el grado en que se le considera benéfico que los materiales similares estén en linea, además, permite que haya tanto exceso como falta de material con respecto al cupo como dato. Aunque el modelo fue probado con áreas de pasillo fijas para este proyecto, potencialmente puede servir como base para modelos donde también los pasillos sean variables a optimizar.

Se utilizó un algoritmo de Optimización por Colonia de Hormigas llamado Sistema de Hormigas Max-Min en modelo Hipercúbico para resolver el problema combinatorio generado por el modelo, en el cual todos los resultados que son propuestos por el algoritmo son factibles. Para generar respuestas siempre factibles, conforme se consume el material por acomodar, las opciones de materiales que pueden tener las últimas casillas por asignar durante la generación constructiva de la solución no siempre corresponderán a las que tienen mayor concentración de feromona. Para reducir este efecto, se utilizó una construcción aleatoria de solución, de modo que las últimas casillas a asignar sean variables. Se propuso también una variable de holgura para permitir una mayor explotación de las probabilidades máximas del algoritmo antes de que éste sea reniciado como fue propuesto originalmente para la técnica de Sistema de Hormigas Max-Min, esta variable funcionó, permitiendo que la solución propuesta mejore un poco antes de

reiniciar a la fase de exploración.

Utilizando como base el modelo de camino de feromonas del Sistema de Hormigas Max-Min, se propuso un modelo para generar una población de caminos que podrían mejorarse aplicando algoritmos evolutivos de manera alternativa a los de Colonias de Hormigas. Se utilizó la Diferenciación Evolutiva como algoritmo para este fin, pues esta se especializa en optimizar valores continuos, requiere pocos parámetros de control y ha demostrado ser efectiva en diversas aplicaciones. Se utilizó un cruce matricial, donde aprovechando que las casillas cercanas en la matriz de solución están relacionadas, se pudieran generar esquemas útiles durante la reproducción, este esquema se utilizó exitósamente de manera inversa cuando la manera directa no funcionó correctamente. Se experimentó con varias formas de aplicar el algoritmo, especialmente en el aspecto de selección de vectores a modificar y contra los cuales evaluar, la nueva propuesta de selección libre de vectores a modificar y evaluar funcionó mejor que la propuesta original DE/Rand/1, que consiste en elegir de manera ordenada el vector a evaluar, sin embargo, fue el método DE/Best/1 con cruce inverso el que mejor se comportó dentro de los algoritmos de Diferenciación Evolutiva. Puesto que el tamaño de un individuo de camino de feromonas es más grande que el problema original, el rendimiento de los algoritmos con DE fue menor a los que utilizaron Colonias de Hormigas.

Se propusieron dos algoritmos para alternar el ACO-MMAS con DE aprovechando que ambos utilizan la misma representación para generar las propuestas de solución. La primer propuesta consistió en permitir que el algoritmo MMAS genere su camino de feromonas hasta que se le considere que éste alcanzó la convergencia. Una vez que esto sucede, en vez de reiniciarse como se propone originalmente, aplicar ruido para generar una población que el algoritmo DE intentará optimizar. Si bien con esto se pudo mejorar la solución encontrada por el algoritmo MMAS durante la primer corrida, estas mejoras no fueron superiores a las que se obtuvieron reiniciando el algoritmo. La segunda propuesta de algoritmo alternado consistió en, comenzando con un algoritmo MMAS, despues de una cantidad de ciclos fija, en adición con los reinicios, se le agregara ruido al camino de feromonas y con esto se generara una población de caminos y se empezara a utilizar un algoritmo DE durante una cantidad limitada de ciclos, para después, tomar el mejor camino de feromonas encontrado y seguir con el algoritmo MMAS. Esta propuesta se comportó un poco mejor que el mejor de los experimentos previos con las técnicas puras.

El alternar por ciclos un algoritmo MMAS con uno DE utilizando la misma representación permite redirigir las tendencias del camino de feromonas, mientras que se explota y explora al mismo el camino actual, por lo que esta puede ser una buena alternativa a un algoritmo ACO-MMAS puro, y dado que con este método, el algoritmo DE utiliza exactamente la misma representación que utiliza un algoritmo ACO en el modelo Hipercúbico, puede ser aplicado para cualquier problema al que pueda ser

aplicado un algoritmo de Colonia de Hormigas en el modelo Hipercúbico. Como trabajo futuro está el aplicar este modelo a otros problemas con revisión fina de variables para validarlo como una mejora básica a los algoritmos de Colonias de Hormigas.

Capítulo 6

Trabajo Futuro

Para este modelo del problema WLP, se utilizó como población para la diferenciación evolutiva la representación del modelo para el ACO-MMAS, sin embargo, no es exclusivo de este problema que esto pueda hacerse. Un tentativo trabajo futuro consistiría en aplicar este algoritmo a otro tipo de problemas o de funciones objetivos de prueba. Del mismo modo, probar el algoritmo secuencial con esos problemas de prueba.

En este problema en especial, al modelarlo como matriz, la cercanía entre los valores de la matriz es relevante, lo que permite que sea útil aplicar operadores como los mostrados para aislar y modificar sólo partes de la matriz. Estos operadores pueden ser útiles para otros tipos de problemas donde la posición de los valores sea relevante.

Se hicieron las pruebas que se consideraron útiles con diversas combinaciones de variables relevantes, sin embargo, estas pruebas no fueron exhaustivas y pueden ser refinadas. También puede cambiar la relevancia de las variables dependiendo de casos específicos del problema, por ejemplo, mientras menos materiales del mismo tipo haya, y si es importante para el caso específico que estén aglutinados, se debe utilizar una variable de restricción de adyacencia mayor que las utilizadas en este documento.

Se manejó un area fija específica para materiales y pasillo para dedicar un mayor enfoque en los algoritmos de optimización que en el comportamiento del modelo, sin embargo, originalmente el modelo fue pensado para que la ubicación de las áreas de pasillos y materiales también pudiera ser optimada; un primer paso es simplemente eliminar la restricción de localización de material, sin embargo, se requieren heurísticas especiales para que las soluciones propuestas sean factibles. Esta propuesta puede ser llevada a cabo como un trabajo futuro.

Bibliografía

[1] Anónimo. Software assists material handling analysis. *Industrial Engineer*, 24(5):21–22, 1992.

[2] J. Ashayeri and L. F. Gelders. Warehouse design optimization. *European Journal of Operational Research*, 21(3):285–294, September 1985.

[3] V. S. B. Rouwenhorst, B. Reuter and G. van Houtum. Warehouse design and control: Framework and literature review. *European Journal of Operational Research*, 122(3):515–533, 2000.

[4] A. E. Barrena González. *Diseño de un Sistema de Control de Inventarios para un Nivel de Servicio Determinado.* ITESM-EGADE, Monterrey, México, 2006.

[5] C. Blum. *Theoretical and practical aspects of Ant Colony Optimization.* Akademische Verlagsfesellschaft, Berlin, 2004.

[6] C. Blum, A. Roli, and M. Dorigo. HC-ACO: The hyper-cube framework for ant colony optimization. In *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, pages 399–403, 2001.

[7] C. Blum, and M. Dorigo. Deception in ant colony optimization. In *Ant Colony Optimization and Swarm Intelligence. 4th International Workshop, ANTS 2004*, pages 118–129. Springer Verlag, 2004.

[8] C-H. Pan and S-Y. Liu. A comparative study of order batching algorithms. *Omega*, 23(5), 1995.

[9] Chih-Ming, Kai-Ying and Mu-Chen. Batching orders in warehouses by minimizing travel distance with genetic algorithms. *Compututers in Industry*, 56(2):169–178, 2005.

[10] P. Corry and E. Kozan. Ant colony optimisation for machine layout problems. *Computational Optimization and Applications*, pages 287–310, 2004.

[11] H. F. Dickie. ABC inventory analysis shoots for dollars not pennies. *Factory Management and Maintenance*, 109(7):92–94, 1951.

[12] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[13] M. Dorigo, V. Maniezzo, and A. Colorni. The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 26(1):29–41, 1996.

[14] C. G. Petersen II. The impact of routing and storage policies on warehouse efficiency. *International Journal of Operations and Production Management*, 23(5):1053–1064, 1999.

[15] D. E. Goldberg. *Genetic Algorithms.* Addison-Wesley, 1989.

[16] M. M. Hassan. A framework for the design of warehouse layout. *Facilities*, 20(13):432–440, 2002.

[17] H. T. Jongen, K. Meer, and E. Triesch. *Optimization Theory.* Kluwer Academic Publishers, London, 2004.

[18] C. Lei. A hybrid optimization approach of max-min ant system and adaptive genetic algorithm for mcm interconnect test generation. *8th International Conference on Electronic Packaging Technology*, 26(14):1–4, 2007.

[19] I. Liiv. Visualization and data mining method for inventory classification. In *IEEE International Conference on Service Operations and Logistics, and Informatics, 2007. SOLI 2007.*, pages 1–6, Aug. 2007.

[20] M. Dorigo, and T. Stutzle. *Ant Colony Optimization.* MIT Press, Cambridge, Mass., 2004.

[21] M. Dorigo, L. M. Gambardella. Ant Colony System: a cooperative learning approach to the Traveling Salesman Problem. In *IEEE Transaction on Evolutionary Computation*, pages 52–66, 1997.

[22] R. Storn, and K. Price. Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical report, Berkeley, 1995.

[23] S. Routroy and R. Kodali. Differential evolution algorithm for supply chain inventory planning. *Journal of Manufacturing Technology Management*, 16(1):7–17, 2005.

[24] S. J. Russell and P. Norvig. *Artificial intelligence: a modern approach.* Prentice Hall/Pearson Education, Englewood Cliffs, N.J, 2003.

[25] R. Storn. On the usage of Differential Evolution for function optimization. In *in NAFIPS'96*, pages 519–523. IEEE, 1996.

[26] T. Stutzle and H. Hoos. Improvements on the Ant System: MAX-MIN Ant System. In *Proceedings of Artificial Neural Nets and Genetic Algorithms 1997*, pages 245–249. Springer Verlag, 1998.

[27] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, volume 2, pages 1368–1373 vol.2, Dec 1994.

[28] L. Yang and Y. Sun. Expected value model for a fuzzy random warehouse layout problem. In *2004 IEEE International Conference on Fuzzy Systems*, volume 2, pages 751 – 756, July 2004.

[29] G. Zhang. Tabu Search approaches for Multiple-Level Warehouse Layout Problems with Adjacency Constraints. *AdvOl-Report*, 6(2001), 2004.