IMPLEMENTACION EN HIDROINFORMÁTICA DE UN MÉTODO DE OPTIMIZACIÓN MATEMÁTICA BASADO EN LA COLONIA DE HORMIGAS

TRABAJO DE GRADO
Presentado como requisito parcial
para la obtención del título de
INGENIERIO CIVIL

AUTOR
CARLOS ALFREDO COY CALIXTO

DIRECTOR
Ing. NELSON OBREGÓN NEIRA Ph.D.

PONTIFICIA UNIVERSIDAD JAVERIANA
FACULTAD DE INGENIERÍA
CARRERA DE INGENIERÍA CIVIL
BOGOTÁ D.C.
2005

REGLAMENTO DE LA PONTIFICIA UNIVERSIDAD JAVERIANA.

Artículo 23. "La Universidad no se hace responsable por los conceptos emitidos por sus alumnos en sus trabajos de tesis. Sólo velará por que no se publique nada contrario al dogma y a la moral católica y por que las tesis no contengan ataques personales contra persona alguna, antes bien se vea en ellas el anhelo de buscar la verdad y la justicia".

Resolución No 13 de julio de 1946.

El proyecto de grado titulado: "*IMPLEMENTACION EN HIDROINFORMÁTICA DE UN MÉTODO DE OPTIMIZACIÓN MATEMÁTICA BASADO EN LA COLONIA DE HORMIGAS* " presentado por Carlos Alfredo Coy Calixto, en cumplimiento parcial de los requisitos exigidos para optar al título de Ingeniero Civil fue aprobado el día _____ del mes de___Junio____ de__2005___.

ING. NELSON OBREGÓN NEIRA
Director

ING. WILSON RODRÍGUEZ CALDERÓN
Evaluador

Bogotá D.C., Junio 15 de 2005

RESUMEN

El siguiente trabajo de grado tiene como objeto (1) proporcionar un marco teórico comprensivo en español del método de optimización matemática basado en la colonia de hormigas que sirva como guía facilitadora para la implementación de trabajos posteriores. El algoritmo de colonia de hormigas es un a clase de metaheurística la cual es inspirada en el comportamiento de las hormigas reales. Se presenta al comienzo una breve descripción de la evolución de los algoritmos desde sus comienzos con la inteligencia artificial, hasta los usados como base para el presente trabajo (algoritmos de colonia de hormigas para problemas continuos). Para mejorar su comprensión, se apoya en dos casos prácticos, la solución al problema del vendedor viajero y la solución a la función Goldstein–Price usando algoritmo de colonia de hormigas. (2) Desarrollar un aplicativo para la calibración de cuencas hidrológicas usando el modelo de lluvia-escorrentía formulado por Thomas (modelo a,b,c,d). El programa es validado usando tres casos de estudio uno teórico tomado del libro de Serrano y dos prácticos de las cuencas de Subachoque y Curubital en la Sabana de Bogotá. Adicionalmente se proporciona el código de programación para Matlab con comentarios línea a línea.

Palabras clave: Colonia de Hormigas, Hormiga Artificial, Feromona, Estimergia, Vendedor Viajero, Goldstein-Price, Modelo Hidrológico, Thomas, Serrano, Subachoque, Curubital.

ABSTRACT

The following work of degree has as object (1) to provide a comprehensive theoretical framework in Spanish of the optimization method math based on the ants colony that serves as facilitator guide for the subsequent projects implementation. Ant colony algorithms are a class of metaheuristics which are inspired from the behaviour of real ants. Is presented in the beginning a short description of the evolution of the algorithms from its beginning with the artificial intelligence, until the used as base for the present work (algorithms of ants colony for continuous problems). To improve its comprehension, is supported in two practical cases, the solution to the traveling salesman problem and the solution to the Goldstein-Price function using algorithm of ants colony. (2) Develop an apply for the hydrological basins calibration using the rain-runoff model formulated by Thomas (model a,b,c,d). The program is validated through three cases of study one of them consisting of the calibration of a theoretic Serrano´s book case and the other two based in a pragmatic hydrologic rain-runoff models the watershed of Subachoque and Curubital at the Savannah of Bogotá. Additionally is provided the programming code for Matlab with line commentaries to line.

Key words: Ants colony, Artificial Ant, Pheromone, stigmergy, Selling Traveler, Goldstein-Price, Hydrological Model, Thomas, Serrano, Subachoque, Curubital.

AGRADECIMIENTO

El crédito por el apoyo a mi trabajo lo debo reconocer en primer lugar al Ing. Nelson Obregón por haberme confiado un tema de tanto interés para él y la línea de investigación, y por mantener esa confianza en mí impulso tras impulso, se que no lo defraude; también al Ing. Wilson Rodríguez por brindarme flexibilidad para programar las reuniones de trabajo tan cruciales para lograr sacar este tema adelante.

A mis padres y hermanas, en especial a mi hermana Lina nunca ha dejado de apoyarme ni de mantener su disposición de colaboración pese a lo poco común del tema. A Laura por animarme a continuar a costa de su tiempo conmigo.

A Oscar y Fernando que más que mis jefes los considero mis amigos y me brindaron vía libre para programar lo que considerara necesario para priorizar éste trabajo de grado.

Y en general a todos los que comprendieron que un tema de investigación más que por requisito sólo se puede adelantar con ganas e interés, sólo hasta reunirlos se logró llevar a cabo.

CONTENIDO

Pag.

LISTA DE TABLAS

Pag.

LISTA DE FIGURAS

Pag.

INTRODUCCION

El siguiente trabajo presenta una excelente metodología de optimización matemática basada en la Colonia de Hormigas. Proporciona una guía para implementación de algoritmos basados en esta para diferentes problemas tanto continuos como discretos y demuestra su aplicación más allá del área de la ingeniería.

Este trabajo tiene un valor agregado al considerar la escasa literatura en el idioma español que se puede tomar de referencia para la implementación de nuevas aplicaciones. Por esto los objetivos del presente trabajo se centran (1) en documentar un claro marco teórico de referencia que sirva para crear nuevas aplicaciones. (2) desarrollar un programa para calibrar el modelo lluvia-escorrentia planteado por Thomas utilizando el método de optimización matemática de colonia de hormigas.

La metodología propuesta está basada en sus comienzos en el Sistema Hormiga planteado por Colorni & al. 1991, para problemas combinatorios, y posteriormente en el diseño propuesto por Johann Dréo y Patrick Siarry de concepto de comunicación horizontal (no jerarquerizado) para la solución a problemas de funciones continuas multimínimas.

Se presentan las aplicaciones desarrolladas para la comprensión del algoritmo con dominio en R^2, y la implementación de otro aplicativo para la de calibración de cuencas hidrográficas adaptado al modelo de Thomas, problema con dominio en R^6.

El primer capitulo recopila el fundamento teórico del proyecto. Se Hace la introducción en el mundo de la optimización matemática, para lo cual se reconstruye la evolución de los Algoritmos desde su comienzo computacional hasta los meta-heurísticos continuos como el requerido para hidroinformática.

El segundo capitulo plasma los objetivos del presente trabajo de grado.

En el tercer capitulo presenta el marco teórico que explica las características de las hormigas reales y artificiales, la estructura del algoritmo de colonia de hormigas, y pasos a tener en cuenta para implementar nuevas aplicaciones. Posteriormente en

el cuarto capitulo se describe el modelo de Thomas, sus ecuaciones, las variables con su significado, posibles valores y la función objetivo planteada.

1. ANTECEDENTES

1.1 MÉTODOS DE OPTIMIZACIÓN MATEMÁTICA

Antes de entrar a explicar en que consiste la optimización de colonia de Hormigas, hagamos una breve recopilación del porqué y el como de los algoritmos para solucionar problemas de optimización. Todo algoritmo remonta su origen primario en la *Inteligencia Artificial*, de la cual encontramos algunas definiciones como:

- Una de las áreas de las ciencias computacionales encargadas de la creación de hardware y software con comportamiento inteligentes.
- El estudio de las computaciones que permiten percibir, razonar y actuar
- Estudia como lograr que las máquinas realicen tareas que, por el momento, son realizadas mejor por los seres humanos.

Desde el punto de vista de los objetivos, la IA puede considerarse como parte de la ingeniería o de la ciencia:

- El objetivo ingenieríl de la IA es resolver problemas reales, actuando como un armamento de ideas acerca de cómo representar y utilizar el conocimiento, y de como ensamblar sistemas
- El objetivo científico de la IA es explicar varios tipos de inteligencia. Determinar qué ideas acerca de la representación del conocimiento, del uso que se le da a éste, y del ensamble de sistemas explican distintas clases de inteligencia.[1]

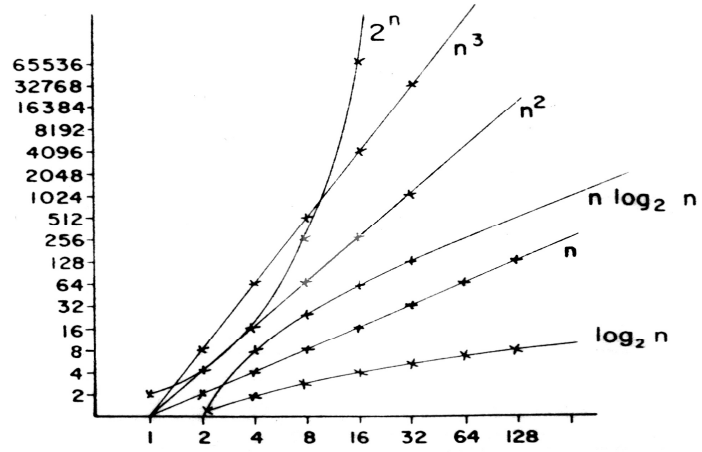
Existen problemas de optimización combinatoria complejos en diversos campos como la economía, el comercio, la ingeniería, la industria o la medicina. Sin embargo, a menudo estos problemas son muy difíciles de resolver en la práctica. El estudio de esta dificultad inherente para resolver dichos problemas es llevado a cabo por Ciencias de la Computación, ya que muchos de ellos pertenecen a la

[1] ABÁSOLO, Mª J. Introducción a la Inteligencia Artificial.-Capítulo 1. [online] http://dmi.uib.es/~abasolo/intart/1-introduccion.html

clase de problemas NP-duros, lo que significa que no existe un algoritmo conocido que los resuelva en un tiempo polinomial.[2]

"Un problema pertenece a la clase NP si puede ser resuelto en tiempo polinomial pero usando una computadora no determinística.

Figura 1. Orden de Magnitud de un Algoritmo



Problemas NP-Duros: Todos los algoritmos requeridos para resolverlos requieren tiempo exponencial en el peor caso. En otras palabras son problemas sumamente difíciles de resolver, un ejemplo: el problema del viajero, $O(n^2 2^n)$[+] [3], este problema para (n=15 ciudades) tiene 43.589'145.600 posibles soluciones.

Según Alonso, Cordón, Fernández de Viana, Herrera[4], Las técnicas existentes se pueden clasificar básicamente en algoritmos exactos o aproximados.

---

[2] ALONSO S., Cordón O., Fernández de Viana I., Herrera F. La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques. Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S. Ingeniería Informática.

[+] Para determinar el Orden de magnitud de un algoritmo suele usarse la notación "O" (big-O). Si un algoritmo tiene complejidad O(g(n)) significa que al correrlo en una computadora con los mismos datos, pero valores incrementales de n, los tiempos resultantes de ejecución serán siempre menores que |g(n)|, Los tiempos más comunes de los algoritmos son: $O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < O(2^n)$

[3] COELLO, Carlos. Introducción a la Computación Evolutiva. http://delta.cs.cinvestav.mx/~ccoello.

[4] ALONSO, Op. cit. p. 14.

Los algoritmos exactos intentan encontrar una solución óptima y demostrar que la solución obtenida es de hecho la óptima global; estos algoritmos incluyen técnicas como, por ejemplo: procesos de vuelta atrás (backtracking), Ramificación y poda (branch and bound), programación dinámica, etc. [3]. Debido a que los algoritmos exactos muestran un rendimiento pobre para muchos problemas se han desarrollado múltiples tipos de algoritmos aproximados que proporcionan soluciones de alta calidad para estos problemas combinatorios (aunque no necesariamente la óptima) en un tiempo computacional breve.

Los algoritmos aproximados se pueden clasificar en dos tipos principales: algoritmos constructivos y algoritmos de búsqueda local.

Los primeros se basan en generar soluciones desde cero añadiendo componentes a cada solución paso a paso. Un ejemplo bien conocido son las heurísticas[**] de construcción voraz o heurísticas greedy [3]. Su gran ventaja es la velocidad: normalmente son muy rápidas y, además, a menudo devuelven soluciones razonablemente buenas. Sin embargo, no puede garantizarse que dichas soluciones sean óptimas con respecto a pequeños cambios a nivel local. En consecuencia, una mejora típica es refinar la solución obtenida por la heurística voraz utilizando una búsqueda local.

Los algoritmos de búsqueda local intentan repetidamente mejorar la solución actual con movimientos a soluciones vecinas (con la esperanza de que sean mejores). El caso más simple son los algoritmos de mejora iterativos: si en el vecindario de la solución actual s se encuentra una solución mejor s', ésta reemplaza la solución actual y se continua la búsqueda a partir de s'; si no se encuentra una solución mejor en el vecindario, el algoritmo termina en un óptimo local.

Desafortunadamente, los algoritmos de mejora iterativos pueden estancarse en soluciones de baja calidad (óptimos locales muy lejanos al óptimo global). Para permitir una mejora adicional en la calidad de las soluciones, la investigación en este campo en las últimas dos décadas ha centrado su atención en el diseño de técnicas de propósito general para guiar la construcción de soluciones o la búsqueda local en las distintas heurísticas.

[**] Una heurística es una técnica que busca soluciones buenas (es decir, casi óptimas) a un costo computacional razonable, aunque sin garantizar factibilidad u optimalidad de las mismas. En algunos casos, ni siquiera puede determinar qué tan cerca del óptimo se encuentra una solución factible en particular. Reeves (1993).

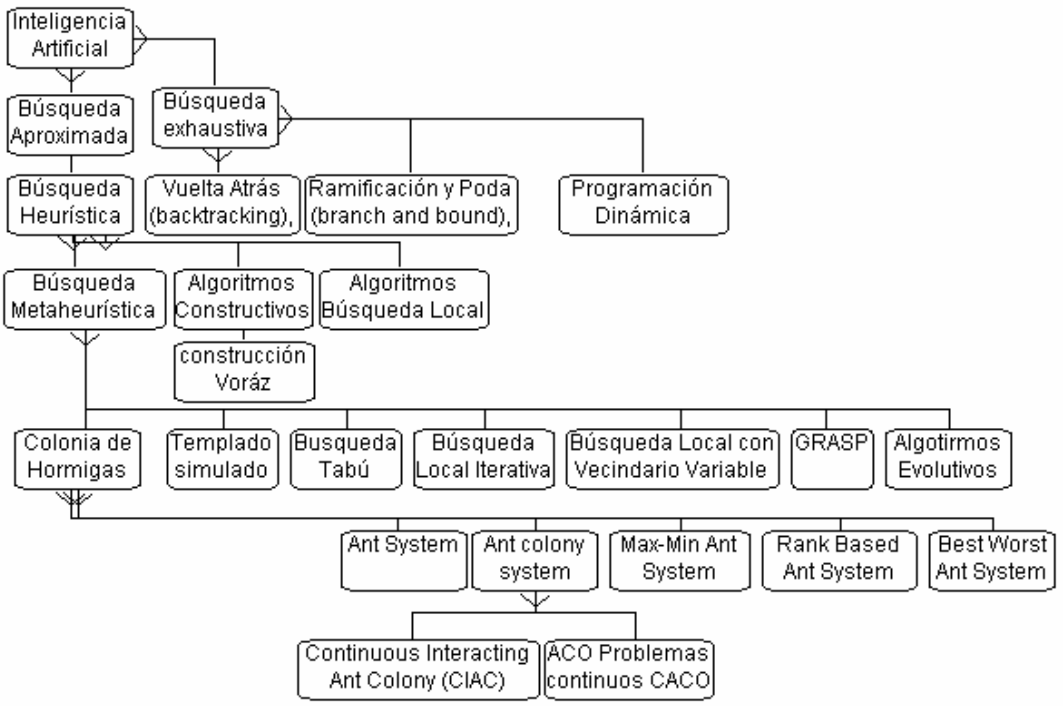
Estas técnicas se llaman comúnmente **metaheurísticas** [4] y consisten en conceptos generales empleados para definir métodos heurísticos. Dicho de otra manera, una **metaheurística** puede verse como un marco de trabajo general referido a algoritmos que puede aplicarse a diversos problemas de optimización (combinatoria) con pocos cambios significativos si ya existe previamente algún método heurístico específico para el problema. De hecho, las metaheurísticas son ampliamente reconocidas como una de las mejores aproximaciones para atacar los problemas de optimización combinatoria [5].

Las metaheurísticas incorporan conceptos de muchos y diversos campos como la genética, la biología, la inteligencia artificial, las matemáticas, la física y la neurología, entre otras. Algunos ejemplos de metaheurísticas son: *Enfriamiento simulado*, *búsqueda tabú*, *búsqueda local iterativa* (*"iterated local search"*), *algoritmos de búsqueda local con vecindario variable* (*"variable neighborhood search"*), GRASP (*"greedy randomized adaptative search procedures"*) y *algoritmos evolutivos*.

Algunos de los más importantes métodos metaheurísticos de optimización se han desarrollado a partir del estudio y la observación de la denominada ***inteligencia de enjambre***. Investigadores del comportamiento animal han propuesto muchos modelos para explicar aspectos de interés en el comportamiento social de los insectos tales como el gran nivel de estructuración que las colonias pueden llegar a tener en comparación con la formación estructural de sus individuos. Estas son capaces de adaptarse rápidamente a cambios ambientales y continúan funcionando cuando un elemento o varios de ellos fallan individualmente.

Dentro de los grupos sociales más estudiados vale la pena mencionar los enjambres de abejas, aves. Una metaheurística relativamente reciente es la *Optimización basada en Colonias de Hormigas* (OCH) (*"Ant Colony Optimization"*, ACO en inglés), la cual se inspira en el comportamiento que rige a las hormigas de diversas especies para encontrar los caminos más cortos entre las fuentes de comida y el hormiguero.

Figura 2. Evolución de los Algoritmos vía Colonia de Hormigas



2. JUSTIFICACIÓN

En nuestro ámbito de trabajo, la Ingeniería Civil, a menudo nos encontramos con la necesidad de desarrollar programas de cálculo por ejemplo en estructuras, diseño de acueductos, programación de recursos en obra, hidrología, etc., que resuelven una serie de ecuaciones previa calibración de las constantes propias de cada aplicación; en muchos casos se reduce a la tarea de probar y probar está encontrar el valor de dichas constantes. Visto de esta forma, estas aplicaciones pueden ser resueltas a través de métodos de optimización matemática.

A pesar de los avances en los métodos de optimización y de los grandes beneficios que trae su implementación, son muy pocas las aplicaciones desarrolladas para ingeniería civil, concretamente en el caso de la colonia de hormigas no es tarea sencilla encontrar información en español que facilite desarrollar nuevos aplicativos; por esta razón y con el ánimo de explorar la efectividad de la colonia de hormigas aplicada a problemas de hidroinformática en Ingeniería Civil, se decidió desarrollar el presente trabajo de grado **"Implementación en Hidroinformática de un Método de Optimización Matemática Basado en la Colonia de Hormigas"**, así se presenta no solo un programa que además de servir como ejemplo guía, permite la calibración del modelo de Thomas para la predicción de caudales de la cuenca dada, sino también una completa guía para la implementación de software para solución de problemas **tanto discretos como continuos** basados en la colonia de Hormigas.

2.1 FORMULACIÓN DEL PROBLEMA

Es Aplicable el método de Optimización de la colonia de Hormigas a la solución de problemas inversos encontrados en Hidroinformática?

2.2 OBJETIVOS

2.2.1 Objetivo principal

- Implementar el método de Optimización basado en la colonia de hormigas para la solución de problemas en hidroinformática (calibración del modelo de Balance Hídrico de Thomas).

2.2.2 Objetivos específicos

- Proporcionar un marco conceptual comprensivo para la implementación del método de Optimización basado en colonia de Hormigas.

- Desarrollar un aplicativo computacional que permita la implementación práctica de la "optimización de la colonia de Hormigas" para la calibración del modelo de Thomas.

3. MARCO CONCEPTUAL

Introducción

Se hará énfasis en la optimización matemática basada en la colonia de hormigas para lo cual se hace una breve descripción de las hormigas y su comportamiento, luego una descripción de las características de una hormiga artificial, la estructura genérica del algoritmo de hormigas, los pasos a seguir para resolver un problema por medio del ACO y para una mayor comprensión del lector se explica el sistema hormiga (Ant System) que fue uno de los primeros algoritmos desarrollados, apoyados en el seguimiento de un ejemplo del algoritmo aplicado al problema del vendedor viajero (TSP).

Subsiguientemente se entra al ACO para problemas continuos (CACO) que hemos de usar en la aplicación Hidroinformática y será explicado aplicándolo a la solución del problema de Goldstein–Price, y una posterior explicación de los cambios realizados para implementarlo en el modelo de Thomas.

Finalmente se presentan los resultados del programa, las conclusiones y recomendaciones.

3.1 LA COLONIA DE HORMIGAS

Las hormigas constituyen la familia ***Formicidae***, orden ***Hymenoptera***. El nombre científico de la hormiga cosechera roja (o agrícola) es ***Pogonomyrmex barbatus***.

La comunicación entre hormigas es muy eficaz y se realiza sobre todo por medios táctiles y químicos, aunque algunas especies emplean mecanismos vibratorios e incluso auditivos. Casi siempre una hormiga 'exploradora' alerta a la colonia y la dirección en que se lanzan sus excitadas compañeras de hormiguero puede verse afectada por uno u otro medio, dependiendo de la especie. Por ejemplo, en el caso de la hormiga faraón, común en las cocinas, la exploradora, al regresar al nido, libera un reguero de secreciones químicas hasta el alimento; de esta forma, la excitación de la exploradora es más considerable cuanto mayor sea la

concentración de alimento descubierto y, por consiguiente, mayor es el número de compañeras que se ven estimuladas a seguirla[5].

El algoritmo hormiga se ha venido proponiendo como un modelo computacional que reemplaza los tradicionales énfasis de control, pre-programación y centralización aplicables en automatización, emergencias, y distribución de funciones entre otras.

Una investigación particularmente exitosa conocida como **optimización de la colonia de hormigas** se dedica a la optimización de problemas discretos. Se ha aplicado exitosamente en gran número de difíciles problemas combinatorios como el del vendedor viajero o la asignación cuadrática.

Las hormigas han sido de gran interés de estudio por tratarse de insectos sociales, viven en colonias y su comportamiento está dirigido más a la supervivencia de la colonia que al del individuo en particular. Otro importante comportamiento de estos insectos tiene que ver con su aprovisionamiento de alimento, específicamente de como van identificando la ruta más corta entre las fuentes de alimento y su nido sin utilizar ayudas visuales[6] [1].

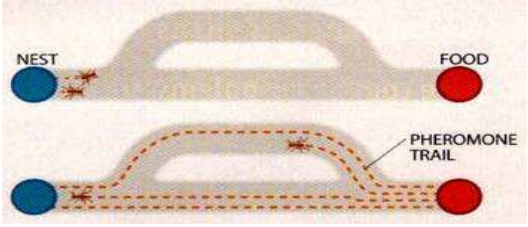
Mientras caminan desde la fuente hasta el nido y viceversa, las hormigas depositan en el suelo una sustancia llamada feromona formando en este camino un rastro de feromonas. Al comienzo de cada búsqueda generalmente, si no encuentra ningún rastro de feromona, una hormiga se mueve de manera básicamente aleatoria, pero cuando existe feromona depositada, las hormigas pueden olerla y tienen mayor tendencia a seguir el rastro cuando escogen su ruta, tendiendo a escoger, probablemente la ruta marcada por la mayor concentración de feromona. El rastro de feromonas permite a las hormigas encontrar el camino más corto de regreso al alimento o al nido, incluso también puede guiar a otros miembros del nido a la fuente de alimento. Esto ocurre gracias a que al ser los caminos más cortos, las hormigas que los siguen consiguen encontrar la comida más rápidamente, por lo que comienzan su viaje de retorno antes. Entonces, en el camino más corto habrá un rastro de feromona ligeramente superior y, por lo tanto, las decisiones de las siguientes hormigas estarán dirigidas en mayor medida a dicho camino. Además, este camino recibirá una proporción mayor de feromona por las hormigas que vuelven por él que por las que vuelven por el camino más largo. Este proceso finaliza haciendo que la probabilidad de que una hormiga escoja el camino más corto aumente progresivamente y que al final el recorrido de la colonia converja al

[5] Enciclopedia Microsoft® Encarta® 2002

[6] Hölldobler y Wilson, 1990

más corto de todos los caminos posibles. Cuando se han armado muchas rutas, la colonia de hormigas es capaz de aprovechar el rastro de feromonas dejado por otros individuos para descubrir la ruta más corta[7]. Por ejemplo en el caso de un obstáculo que se presenta en el camino de las hormigas de la comida al nido, estas bordean indistintamente los dos lados del obstáculo buscando el rastro del antiguo camino de feromona. Pero las hormigas que escogen la ruta corta alrededor del obstáculo reconstruyen el sendero de feromona más rápido. Esto se debe a que la ruta corta recibe una mayor cantidad de feromona por unidad de tiempo, y por tanto más hormigas toman la ruta corta.

Figura 3. Rastro de Feromona (variable en el tiempo)



Esta convergencia se complementa con la acción del entorno natural que provoca que la feromona se evapore transcurrido un cierto tiempo. Así, los caminos menos prometedores pierden progresivamente feromona porque son visitados cada vez por menos hormigas (ver fig. 1 y fig. 2). Sin embargo, algunos estudios biológicos han demostrado que los rastros de feromona son muy persistentes -la feromona puede permanecer desde unas pocas horas hasta varios meses dependiendo de varios aspectos distintos, como la especie de las hormigas, el tipo de suelo, lo que provoca una menor influencia del efecto de la evaporación en el proceso de búsqueda del camino más corto. Numerosos experimentos [6] muestran que, debido a la gran persistencia de feromona, es difícil que las hormigas "olviden" un camino que tiene un alto nivel de feromona aunque hayan encontrado un camino aún más corto. Hay que tener en cuenta que si se traslada este comportamiento directamente al ordenador para diseñar un algoritmo de búsqueda podemos encontrarnos con que se quede rápidamente estancado en un óptimo local.

[7] Este modo de transmisión de la información de un individuo a otro a través del medio ambiente se ha denominado Estimergia

Figura 4. Comportamiento de las Hormigas ante un Obstáculo.

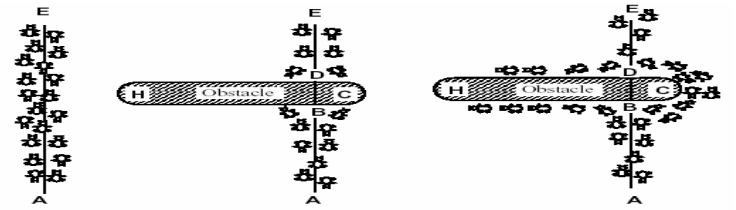
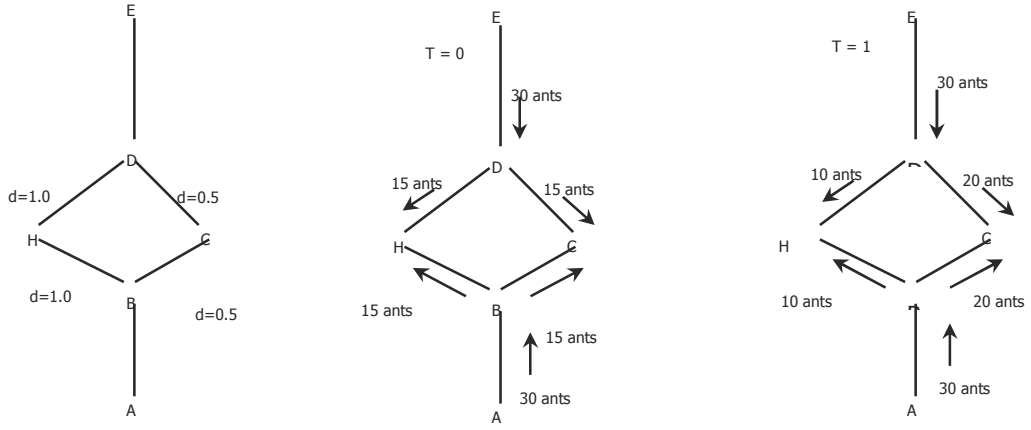


Figura 5 Evolución de experimento con obstáculos y hormigas.



Este tipo de comportamiento permite describir las bondades del mecanismo de Optimización por el cual cada individuo de la colonia hace una pequeña contribución. Los modelos computacionales han venido desarrollandose simulando procesos de búsqueda de alimento; los resultados han sido satisfactorios, mostrando que un simple modelo de probabilidad es suficiente para justificar los complejos y variados modelos colectivos.

Comencemos por llamar hormiga a un agente ínter actor, del algoritmo- Hormiga que vamos a explicar, que al igual que en una colonia verdadera cada agente representa un individuo de la colonia.

Grassé en 1946 observó que los insectos son capaces de responder a los llamados "estímulos significativos" los cuales activan una reacción genéticamente codificada. En las hormigas los efectos de esa reacción pueden tomarse como un nuevo

estimulo significante tanto para el insecto que lo produce, como para otros individuos de la colonia. La producción de un nuevo estimulo significante como consecuencia de la reacción a otro estimulo significante, determina una fórmula de coordinación de actividades y puede ser interpretada como una fórmula de comunicación indirecta.

3.2 OPTIMIZACION DE COLONIA DE HORMIGAS (OCH ACO)[8][1]

Los algoritmos de ACO son esencialmente *algoritmos constructivos*: Esto es, en cada iteración del algoritmo, cada hormiga construye una solución al problema recorriendo un grafo de construcción. Cada arco del grafo, que representa los posibles pasos que la hormiga puede dar, tiene asociada dos tipos de información que guían el movimiento de la hormiga:

- *Información heurística*, que mide la preferencia heurística de moverse desde el nodo *i* hasta el nodo *j*, o sea, de recorrer el arco o tramo a_{ij}. Se nota por η_{ij}. Las hormigas no modifican esta información durante la ejecución del algoritmo.
- *Información de los rastros de feromona artificiales*, que mide la "deseabilidad aprendida" del movimiento de *i* a *j*. Imita a la feromona real que depositan las hormigas naturales. Esta información se modifica durante la ejecución del algoritmo dependiendo de las soluciones encontradas por las hormigas. Se nota por τ_{ij}.

En esta sección se presentan los pasos que llevan a las hormigas reales al ACO. A partir de ahora debe tenerse en cuenta que los algoritmos ACO presentan una doble perspectiva:

- Por un lado, son una abstracción de algunos patrones de comportamiento naturales relacionados con el comportamiento que permite encontrar el camino más corto.

- Por otro lado, incluyen algunas características que no tienen una contrapartida natural, pero que permiten que se desarrollen algoritmos para obtener buenas soluciones al problema que se pretende resolver (por ejemplo, el uso de información heurística que guíe el movimiento de las hormigas).

Los problemas de optimización combinatoria pueden representarse como una función de la forma *G = (N, A)*, donde *A* es el conjunto de arcos que conectan el

[8] En una referencia dependiendo del autor citado se puede hablar de AS(Ant System), ACO (Ant Colony Optimization), o OCH (Optimización Basada en Colonia de Hormigas), pero en cualquiera de los tres casos se trata del mismo concepto básico.

conjunto de componentes *N*. Donde *G* se denomina función de construcción *G* Por tanto, tenemos que:

- las componentes n_i son los nodos del problema,
- los estados *d* (y por tanto las soluciones *S*) se corresponden con caminos en la gráfica solución, esto es, secuencias de nodos o arcos,
- los arcos de la gráfica, a_{ij}, son conexiones/transiciones que definen la estructura del vecindario. *d2 = <d1,s>* será vecino de *d1* si el nodo *i* es la última componente de *d1* y el arco a_{ij} existe en el gráfico,
- deben existir los costos explícitos c_{ij} asociados con cada arco, y
- las componentes o las conexiones deben tener asociados rastros de feromona τ, que representan un tipo de memoria indirecta y a largo plazo del proceso de búsqueda, como valores heurísticos η, que representan la información heurística disponible en el problema a resolver.

3.2.1 Características de la Hormiga Artificial

La hormiga artificial es un agente computacional simple que intenta construir soluciones posibles al problema explotando los rastros de feromona disponibles y la información heurística. Sin embargo, en algunos problemas, puede también construir soluciones no válidas que podrán ser penalizadas dependiendo de lo inadecuado de la solución. Tiene las siguientes propiedades[9]:

- Busca soluciones válidas de costo mínimo para el problema a solucionar.
- Tiene una memoria *LT* que almacena información sobre el camino seguido hasta el momento, esto es, *LT* almacena la secuencia generada. Esta memoria puede usarse para (i) construir soluciones válidas, (ii) evaluar la solución generada, y (iii) reconstruir el camino que ha seguido la hormiga.
- Tiene un estado inicial *dinicial*, que normalmente se corresponde con una secuencia unitaria y una o más condiciones τ de parada asociadas.
- Comienza en el estado inicial y se mueve siguiendo estados válidos, construyendo la solución asociada incrementalmente.
- Cuando está en un estado *dr = <di-1, i>* (es decir, actualmente está localizada en el nodo *i* y ha seguido previamente la secuencia *di-1*, puede moverse a cualquier nodo

j de su vecindario posible *N(i)*, definido como

$$N(i) = \left\{ s \middle| \left(a_{ij} \in A \right) y \left(< \delta_i, s > \in \tilde{\Delta} \right) \right\}$$

[9] Ver [2] páginas 11-32.

• El movimiento se lleva a cabo aplicando una regla de transición, que es función de los rastros de feromona que están disponibles localmente, de los valores heurísticos de la memoria privada de la hormiga y de las restricciones del problema.
• Cuando durante el procedimiento de construcción una hormiga se mueve desde el nodo *i* hasta el *j,* puede actualizar el rastro de feromona τ_{ij} asociado al arco a_{ij}. Este proceso se llama actualización en línea de los rastros de feromona paso a paso.
• El procedimiento de construcción acaba cuando se satisface alguna condición de parada, normalmente cuando se alcanza un estado objetivo.
• Una vez que la hormiga ha construido la solución puede reconstruir el camino recorrido y actualizar los rastros de feromona de los arcos/componentes visitados/as utilizando un proceso llamado *actualización en línea a posteriori*. Este es el único mecanismo de comunicación entre las hormigas, utilizando la estructura de datos que almacena los niveles de cada arco/componente (memoria compartida).

3.2.2 Modo de Funcionamiento y Estructura Genérica de un Algoritmo de ACO[7]

Como se ha visto en las secciones anteriores, el modo de operación básico de un algoritmo de ACO es como sigue: las m hormigas (artificiales) de la colonia se mueven, concurrentemente y de manera asíncrona, a través de los estados adyacentes del problema. Este movimiento se realiza siguiendo una regla de transición que está basada en la información local disponible en las componentes (nodos). Esta información local incluye la información heurística y memorística (rastros de feromona) para guiar la búsqueda. Al moverse por el espacio de construcción, las hormigas construyen incrementalmente soluciones. Opcionalmente, las hormigas pueden depositar feromona cada vez que crucen un arco (conexión) mientras que construyen la solución (actualización en línea paso a paso de los rastros de feromona)[10]. Una vez que cada hormiga ha generado una solución se evalúa ésta y puede depositar una cantidad de feromona que es función de la calidad de su solución (actualización en línea a de los rastros de feromona). Esta información guiará la búsqueda de las otras hormigas de la colonia en el futuro.

[10] Para la solución de los algoritmos aquí trabajados no se usará actualización en línea, no genera mayor beneficio.

Además, el modo de operación genérico de un algoritmo de ACO incluye dos procedimientos adicionales, la evaporación de los rastros de feromona y las acciones de cualidades atribuibles tan solo a hormigas artificiales y que se crean según las características de cada problema. La evaporación de feromona la lleva a cabo el entorno y se usa como un mecanismo que evita el estancamiento en la búsqueda y permite que la hormigas busquen y exploren nuevas regiones del espacio. Algunas de estas cualidades son por ejemplo observar la calidad de todas las soluciones generadas y depositar una nueva cantidad de feromona adicional sólo en las transiciones/componentes asociadas a algunas soluciones, o aplicar un procedimiento de búsqueda local a las soluciones generadas por las hormigas antes de actualizar los rastros de feromona. En ambos casos, estas funciones reemplazan la actualización en línea a posteriori de feromona y el proceso pasa a llamarse *actualización fuera de línea de rastros de feromona.*

La estructura de un algoritmo de ACO genérico es como sigue [2]:

```
1  Procedimiento Metaheuristica_ACO()
2  Inicializacion_de_parámetros
3  mientras (criterio_de_terminación_no_satisfecho)
4    Programación_de_actividades
5      Generación_de_Hormigas_y_actividad()
6      Evaporacion_de_Feromona()
7      Acciones_de _optimización adicionales() {opcional}
8    fin Programación_de_actividades
9  fin mientras
10 fin Procedimiento

1 Procedimiento Generación_de_Hormigas_y_actividad()
2  repetir en paralelo desde k=1 hasta m (numero_hormigas)
3     Nueva_Hormiga(k)
4  fin repetir en paralelo
5 fin Procedimiento

1 Procedimiento Nueva_Hormiga(id_Hormiga)
2  inicializa_hormiga(id_Hormiga)
3  L = actualiza_memoria_hormiga()
4  mientras (estado_actual ≠ estado_objetivo)
5     P = calcular_probabilidades_de_transición(A,L,W)
6     siguiente_estado = aplicar_política_decisión(P,W)
7     mover_al_siguiente_estado(siguiente_estado)
      si (actualizacion_feromona_en_linea_paso_a_paso)
```

```
8          depositar_feromona_en_el_arco_vistado()
       fin si
9      L = actualizar_estado_interno()
10    fin mientras
   si (actualizacion_feromona_en_linea_a_posteriori)
11    para cada arco visitado
12              depositar_feromona_en_el_arco_visitado()
13    fin para
   fin si
14    liberar_recursos_hormiga(id_Hormiga)
15 fin Procedimiento
```

El primer paso incluye la inicialización del los valores de los parámetros que se tienen en consideración en el algoritmo. Entre otros, se deben fijar el rastro inicial de feromona asociado a cada transición, τ_0, que es un valor positivo pequeño, normalmente el mismo para todas las componentes/conexiones, el número de hormigas en la colonia, m, y los pesos que definen la proporción en la que afectarán la información heurística y memorística en la regla de transición probabilística.

El procedimiento principal de la metaheurística ACO controla, mediante el programa base, la planificación de las tres componentes mencionadas en esta sección: (i) la generación y puesta en funcionamiento de las hormigas artificiales, (ii) la evaporación de feromona, y (iii) las acciones de optimización propias. La implementación de este programa determinará la sincronía existente entre cada una de las tres componentes.

Como se ha comentado antes, varias componentes son o bien opcionales, o bien dependientes estrictamente del algoritmo AC especifico, por ejemplo cuándo y cómo se deposita la feromona. Generalmente, la actualización en línea paso a paso de los rastros de feromona y la actualización en línea a posteriori de los rastros de feromona son mutuamente excluyentes y no suelen estar presentes a la vez ni faltar ambas al mismo tiempo (si las dos faltan, sol las acciones de optimización propias las que actualizan los rastros de feromona).

Por otro lado, hay que remarcar que el procedimiento `actualiza_memoria_hormiga()` se encarga de especificar el estado inicial desde el que la hormiga comienza su camino y, además almacenar la componente correspondiente en la memoria de la hormiga L. La decisión sobre cuál será dicho nodo depende del algoritmo específico (puede ser una elección aleatoria o una fija para toda la colonia, o una elección aleatoria o fija para cada hormiga, etc.).

Por último, cabe comentar que los procedimientos `calcular_probabilidades_de_transición` y `aplicar_política_decisión` tienen en consideración el estado actual de la hormiga, los valores actuales de la feromona visibles en dicho nodo y las restricciones del problema Ω para establecer el proceso de transición probabilístico hacia otros estados válidos.

3.2.3 Pasos a Seguir para Resolver un Problema Mediante ACO

Observando las aplicaciones actuales de la ACO, se pueden identificar algunas directivas sobre como atacar problemas utilizando esta metaheurística. Estas directivas se pueden resumir en las seis tareas de diseño que se enumeran a continuación:

Representar el problema como un conjunto de componentes y transiciones o a través de un grafo ponderado que será recorrido por las hormigas para construir soluciones.

Definir de manera apropiada el significado de los rastros de feromona τrs, esto es, el tipo de decisión que inducen. Éste es un paso crucial en la implementación de un algoritmo ACO y, a menudo, una buena definición de los rastros de feromona no es una tarea trivial. De hecho, normalmente implica tener un buen conocimiento del problema que se quiere solucionar.

Definir de manera apropiada la preferencia heurística de cada decisión que debe tomar una hormiga mientras que construye una solución, es decir, definir la información heurística ηij asociada a cada componente o transición. Hay que remarcar que la información heurística es crucial para un buen rendimiento si no existen o no pueden ser aplicados algoritmos de búsqueda local.

Si es posible, implementar una búsqueda local eficiente para el problema que se desea solucionar, porque los resultados de muchas aplicaciones del ACO a problemas de optimización combinatoria NP-duros demuestran que el mejor rendimiento se alcanza cuando se complementa con optimizaciones locales [2].

Escoger un algoritmo ACO especifico[2] y aplicarlo al problema que hay que solucionar, teniendo en cuenta, obviamente, todos los aspectos previamente comentados.

Refinar los parámetros del algoritmo ACO. Un buen punto de partida para la especificación de los valores de los parámetros es usar configuraciones de parámetros que han demostrado ser buenas cuando se aplicaban en el algoritmo ACO a problemas similares o a una gran variedad de problemas distintos. Otra posible alternativa para evitar el esfuerzo y tiempo necesarios para esta tarea es utilizar procedimientos automáticos de refinamiento de parámetros.

Debe quedar claro que los pasos descritos sólo dan una idea a general sobre la implementación del algoritmo ACO. Además, en muchas ocasiones la implementación es un proceso iterativo, donde cuando se obtiene una visión más en profundidad del problema y del comportamiento del algoritmo, deben revisarse algunas de las decisiones iniciales. Por último, se debe insistir en el hecho de que probablemente los pasos más importantes son los cuatro primeros, ya que una elección poco acertada en esos punto no suele poder arreglarse con un buen refinamiento de parámetros.

3.3 ANT SYSTEM (AS)

El AS[1] , desarrollado por Dorigo, Maniezzo y Colorni en 1991, fue el primer algoritmo de ACO. Inicialmente, se presentaron 3 variantes distintas: A*S-densidad*, A*S-cantidad* y A*S-ciclo*, que se diferenciaban en la manera en que se actualizaban los rastros de feromona. En los dos primeros, las hormigas depositaban feromona mientras que construían sus soluciones (esto es, aplicaban una *actualización en-línea paso a paso de feromona*), con la diferencia de que la cantidad de feromona depositada en el AS-densidad es constante, mientras que la depositada en AS-cantidad dependía directamente de la deseabilidad heurística de la transición ηij. Por último, en AS-ciclo, la deposición de feromona se lleva a cabo una vez que la solución está completa (actualización en línea a posteriori de feromona). Esta última variante era la que obtenía unos mejores resultados y es por tanto la que se conoce como AS en la literatura (y en el resto de este trabajo).

El AS se caracteriza por el hecho de que la actualización de feromona se realiza una vez que todas las hormigas han completado sus soluciones, y se lleva a cabo

[2] Para conocer en detalle los Algoritmos de colonia de Hormigas consulte los títulos [1,2,7] de la bibliografía.

como sigue: primero, todos los rastros de feromona se reducen en un factor constante, implementándose de esta manera la evaporación de feromona. A continuación cada hormiga de la colonia deposita una cantidad de feromona que es función de la calidad de su solución. Inicialmente, el SH no usaba ninguna acción del demonio, pero es relativamente fácil, por ejemplo, añadir un procedimiento de búsqueda local para refinar las soluciones generadas por las hormigas.
Las soluciones en el AS se construyen como sigue. En cada paso de construcción, una hormiga *k* escoge ir al siguiente nodo con una probabilidad que se calcula como:

$$p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta}}{\sum_{j \notin LT_k} [\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta}} & si \quad j \notin LT_k \\ 0 & otro\ caso \end{cases} \qquad \textbf{(F. 3.3.1)}$$

Donde $\tau(i,j)$ es la cantidad de rastro de feromona en el tramo *(i,j)* , y $\eta(i,j)=1/d_{ij}$ es la *visibilidad* una función heurística igual al inverso de la distancia entre las ciudades *i* y *j*, α es un parámetro que estima la importancia relativa de uso del sendero en función de la feromona depositada y β de cercanía en términos de *visibilidad*, *Q* es un valor escogido al azar que representa la cantidad de feromona que se repartirá al final de cada Tour tasada por la distancia total del recorrido.
si $\alpha=0$, aquellos nodos con una preferencia heurística mejor tienen una mayor probabilidad de ser escogidos, haciendo el algoritmo muy similar a un algoritmo voraz probabilístico clásico (con múltiples puntos de partida en caso de que las hormigas estén situadas en nodos distintos al comienzo de cada iteración). Sin embargo, si $\beta=0$, sólo se tienen en cuenta los rastros de feromona para guiar el proceso constructivo, lo que puede causar un rápido estancamiento, esto es, una situación en la que los rastros de feromona asociados a una solución son ligeramente superiores que el resto, provocando por tanto que las hormigas siempre construyan las mismas soluciones, normalmente óptimos locales. Por tanto es preciso establecer una adecuada proporción entre la información heurística y la información de los rastros de feromona.

Como se ha dicho, la deposición de feromona se realiza una vez que todas las hormigas han acabado de construir sus soluciones. Primero, los rastros de feromona asociados a cada arco se evaporan reduciendo todos los rastros de feromona en un factor constante:

$$\tau_{ij} \leftarrow (1-\rho) \cdot \tau_{ij} \qquad \textbf{(F. 3.3.2)}$$

donde $\rho \in (0,1]$ es la tasa de evaporación. El siguiente paso de cada hormiga es recorrer de nuevo el camino que ha seguido (el camino esta almacenado en su memoria local (lista tabú *LTk*) y deposita una cantidad de feromona $\Delta\tau^{\kappa}_{ij}$ en cada conexión por la que ha viajado:

$$\tau_{ij} \leftarrow \tau_{ij} + \Delta\tau_{ij}, \qquad \forall a_{ij} \in S_k \qquad \textbf{(F. 0.1.3.3)}$$

Donde $\Delta\tau^{\kappa}_{ij} = f(C(S_k))$, es decir, la cantidad de feromona que se deposita depende de la calidad *C(Sk)* de la solución *Sk* construida por la hormiga *k*.

Para resumir la descripción del AS, mostramos a continuación el procedimiento

`Nueva_Hormiga` para este algoritmo de ACO en particular:

1 *Procedimiento* Nueva_Hormiga(id_Hormiga)
2 k = id_Hormiga; r = generar_estado_inicial; Sk = i
3 Lk = i
4 *mientras* (estado_actual ≠ estado_objetivo)
5 para cada $s \in Nk(r)$ hacer $p_k(i,j) = \begin{cases} \dfrac{[\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta}}{\sum_{j \notin LT_k} [\tau(i,j)]^{\alpha} \cdot [\eta(i,j)]^{\beta}} \\ \end{cases}$

```
6     siguiente_est=aplicar_política_de_decisión(P, Nk(i))
7     i = siguiente_est; Sk = <Sk, i>
8     ---
9     Lk=Lk∪i
10 fin mientras
          {se ejecuta el procedimiento
         Evaporación_de_feromona() se lanza y evapora la
         feromona en cada arco aij: τij={1-ρ)·τij}
11 para cada arco aij∈Sk hacer
12   τij=τij+f(c(Sk))
13 fin para
14 liberar_recursos_hormiga(id_Hormiga)
15 fin Procedimiento
```

La línea 8 vacía se incluye para hacer notar que no existe una actualización de feromona en-línea paso a paso y que, antes de la línea 12, el demonio debe haber aplicado la evaporación de feromona. De hecho, este es un ejemplo donde la

construcción `Programación_de_actividades` interfiere con el funcionamiento de los procedimientos fundamentales de la metaheurística ACO, tal como se indicaba anteriormente.

Antes de finalizar esta sección es importante resaltar que los creadores del AS también propusieron una versión extendida del algoritmo que normalmente mejoraba los resultados obtenidos, llamada AS elitista [1]. En el AS elitista, una vez que las hormigas han depositado feromona en las conexiones asociadas a sus respectivas soluciones, el demonio realiza una deposición adicional de feromona en los arcos que pertenecen a la mejor solución encontrada hasta el momento en el proceso de búsqueda (esta solución se denomina la mejor global de aquí en adelante). La cantidad de feromona depositada, que depende de la calidad de la mejor solución global, se incrementa en un factor *e*, que se corresponde con el número de hormigas elitistas que se consideran.

3.3.1 Ejemplo Aplicativo del Ant System

Tomando como base el comportamiento de estas colonias, apliquemos ahora el Ant System-Cicle simulado a un problema como el del vendedor viajero[11].
El primer problema que fue atacado por un algoritmo de ACO fue el TSP, ya que este problema es una instancia bien conocida de un problema NP-duro, que además incluye de manera inmediata un problema de camino mínimo, haciendo por tanto que su adaptación al comportamiento real de las hormigas para resolverlo fuera una tarea casi inmediata. Desde la primera aplicación del AS en la memoria de la tesis de Dorigo en 1991, se convirtió en un problema estándar para realizar pruebas en otros modelos posteriores que ofrecían un mejor rendimiento que el AS [8].

El problema del vendedor viajero pretende encontrar el recorrido más corto para visitar todas las ciudades que tiene en su programa.

En este caso se da a un número de hormigas (*m*) el papel del vendedor viajero y cada una de ellas hará el recorrido de ciudad (*r*) en ciudad (*r+1*). Escoger a que ciudad desplazarse es una decisión que toma una función de probabilidad que depende de dos factores:
1. El número de veces que ha sido usado el tramo (ciudad origen, destino [*i,j*]) lo cual se mide por el rastro acumulado de feromona,
2. La longitud propia del tramo.

[11] Traveling Salesman Problem TSP.

Para mayor facilidad y comprensión del modelo de optimización se hará la explicación por medio del seguimiento paso a paso de un sencillo ejemplo del vendedor viajero.
Comencemos por definir el problema:

Supongamos que el vendedor en cuestión necesita promocionar y distribuir su producto por seis ciudades principales [Bogotá, Sogamoso, Bucaramanga, Barranquilla, Medellín y Cali]. Aunque un recorrido por tierra, por las características de nuestra red vial implicaría recorrer algunos tramos de la vía más de una vez, cosa que no se hará en este ejemplo; imaginaremos carreteras virtudes tales que permitan hacer el recorrido a todas las ciudades sin repetir tramos, esto es hallar la *distancia Euclidiana* de cada tramo y del recorrido total.
Para tener certeza de haber encontrado la solución óptima al problema, vamos a ubicar las ciudades en seis puntos que pertenecen a una circunferencia de radio 100km y centro en coordenadas (0,0) esto nos permite dar coordenadas a cada ciudad, calcular las distancias entre cada par y determinar la solución óptima que se logra al recorrer las ciudades en el sentido de las manecillas del reloj o el sentido contrario, pero siguiendo ese orden.

Tabla 1 Lista de Ciudades con sus Coordenadas

RADIO	CENTRO	0,0
100 km	COORDENADAS	
PUNTO	X	Y
A Bogotá	-80	60
B Sogamoso	-46	88,79
C Bucaramanga	30	95,39
D Barranquilla	90	-43,6
E Medellín	-8	-99,68
F Cali	-83	-55,78
Longitud Optima		587.859

Se definen las condiciones iniciales

Tabla 2 condiciones iniciales (distancias, visibilidad, feromona inicial)

1	2	3	4	5	6
			NC		1
ORIGEN	DESTINO		DISTANCIA	VISIVILIDAD	FEROMONA
			dij	ηij	τij(NC)
A	B	A-B	44,552	0,022	0,030
A	C	A-C	115,553	0,009	0,030
A	D	A-D	199,080	0,005	0,030
A	E	A-E	175,162	0,006	0,030
A	F	A-F	115,819	0,009	0,030
B	C	B-C	76,286	0,013	0,030
B	D	B-D	189,798	0,005	0,030
B	E	B-E	192,263	0,005	0,030
B	F	B-F	149,230	0,007	0,030
C	D	C-D	151,388	0,007	0,030
C	E	C-E	198,737	0,005	0,030
C	F	C-F	188,736	0,005	0,030
D	E	D-E	112,911	0,009	0,030
E	F	E-F	86,903	0,012	0,030

$d_{ij\ (distancia\ entre\ nodos)}$: se calcula trigonométricamente con base en las coordenadas de cada arco a_{ij},

$\eta_{ij\ (visibilidad)}$ se calcula como el inverso de la distancia

La columna 6 $\tau_{ij}(NC)$ $_{(feromona\ en\ el\ tramo)}$, se debe leer como una atracción (que tan prometedor es el nodo *j* visto desde el punto *i*) en términos de feromona, que a fin de cuentas depende de que tantas hormigas lo han usado. Para el primer ciclo es un valor aleatorio y único para cada tramo (i,j).

Y algunas constantes del ejercicio[12]

Tabla 3 definición de constantes

CONSTANTES	m = No. Hormigas	pueblos (NP)	α	β	ρ	Q	Nc max
	6	6	1	2	0,5	10	6

Se ubican aleatoriamente *m* hormigas en las diferentes ciudades.

Cada hormiga de la colonia artificial cuenta con una "lista tabú" LT_k que usan para llevar registro de las ciudades visitadas; esta memoria se actualiza después de cada paso de tiempo adicionando la nueva ciudad visitada, y se vacía al terminar cada tour. Pata t = 0, la lista tabú $\boldsymbol{LT_k}$ tendrá como primer elemento la ciudad de arranque de cada hormiga *k.*

[12] Los valores tomados para estas constantes corresponden a valores de configuración aconsejados en estudios anteriores. Ver [9] pag. 10.

Tabla 4 Lista tabú en t = 0 para cada hormiga k (A1,..,A6)

H O R M I G A S																		
A N T																		
Tiempo T	A1			A2			A3			A4			A5			A6		
	L T.	LONG. REC	$\Delta\tau^k_{ii}$	L T.	LONG. REC	$\Delta\tau^k_{ii}$	L T.	LONG. REC	$\Delta\tau^k_{ii}$	L T.	LONG. REC	$\Delta\tau^k_{ii}$	L T.	LONG. REC	$\Delta\tau^k_{ii}$	L T.	LONG. REC	$\Delta\tau^k_{ii}$
0	A			A			B			C			D			D		

Una hormiga *k* en la ciudad *i* escoge a que ciudad *j* moverse, esa ciudad *j* no debe estar dentro de su memoria de ciudades visitadas (lista tabú LT_k, $Lt_k(t=0)$ver Tabla 4 Lista tabú en t = 0 para cada hormiga k (A1,..,A6)) y aplica la formula de probabilidad **F.1**:

Tabla 5 aplicación de la función de probabilidad

1	2	3	4	5	6	7	8	9	10	11	12	13	14
T	ANT	NODO i	j	TRAMO	τij	ηij	τij^α	ηij^β	$\tau ij^\alpha * \eta ij^\beta$	$\Sigma \tau ij^\alpha * \eta ij^\beta$	$\tau ij^\alpha * \eta ij^\beta$ / $\Sigma \tau ij^\alpha * \eta_{ij}^\beta$	N j DESTINO	
0	A1	A	B	A-B	0,03	0,0224	0,0300	0,000504	0,000015	2,133E-05	0,7085	B	A-B
			C	A-C	0,03	0,0087	0,0300	0,000075	0,000002		0,1053		
			D	A-D	0,03	0,0050	0,0300	0,000025	0,000001		0,0355		
			E	A-E	0,03	0,0057	0,0300	0,000033	0,000001		0,0458		
			F	A-F	0,03	0,0086	0,0300	0,000075	0,000002		0,1048		
	A2	A	B	A-B	0,03	0,0224	0,0300	0,000504	0,000015	2,133E-05	0,7085	B	A-B
			C	A-C	0,03	0,0087	0,0300	0,000075	0,000002		0,1053		
			D	A-D	0,03	0,0050	0,0300	0,000025	0,000001		0,0355		
			E	A-E	0,03	0,0057	0,0300	0,000033	0,000001		0,0458		
			F	A-F	0,03	0,0086	0,0300	0,000075	0,000002		0,1048		
	A3	B	A	B-A	0,03	0,0224	0,0300	0,000504	0,000015	2,326E-05	0,6498	A	B-A
			C	B-C	0,03	0,0131	0,0300	0,000172	0,000005		0,2216		
			D	B-D	0,03	0,0053	0,0300	0,000028	0,000001		0,0358		
			E	B-E	0,03	0,0052	0,0300	0,000027	0,000001		0,0349		
			F	B-F	0,03	0,0067	0,0300	0,000045	0,000001		0,0579		
	A4	C	A	C-A	0,03	0,0087	0,0300	0,000075	0,000002	1,031E-05	0,2179	B	C-B
			B	C-B	0,03	0,0131	0,0300	0,000172	0,000005		0,4999		
			D	C-D	0,03	0,0066	0,0300	0,000044	0,000001		0,1269		
			E	C-E	0,03	0,0050	0,0300	0,000025	0,000001		0,0737		
			F	C-F	0,03	0,0053	0,0300	0,000028	0,000001		0,0817		
	A5	D	A	D-A	0,03	0,0050	0,0300	0,000025	0,000001	6,249E-06	0,1211	E	D-E
			B	D-B	0,03	0,0053	0,0300	0,000028	0,000001		0,1333		
			C	D-C	0,03	0,0066	0,0300	0,000044	0,000001		0,2095		
			E	D-E	0,03	0,0089	0,0300	0,000078	0,000002		0,3765		
			F	D-F	0,03	0,0058	0,0300	0,000033	0,000001		0,1596		
	A6	D	A	D-A	0,03	0,0050	0,0300	0,000025	0,000001	6,249E-06	0,1211	E	D-E
			B	D-B	0,03	0,0053	0,0300	0,000028	0,000001		0,1333		
			C	D-C	0,03	0,0066	0,0300	0,000044	0,000001		0,2095		
			E	D-E	0,03	0,0089	0,0300	0,000078	0,000002		0,3765		
			F	D-F	0,03	0,0058	0,0300	0,000033	0,000001		0,1596		

En la tabla anterior se puede observar el proceso de evaluación y selección del siguiente destino para cada hormiga *k*.

La **columna 3** indica el nodo o ciudad de origen que en el caso de la tabla anterior para un t=0 es la primera ciudad de la lista tabú Tabla 4 Lista tabú en t = 0 para cada hormiga k (A1,..,A6),

La **columna 4** para cada hormiga muestra las ciudades destino *j* permitidas es decir que no están en la lista tabú, de estas dos se genera la columna cinco de posibles recorridos para las cuales se aplica la función de probabilidad

(F. para lo cual toma datos de

Tabla 2 condiciones iniciales (distancias, visibilidad, feromona inicial) **y Tabla 3 definición de** constantes.

La **columna 11** es la sumatoria de la columna 10 para cada hormiga *k*.

La **columna 13** (siguiente destino) es el mayor valor de la columna 12 para cada hormiga.

Cada unidad de tiempo representa el viaje a otra ciudad de cada una de las *k* hormigas. En cada paso de tiempo t = t+1, se actualiza la lista tabú con la información de la columna 13 y se guarda en memoria la distancia del tramo recorrido (columna 14).

Tabla 6 Lista tabú en t = 1 para cada hormiga k (A1,..,A6)

	H O R M I G A S A N T[13]																	
Tiempo T	A1			A2			A3			A4			A5			A6		
	LT.	LONG. REC	$\Delta\tau^k_{ij}$	LT.	LONG. REC	$\Delta\tau^k_{ij}$	LT.	LONG. REC	$\Delta\tau^k_{ij}$	LT.	LONG. REC	$\Delta\tau^k_{ij}$	LT.	LONG. REC	$\Delta\tau^k_{ij}$	LT.	LONG. REC	$\Delta\tau^k_{ij}$
0	A	44,55		A	44,55		B	44,55		C	76,29		D	112,91		D	112,91	
1	B			B			A			B			E			E		

Este procedimiento se repite hasta que todas las hormigas terminan su recorrido es decir visitan las seis ciudades.

[13] LONG. REC. Es es la longitud de cada arco recorrido, cada arco esta formado por las ciudades en la lista tabú para los tiempos (t, t+1)

Tabla 7 Función de probabilidad de t=1 a t=4

1	2	3	4	5	6	7	8	9	10	11	12	13	14
T	ANT	NODO i	j	TRAMO	τ_{ij}	η_{ij}	τ_{ij}^{α}	η_{ij}^{β}	$\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}$	$\Sigma\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}$	$\tau_{ij}^{\alpha} * \eta_{ij}^{\beta} / \Sigma\tau_{ij}^{\alpha} * \eta_{ij}^{\beta}$	N j DESTINO	
										8,147E-06			
			C	B-C	0,03	0,0131	0,0300	0,000172	0,000005		0,6328		
	A1	B	D	B-D	0,03	0,0053	0,0300	0,000028	0,000001		0,1022	C	B-C
			E	B-E	0,03	0,0052	0,0300	0,000027	0,000001		0,0996		
			F	B-F	0,03	0,0067	0,0300	0,000045	0,000001		0,1654		
			B					0,000000	0,000000	8,147E-06	0,0000		
			C	B-C	0,03	0,0131	0,0300	0,000172	0,000005		0,6328		
	A2	B	D	B-D	0,03	0,0053	0,0300	0,000028	0,000001		0,1022	C	B-C
			E	B-E	0,03	0,0052	0,0300	0,000027	0,000001		0,0996		
			F	B-F	0,03	0,0067	0,0300	0,000045	0,000001		0,1654		
			A				0,0000	0,000000	0,000000	6,218E-06	0,0000		
			C	A-C	0,03	0,0087	0,0300	0,000075	0,000002		0,3613		
	A3	A	D	A-D	0,03	0,0050	0,0300	0,000025	0,000001		0,1217	C	A-C
			E	A-E	0,03	0,0057	0,0300	0,000033	0,000001		0,1573		
			F	A-F	0,03	0,0086	0,0300	0,000075	0,000002		0,3597		
1			A	B-A	0,03	0,0224	0,0300	0,000504	0,000015	1,811E-05	0,8348		
							0,0000	0,000000	0,000000		0,0000		
	A4	B	D	B-D	0,03	0,0053	0,0300	0,000028	0,000001		0,0460	A	B-A
			E	B-E	0,03	0,0052	0,0300	0,000027	0,000001		0,0448		
			F	B-F	0,03	0,0067	0,0300	0,000045	0,000001		0,0744		
			A	E-A	0,03	0,0057	0,0300	0,000033	0,000001	6,521E-06	0,1499		
			B	E-B	0,03	0,0052	0,0300	0,000027	0,000001		0,1245		
	A5	E	C	E-C	0,03	0,0050	0,0300	0,000025	0,000001		0,1165	F	E-F
							0,0000	0,000000	0,000000		0,0000		
			F	E-F	0,03	0,0115	0,0300	0,000132	0,000004		0,6091		
			A	E-A	0,03	0,0057	0,0300	0,000033	0,000001	6,521E-06	0,1499		
			B	E-B	0,03	0,0052	0,0300	0,000027	0,000001		0,1245		
	A6	E	C	E-C	0,03	0,0050	0,0300	0,000025	0,000001		0,1165	F	E-F
			E				0,0000	0,000000	0,000000		0,0000		
			F	E-F	0,03	0,0115	0,0300	0,000132	0,000004		0,6091		
										2,911E-06			
			C				0,0000	0,000000	0,000000		0,0000		
	A1	C	D	C-D	0,03	0,0066	0,0300	0,000044	0,000001		0,4497	D	C-D
			E	C-E	0,03	0,0050	0,0300	0,000025	0,000001		0,2610		
			F	C-F	0,03	0,0053	0,0300	0,000028	0,000001		0,2893		
			B					0,000000	0,000000	2,911E-06	0,0000		
			C				0,0000	0,000000	0,000000		0,0000		
	A2	C	D	C-D	0,03	0,0066	0,0300	0,000044	0,000001		0,4497	D	C-D
			E	C-E	0,03	0,0050	0,0300	0,000025	0,000001		0,2610		
			F	C-F	0,03	0,0053	0,0300	0,000028	0,000001		0,2893		
			A				0,0000	0,000000	0,000000	2,911E-06	0,0000		
			C				0,0000	0,000000	0,000000		0,0000		
	A3	C	D	C-D	0,03	0,0066	0,0300	0,000044	0,000001		0,4497	D	C-D
			E	C-E	0,03	0,0050	0,0300	0,000025	0,000001		0,2610		
			F	C-F	0,03	0,0053	0,0300	0,000028	0,000001		0,2893		
2			A				0,0000	0,000000	0,000000	3,971E-06	0,0000		
				A-			0,0000	0,000000	0,000000		0,0000		
	A4	A	D	A-D	0,03	0,0050	0,0300	0,000025	0,000001		0,1906	F	A-F
			E	A-E	0,03	0,0057	0,0300	0,000033	0,000001		0,2462		
			F	A-F	0,03	0,0086	0,0300	0,000075	0,000002		0,5632		
			A	F-A	0,03	0,0086	0,0300	0,000075	0,000002	4,426E-06	0,5053		
			B	F-B	0,03	0,0067	0,0300	0,000045	0,000001		0,3044		
	A5	F	C	F-C	0,03	0,0053	0,0300	0,000028	0,000001		0,1903	A	F-A
				F-			0,0000	0,000000	0,000000		0,0000		
			F	F-F			0,0000	0,000000	0,000000		0,0000		
			A	F-A	0,03	0,0086	0,0300	0,000075	0,000002	4,426E-06	0,5053		
			B	F-B	0,03	0,0067	0,0300	0,000045	0,000001		0,3044		
	A6	F	C	F-C	0,03	0,0053	0,0300	0,000028	0,000001		0,1903	A	F-A
			E	F-E			0,0000	0,000000	0,000000		0,0000		
			F	F-F			0,0000	0,000000	0,000000		0,0000		

1	2	3	4	5	6	7	8	9	10	11	12	13	14
T	ANT	NODO i	j	TRAMO	τij	ηij	τij^{α}	ηij^{β}	$\tau ij^{\alpha} * \eta ij^{\beta}$	$\Sigma \tau ij^{\alpha} * \eta ij^{\beta}$	$\tau ij^{\alpha} * \eta ij^{\beta} / \Sigma \tau ij^{\alpha} * \eta_{ij}{}^{\beta}$	N j DESTINO	
										3,351E-06			
			C				0,0000	0,000000	0,000000		0,0000		
	A1	D	D	D-D			0,0000	0,000000	0,000000		0,0000	E	D-E
			E	D-E	0,03	0,0089	0,0300	0,000078	0,000002		0,7023		
			F	D-F	0,03	0,0058	0,0300	0,000033	0,000001		0,2977		
			B					0,000000	0,000000	3,351E-06	0,0000		
			C				0,0000	0,000000	0,000000		0,0000		
	A2	D	D	D-D			0,0000	0,000000	0,000000		0,0000	E	D-E
			E	D-E	0,03	0,0089	0,0300	0,000078	0,000002		0,7023		
			F	D-F	0,03	0,0058	0,0300	0,000033	0,000001		0,2977		
			A				0,0000	0,000000	0,000000	3,351E-06	0,0000		
			C				0,0000	0,000000	0,000000		0,0000		
	A3	D	D	D-D			0,0000	0,000000	0,000000		0,0000	E	D-E
			E	D-E	0,03	0,0089	0,0300	0,000078	0,000002		0,7023		
3			F	D-F	0,03	0,0058	0,0300	0,000033	0,000001		0,2977		
			A				0,0000	0,000000	0,000000	4,970E-06	0,0000		
				F-			0,0000	0,000000	0,000000		0,0000		
	A4	F	D	F-D	0,03	0,0058	0,0300	0,000033	0,000001		0,2007	E	F-E
			E	F-E	0,03	0,0115	0,0300	0,000132	0,000004		0,7993		
			F	F-F			0,0000	0,000000	0,000000		0,0000		
			A	A-A			0,0000	0,000000	0,000000	1,736E-05	0,0000		
			B	A-B	0,03	0,0224	0,0300	0,000504	0,000015		0,8706		
	A5	A	C	A-C	0,03	0,0087	0,0300	0,000075	0,000002		0,1294	B	A-B
				F-			0,0000	0,000000	0,000000		0,0000		
			F	F-F			0,0000	0,000000	0,000000		0,0000		
			A	A-A			0,0000	0,000000	0,000000	1,736E-05	0,0000		
			B	A-B	0,03	0,0224	0,0300	0,000504	0,000015		0,8706		
	A6	A	C	A-C	0,03	0,0087	0,0300	0,000075	0,000002		0,1294	B	A-B
			E	F-E			0,0000	0,000000	0,000000		0,0000		
			F	F-F			0,0000	0,000000	0,000000		0,0000		
										3,972E-06			
			C				0,0000	0,000000	0,000000		0,0000		
	A1	E	D	E-D			0,0000	0,000000	0,000000		0,0000	F	E-F
			E	E-E			0,0000	0,000000	0,000000		0,0000		
			F	E-F	0,03	0,0115	0,0300	0,000132	0,000004		1,0000		
			B					0,000000	0,000000	3,972E-06	0,0000		
			C				0,0000	0,000000	0,000000		0,0000		
	A2	E	D	E-D			0,0000	0,000000	0,000000		0,0000	F	E-F
			E	E-E			0,0000	0,000000	0,000000		0,0000		
			F	E-F	0,03	0,0115	0,0300	0,000132	0,000004		1,0000		
			A				0,0000	0,000000	0,000000	3,972E-06	0,0000		
			C				0,0000	0,000000	0,000000		0,0000		
	A3	E	D	E-D			0,0000	0,000000	0,000000		0,0000	F	E-F
			E	E-E			0,0000	0,000000	0,000000		0,0000		
4			F	E-F	0,03	0,0115	0,0300	0,000132	0,000004		1,0000		
			A				0,0000	0,000000	0,000000	2,353E-06	0,0000		
				E-			0,0000	0,000000	0,000000		0,0000		
	A4	E	D	E-D	0,03	0,0089	0,0300	0,000078	0,000002		1,0000	D	E-D
			E	E-E			0,0000	0,000000	0,000000		0,0000		
			F	E-F			0,0000	0,000000	0,000000		0,0000		
			A	B-A			0,0000	0,000000	0,000000	5,155E-06	0,0000		
			B	B-B			0,0000	0,000000	0,000000		0,0000		
	A5	B	C	B-C	0,03	0,0131	0,0300	0,000172	0,000005		1,0000	C	B-C
				B-			0,0000	0,000000	0,000000		0,0000		
			F	B-F			0,0000	0,000000	0,000000		0,0000		
			A	B-A			0,0000	0,000000	0,000000	5,155E-06	0,0000		
			B	B-B			0,0000	0,000000	0,000000		0,0000		
	A6	B	C	B-C	0,03	0,0131	0,0300	0,000172	0,000005		1,0000	C	B-C
			E	B-E			0,0000	0,000000	0,000000		0,0000		
			F	B-F			0,0000	0,000000	0,000000		0,0000		

Tabla 8 Lista tabú y acumulado de distancias al terminar el tour

H O R M I G A S																	
A N T																	
Tiempo T	A1			A2			A3			A4			A5			A6	
	L. T.	LONG. REC	$\Delta\tau^k_{ii}$	L. T.	LONG. REC	$\Delta\tau^k_{ii}$	L. T.	LONG. REC	$\Delta\tau^k_{ii}$	L. T.	LONG. REC	$\Delta\tau^k_{ii}$	L. T.	LONG. REC	$\Delta\tau^k_{ii}$	L. T.	LONG. REC	$\Delta\tau^k_{ii}$
0	A	44,55		A	44,55		B	44,55		C	76,29		D	112,91		D	112,91	
1	B	76,29		B	76,29		A	115,55		B	44,55		E	86,90		E	86,90	
2	C	151,39		C	151,39		C	151,39		A	115,82		F	115,82		F	115,82	
3	D	112,91		D	112,91		D	112,91		F	86,90		A	44,55		A	44,55	
4	E	86,90		E	86,90		E	86,90		E	112,91		B	76,29		B	76,29	
5	F	115,82		F	115,82		F	149,23		D	151,39		C	151,39		C	151,39	
6	A			A			B			C			D			D		
L. TOTAL		**587,859**			**587,859**			**660,537**			**587,859**			**587,859**			**587,859**	

Al terminar sus ciclos se obtiene la longitud del viaje de cada hormiga. Como vemos en la tabla anterior, cinco de las seis hormigas lograron el recorrido óptimo de L=587.859.

Cuando todas las hormigas han completado el tour, se actualiza el rastro de feromona adicionando a cada tramo usado la sumatoria de feromona que cada hormiga aporta en los tramos de su recorrido y que es inversamente proporcional a la longitud del recorrido total. La actualización de Feromona se obtiene aplicando la siguiente formula:

$$\Delta\tau^k{}_{ij} = \frac{Q}{L_{\cdot k}} \qquad \textbf{(F. 3.3.1.1)}$$

De donde el valor de Q se toma de la tabla 3 de constantes(es un valor aleatorio), y L_k se toma de
Tabla 8 Lista tabú y acumulado de distancias al terminar el tour.

Tabla 9 Cálculo del Incremento de Feromona

1	2	3	$\Delta\tau^{k}_{ij}$														
T	ANT		A-B	A-C	A-D	A-E	A-F	B-C	B-D	B-E	B-F	C-D	C-E	C-F	D-E	D-F	E-F
0	A1	A-B	0,017														
	A2	A-B	0,017														
	A3	B-A	0,015														
	A4	C-B						0,017									
	A5	D-E													0,017		
	A6	D-E													0,017		
1	A1	B-C						0,017									
	A2	B-C						0,017									
	A3	A-C		0,015													
	A4	B-A	0,017														
	A5	E-F															0,017
	A6	E-F															0,017
2	A1	C-D										0,017					
	A2	C-D										0,017					
	A3	C-D										0,015					
	A4	A-F					0,017										
	A5	F-A					0,017										
	A6	F-A					0,017										
3	A1	D-E													0,017		
	A2	D-E													0,017		
	A3	D-E													0,015		
	A4	F-E															0,017
	A5	A-B	0,017														
	A6	A-B	0,017														
4	A1	E-F															0,017
	A2	E-F															0,017
	A3	E-F															0,015
	A4	E-D													0,017		
	A5	B-C						0,017									
	A6	B-C						0,017									
			0,100	**0,015**	**0,000**	**0,000**	**0,051**	**0,085**	**0,000**	**0,000**	**0,000**	**0,049**	**0,000**	**0,000**	**0,100**	**0,000**	**0,100**

Para calcular la Feromona que queda en el tramo, debemos evaporar la feromona inicial $\tau_{ij}(t)$ afectándola por la constante de evaporación ρ y adicionarle la feromona que acabamos de calcular para cada tramo; esto es:

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij} \qquad \textbf{(F. 3.3.1.2)}$$

Tabla 10 Actualización de Feromona

1	2	3	4	5	6	7
			NC		1	2
			DISTANCIA	VISIVILIDAD	FEROMONA	
			dij	ηij	$\tau ij(NC)$	$\tau_{ij} + \Delta\tau_{ij}$
A	B	A-B	44,5518	0,02245	0,03	0,183237
A	C	A-C	115,5528	0,00865	0,03	0,030139
A	D	A-D	199,0803	0,00502	0,03	0,015
A	E	A-E	175,1619	0,00571	0,03	0,015
A	F	A-F	115,8189	0,00863	0,03	0,100054
B	C	B-C	76,2860	0,01311	0,03	0,168098
B	D	B-D	189,7976	0,00527	0,03	0,015
B	E	B-E	192,2627	0,00520	0,03	0,015
B	F	B-F	149,2296	0,00670	0,03	0,015
C	D	C-D	151,3876	0,00661	0,03	0,132204
C	E	C-E	198,7368	0,00503	0,03	0,015
C	F	C-F	188,7362	0,00530	0,03	0,015
D	E	D-E	112,9113	0,00886	0,03	0,149215
D	F	D-F	173,4282	0,00577	0,03	0,015
E	F	E-F	86,9035	0,01151	0,03	0,183237

La columna 7 muestra la actualización de Feromona, este valor será el valor de Feromona que se toma para evaluar el ciclo II o segundo tour. Con estos parámetros la hormiga A3 alcanza el recorrido óptimo al igual que sus compañeras.

La colonia de hormigas puede interpretarse como un sistema de aprendizaje con refuerzo, en que los refuerzos modifican la fuerza de conexiones entre las ciudades. De hecho, las anteriores fórmulas dictan que una hormiga puede con la probabilidad q_o , aprovecharse de la experiencia acumulada por la colonia en términos de huella de feromona (la huella de feromona tenderá a crecer en los tramos utilizados que pertenecen a circuitos cortos, haciéndolos más deseables), ó con probabilidad (β), aplique una exploración parcial(exploración parcial hacia los tramos cortos) de nuevos caminos escogiendo al azar la ciudad para moverse, con una distribución de probabilidad que es una función de los tres parámetros, la huella de feromona acumulada, la visibilidad, y la memoria activa LT_k.

3.3.2 Algoritmo Ant Colony usando el Concepto de no jerarquización Heterarchical) aplicado a la Optimización de Funciones Continuas Multiminimas

Hasta ahora se ha profundizando en los conceptos de la Optimización de la Colonia de Hormigas, sin embargo aún no se ha tocado el siguiente nivel que corresponde a la solución de problemas continuos y que finalmente es el que aplicaremos al modelo de hidroinformática. Para tal fin el modelo de algoritmo a implementar estará basado en el desarrollado por Johann Dréo and Patrick Siarry[10], es un algoritmo tipo CIAC "Continuous Interacting Ant Colony", para la solución de problemas de optimización de funciones continuas multi-mínimas pero con las mejoras que se explicaran más adelante.

Características del Algoritmo

El término Heterarchy fue introducido por Wilson en 1988 para describir como la información fluía dentro una colonia de hormigas (Wilson & Holldobler 1988):
Una colonia de hormiga es un tipo especial de jerarquía, que finalmente puede ser llamado un heterarchy. Esto significa que las propiedades de alto nivel afectan los niveles más bajos en algún grado, pero induciendo actividad en bajo nivel se retroalimenta para influir los niveles más altos.
El concepto más importante aquí es que información fluye en la colonia, cada hormiga puede comunicarse con cualquier otra sin distinción de rango. Para explicar estos conceptos debemos referirnos a los canales de comunicación. Este algoritmo CIAC[14], contempla dos canales de comunicación diferentes:

Comunicación Indirecta: Canal de estimergia, como se definió anteriormente es la comunicación que tienen las hormiga a través del medio ambiente en forma de Feromona acumulada en el trayecto.

- Alcance: Cuando una hormiga pone un punto de feromona, todas las hormigas pueden posteriormente percíbalo.
- Memoria: la información persiste en el sistema durante un cierto período de tiempo, independientemente de los agentes.
- Integridad: la información es modificada por el tiempo, para reproducir la feromona evaporación

- Comunicación Directa: Es un concepto de comunicación a través de mensajes generados y recibidos por las hormigas

[14] Continuous Interacting Ant Colony CIAC

- Alcance: Cuando una hormiga envía un mensaje, una única hormiga puede percibirlo.
- Memoria: La información persiste en el sistema durante un período seguro de tiempo, bajo la forma de memoria de hormiga.

Integridad: Las informaciones almacenadas son estáticas.

El algoritmo fue planteado para trabajar con estos dos canales de comunicación, se inspiro no en los modelos tradicionales que se enfocaban en simular como las hormigas encuentran el camino más corto del mido a la fuente de alimento, sino en simular la forma en que las hormigas abandonan el nido y exploran el terreno, posteriormente y a medida que van regresando al nido, vuelven a salir con preferencia a la orientación de mejor fuente de alimento.
Este concepto de comunicación directa que se haría en el nido está basado en la modelación de la tropolaxis[3].

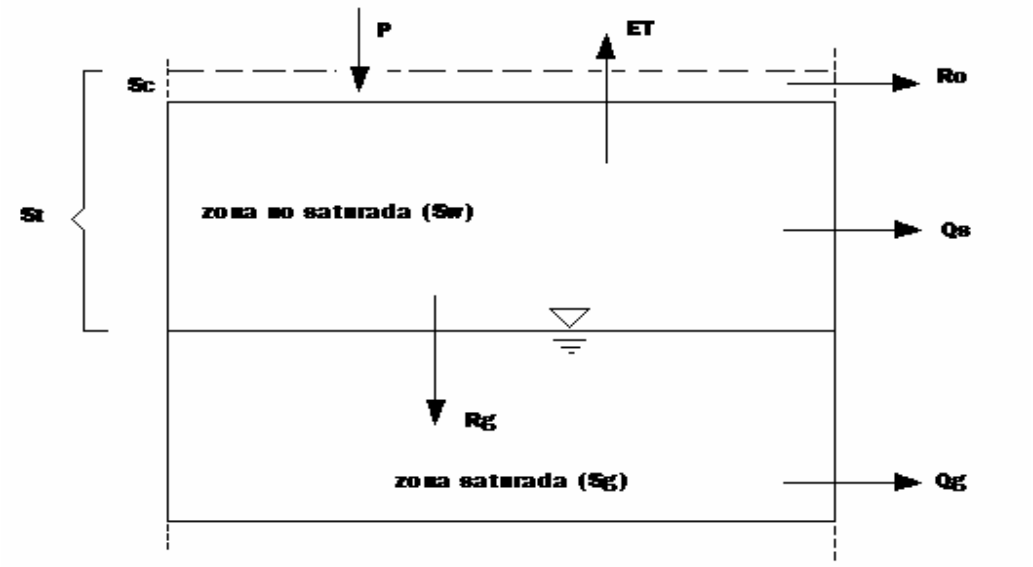
[3] Es el intercambio del liquido alimenticio entre hormigas.

4. MODELO DE THOMAS[15] (modelo abcd)

El método de Thomas establece el balance hídrico hallando el caudal efluente en una cuenca determinada a partir de la ecuación de continuidad a un volumen de control con almacenamiento $S_w + S_c$, teniendo en cuenta 4 parámetros que se calibraran mediante el problema inverso y que variarán de acuerdo al tipo de zona en estudio.

De acuerdo con Obregón (2002), para explicar las variables que intervienen en el modelo, supongamos una parte del suelo donde se identifican tres zonas (fig. 6), a primera es la parte superficial que corresponde a los diferentes componentes del almacenamiento superficial (S_C). La segunda es la zona no saturada con almacenamiento (S_W), allí se genera un caudal subterráneo o subsuperficial (Qs) que Thomas desprecia por que en comparación con la precipitación resulta bastante pequeño. Finalmente la tercera zona o zona saturada con almacenamiento (Sg), y debido a que el espesor de la zona Sc es muy pequeño con respecto a $St = Sc + Sw$, se puede asumir para $St=Sw$.

Figura 6 Caracterización de la ecuación de continuidad para el caudal



[15] Obregón N, Fragala F, Blanco A, Gómez L. Implementación del modelo de Thomas en la cuenca alta del río Checua para la estimación de la recarga (Sabana de Bogotá, Cundinamarca, Colombia) XV Seminario nacional de Hidráulica e Hidrología. Medellín, Agosto 2002

Para la primera zona Sw, se parte de la ecuación de continuidad y se tiene:

$$P_i - ET_i - R_{gi} - R_{Oi} = \frac{\Delta S_W}{\Delta_t} = S_{wi} - S_{wi-1} \qquad \textbf{(F. 4.1)}$$

Donde:
P_i , es la precipitación media de la zona en un tiempo i.
ET_i , es la Evapotranspiración media de la zona en un tiempo i.
I_i es la infiltración en un tiempo i.
$\Delta S_w/\Delta t = (S_i - S_i\text{-}1)$ es el cambio del contenido de humedad a la zona no saturada en un tiempo i.
R_{oi} es la escorrentía superficial o directa en un tiempo i.

Para la segunda zona, Thomas considera $Ii=Rg_i$, lo que se infiltra es lo que recarga el acuífero, debido a que en el modelo se desperdicia la escorrentía subsuperficial q_{sub}.

Para la zona saturada.

$$R_{gi} - Q_{gi} = \frac{\Delta S_g}{\Delta_t} = S_{g_i} - S_{g_i-1} \qquad \textbf{(F.4.2)}$$

Rgi es la recarga del acuífero.
Qgi es el caudal subterráneo que alimenta el río.
$\Delta S_g/\Delta t =$ es el cambio de almacenamiento en esta zona.

Partiendo de la ecuación (F.4.1) y sabiendo que I_i igual a R_{gi}, se agrupa de la siguiente manera:

$$(P_i + S_{wi-1}) - (ET_i + S_{wi}) = R_{oi} + I_i \qquad \textbf{(F.4.3)}$$

igualamos:

$$W_i = P_i + S_{wi-1} \qquad \textbf{(F.4.4)}$$

Siendo W_i = Agua disponible en in intervalo de tiempo *i*

$$Y_i = ET_i + S_{wi} \quad \textbf{(F.4.5)}$$

Y_i= Humedad inicial al comienzo de cada intervalo *i*

Por tanto:

$$W_i - Y_i = R_{oi} + I_i \quad \textbf{(F.4.6)}$$

Por métodos empíricos, Thomas establece que:

$$S_{wi} = Y_i * e^{-ETP_i / b} \quad \textbf{(F.4.7)}$$

mostrando que con el paso del tiempo la humedad del suelo asume un decaimiento exponencial, a medida que aumenta la evapotranspiración potencial (*ETP_i)*.
Además establece que la variable Yi está dada por la siguiente expresión no lineal

$$Y_i = \frac{(W_i + b)}{2a} - \left[\left(\frac{W_i + b}{2a} \right)^2 - \frac{W_i * b}{a} \right]^{0.5} \quad \textbf{(F.4.8)}$$

Siendo a y b parámetros del modelo, donde:
a (0≤1) refleja la tendencia de la escorrentía a ocurrir antes de que el suelo este completamente saturado.
b es el límite superior de la suma de evapotranspiración real y la humedad del suelo

$$I_i = c * (W_i + Y_i) = R_{gi} \quad \textbf{(F.4.9)}$$

c , según Thomas es un coeficiente de reparto que ayuda a diferenciar la escorrentía directa (*R0)* de la recarga subterránea (*Rg=I)* .

$$Q_{gi} = d * S_{gi} \quad \textbf{(F.4.10)}$$

d es parámetro de almacenamiento que representa la fracción de acuífero que viene a ser descargado al río (caudal subterráneo).

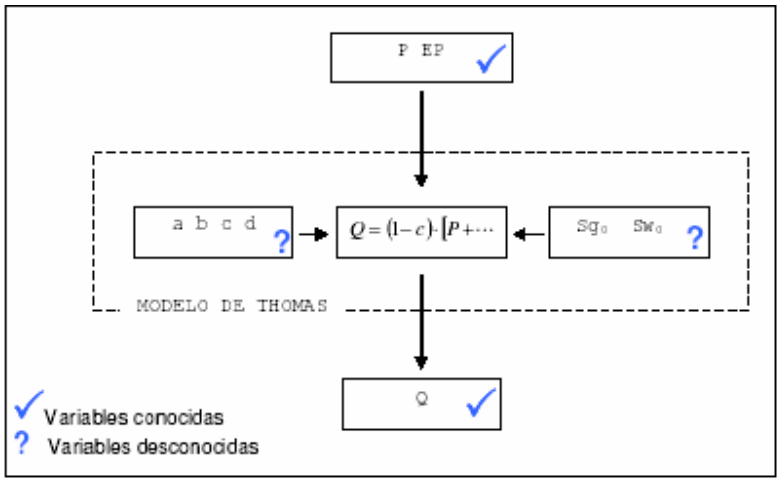
Sustituyendo la ecuación (F.4.10) en (F.4.2) obtenemos el almacenamiento de agua subterránea:

$$S_{gi} = \frac{(S_{gi-1}) + R_{gi}}{1+d} \qquad \text{(F.4.11)}$$

El Caudal Simulado final es:
$Q_{si}=R_{oi}+Q_{gi}$

Estas 5 penúltimas ecuaciones presentan 4 coeficientes (a,b,c,d), que variarán según la zona de estudio dependiendo de las condiciones geológicas e hidroclimatológicas de la zona.
Mediante software con métodos iterativos y suponiendo un valor inicial S_{wo} y S_{go} se hallan estos 4 parámetros, hallando el caudal simulado de la subcuenca que debe ser equivalente al suministrado por la estación metereológica[16].

Figura 2. *Variables en el problema de la calibración del modelo de Thomas.*



Las entradas (precipitaciones y evapotranspiraciones) y las salidas (caudales) son datos obtenidos de registros históricos. Estos datos se les conoce también como *precipitaciones observadas*, *evapotranspiraciones observadas* y *caudales observados*.

La calibración del modelo de Thomas se logra cuando se lleguen a establecer los valores de a, b, c, d, $Sw0$ y $Sg0$ tales que al ingresar la precipitación observada P y la evapotranspiración observada EP, se obtenga un caudal $Qsim$ igual al caudal observado $Qobs$.

[16] OLARTE RAFAEL, 2003. p. 82.

En trabajos anteriores de investigación como el de Alley[17] se sugieren rangos entre los cuales pueden variar los parámetros, ampliándolos un poco se fijan los espacios de busque de cada variable y la precisiones respectivas.

Tabla 11 Rangos y precisiones de la variable de calibración

Variable	Valor mínimo	Valor máximo	Rango de datos	Precisión	No Datos
a	0,8	1	0,2	0,001	201
b	10	350	340	1	341
c	0,001	0,9	0,899	0,001	900
d	0,001	1	0,999	0,001	1000
Sw_0	0	500	500	1	501
Sg_0	0	500	500	1	501

Para lograr una buena calibración, no se trabaja con un solo valor de P, de EP y de Q en un período de tiempo específico. Se trabaja con varios valores correspondientes a varios períodos de tiempo(preferiblemente valores mensuales de todo un año). Así, en realidad la calibración se logra cuando se logre reducir a cero algunas de las siguientes funciones (llámense *funciones objetivo*)[18]:

$$F(\theta) = \frac{1}{n} \cdot \sqrt{\sum_{i=1}^{n} [Qobs_i - Qsim_i(\theta)]^2} \qquad \text{(F.4.12)}$$

$$F(\theta) = \frac{1}{n} \cdot \sum_{i=1}^{n} |Qobs_i - Qsim_i(\theta)| \qquad \text{(F.4.13)}$$

$$F(\theta) = \max |Qobs_i - Qsim_i(\theta)| \qquad \text{(F.4.14)}$$

$$F(\theta) = \frac{1}{n} \cdot \sum_{i=1}^{n} (Qobs_i - Qsim_i(\theta)) \qquad \text{(F.4.15)}$$

en donde
$Qobsi$ = caudal medido (obtenido de los archivos)
$Qsimt(\theta)$ = caudal calculado por el modelo
θ = conjunto (vector) de los parámetros y condiciones iniciales empleados por el modelo
n = número de caudales
La función objetivo que se escogió fue la norma cuadrática normalizada (f. 4.12)

[17] Alley. 1984. p. 1142 Tabla 2.
[18] SOROOSHIAN, GUPTA and BASTIDAS. 1998. p. 16.

5. DESARROLLO DEL APLICATIVO COMPUTACIONAL

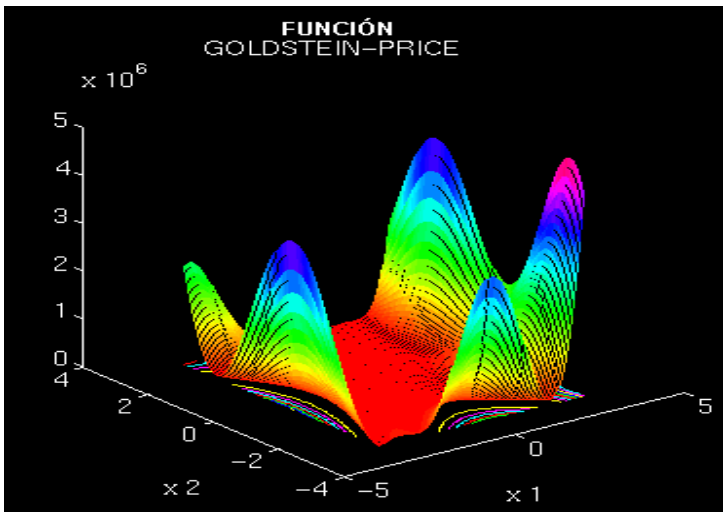
La manera más sencilla de entender los diferentes pasos del algoritmo es a través de un ejemplo como se demostró en el caso del vendedor viajero, para este algoritmo la función a solucionar como explicación es la Función Goldstein – Price.

La función Goldstein – Price es una función test para optimizaciones globales, su ecuación es:

fG(x1,x2) = [1+(x1+x2+1)^2·(19-14·x1+3·x1^2-14·x2+6·x1·x2+3·x2^2)] ·[30+(2·x1-3·x2)^2·(18-32·x1+12·x1^2+48·x2-36·x1·x2+27·x2^2)]

El objetivo de la calibración es encontrar el mínimo global que se encuentra en [0,-1] donde la función fG(0,-1) = 3

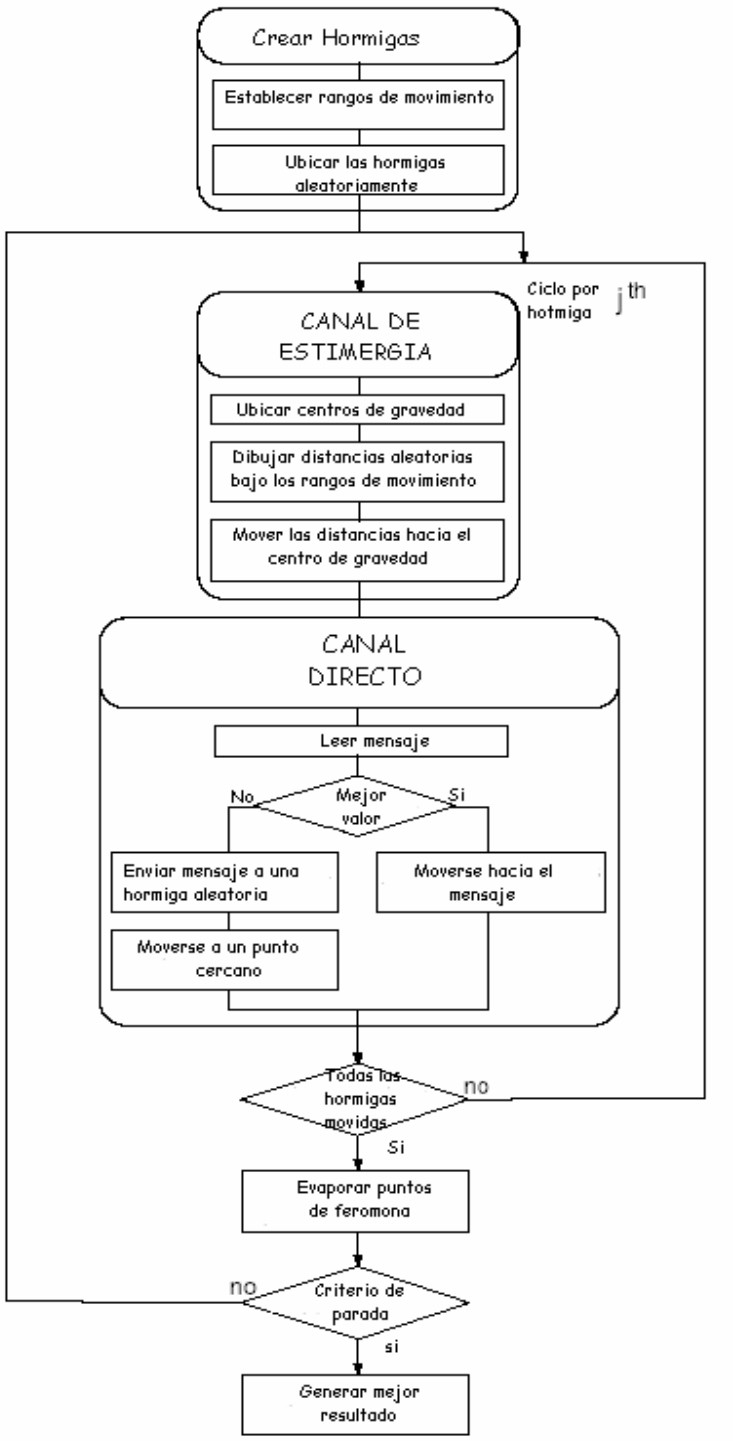
Figura 7. Función Goldstein - Price



Implementación

Para la solución de la función se amplió el espacio de búsqueda de la solución a [-5<x1<5, -5<x2<5].

Figura 8. Diagrama de flujo del algoritmo CIAC



Algunas variables con las que se puede jugar para optimizar tiempos de ejecución:

Número de hormigas (m)
Número de mensajes (nmens)
Número de ciclos (CICLO)
Área máxima de búsqueda por hormiga
(se expresa como un radio de avance y se almacena en rmov) (rmov)
Tan solo se cambia el valor máximo de exploración, ya que para las exploraciones subsiguientes el área de búsqueda irá disminuyendo conforme vaya mejorando la función objetivo.

El esquema general del algoritmo es como se describe a continuación:

```
1  Procedimiento Metaheurístico CACO()

2       Inicialización de parámetros
3    Generación de hormigas (ubicación aleatoria f.pos)
4          Evaluación de Aptitud inicial    (f.hormiguero(aptitud(goldsteinprice)))
(Q)
5          Evaluación de los mensajes iniciales (f.mensajes), trabaja como          la
feromona.
6          for (número de iteraciones)
7         Programación de actividades
8              Distancia de avance (f.distancia(aptitud))
9            CANAL DE ESTIMERGIA
10             Cálculo de Centros de Gravedad ()
11             Evaluación de aptitud a los centros de gravedad
12             Acciones de optimización adicionales()   {opcional}
13           MOVIMIENTO DE HORMIGAS
14             leer mensaje
15             Si aptitud(mensaje)>aptitud(centro de gravedadj)
16             Local= coord. destino – coord. Hormiga j
17             [dire(1),dire(2)]=cart2pol(local(1),local(2))
18             Vdesp=0: presición :dire(2)
19             for  p=1:length(Vdesp)
20             [dx1,dx2]=pol2cart(dire(1),Vdesp(p))
21             x1=pos(j)+dx1;      x2=pos(j)+dx2
22             p intermedio=aptitud(goldsteinprice(x1,x2))
23             guardar mejor solución
24             guardar resultados de hormigas (f.hormiguero)
```

25 guardar los mensajes de las hormigas que mejoraron
26 fin for hormigas
27 fin for ciclos
28 fin Procedimiento

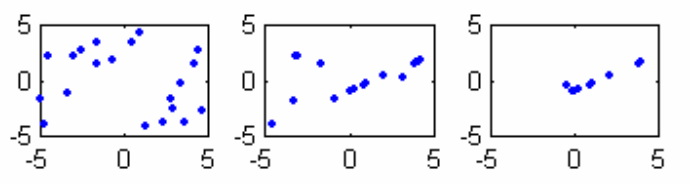
Cada paso se trabaja de la siguiente forma[19]:

a. Inicializar variables.

b. Crear hormigas: las hormigas se crean al comienzo del programa y se asume que vivirán hasta encontrar la mejor solución. No mueren ni se crean nuevas hormigas.

Recuérdese que este algoritmo simula la forma en que las hormigas exploran el terreno después de abandonar el nido, por ello la exploración inicial (sin conocimiento de fuentes de alimento) se debe hacer sobre todo el terreno; por lo tanto las hormigas se ubican aleatoriamente sobre la superficie a explorar. En este caso se hizo dando valores aleatorios a las coordenadas x1 y x2(todos dentro de los rangos de exploración), la mejor forma es a través de una matriz que contenga la información de coordenadas de todas las hormigas. **Pos(2,m)**♣

Figura 9. Variación de posición en tres ciclos consecutivos



[19] Se recomienda al lector seguir la explicación observando el código del programa y los comentarios de línea para mayor comprensión.

♣ Los nombres en sostenida y alineados a la derecha representan variables o matrices importantes y en paréntesis se especifica su dimensión.

Figura 10 Varición de la posición en diez ciclos

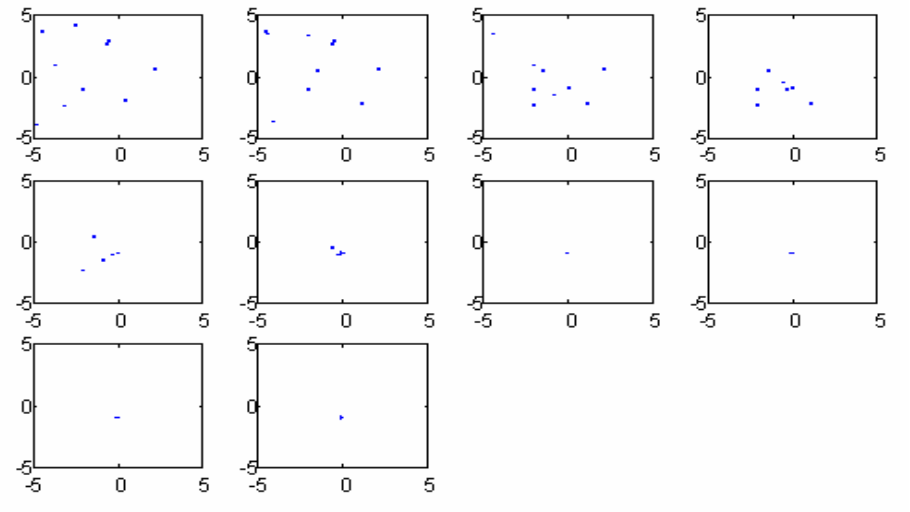


Figura 11 Ubicación de las hormigas al comienzo de la optimización

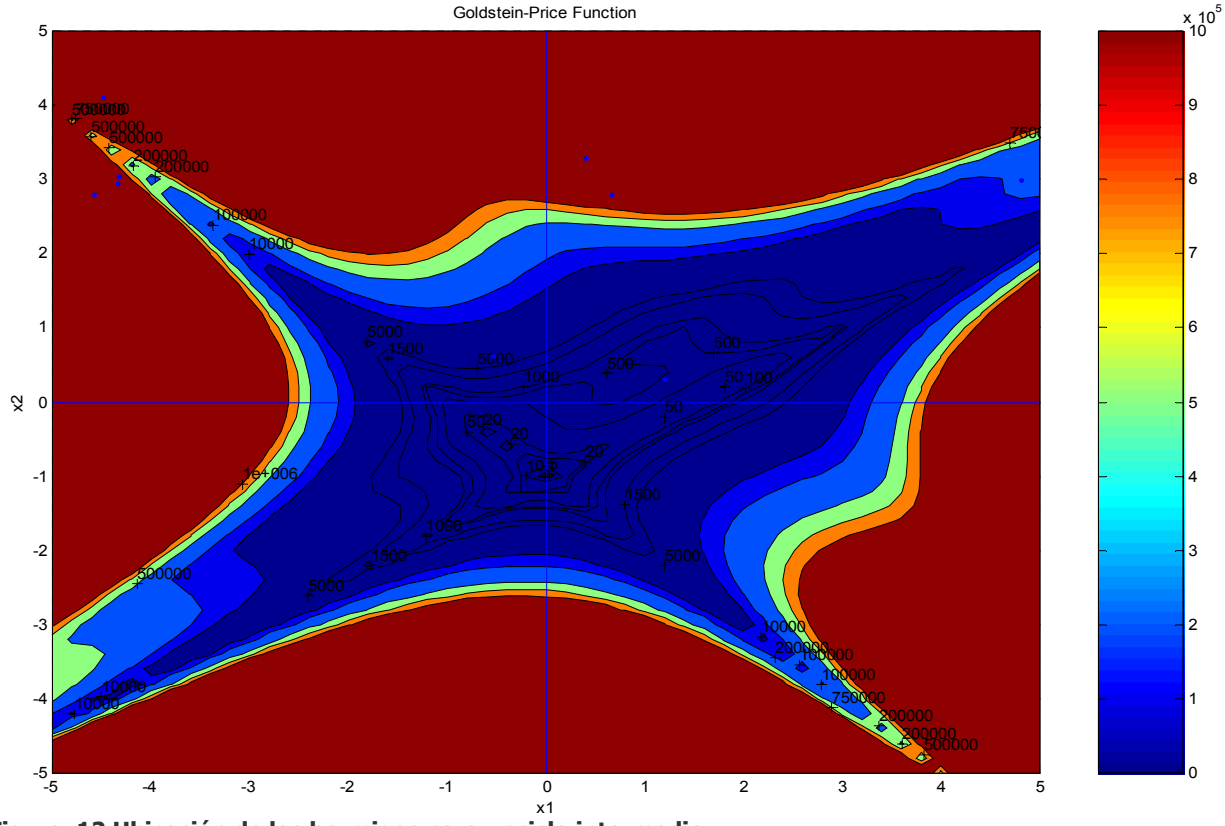


Figura 12 Ubicación de las hormigas para unciclo intermedio

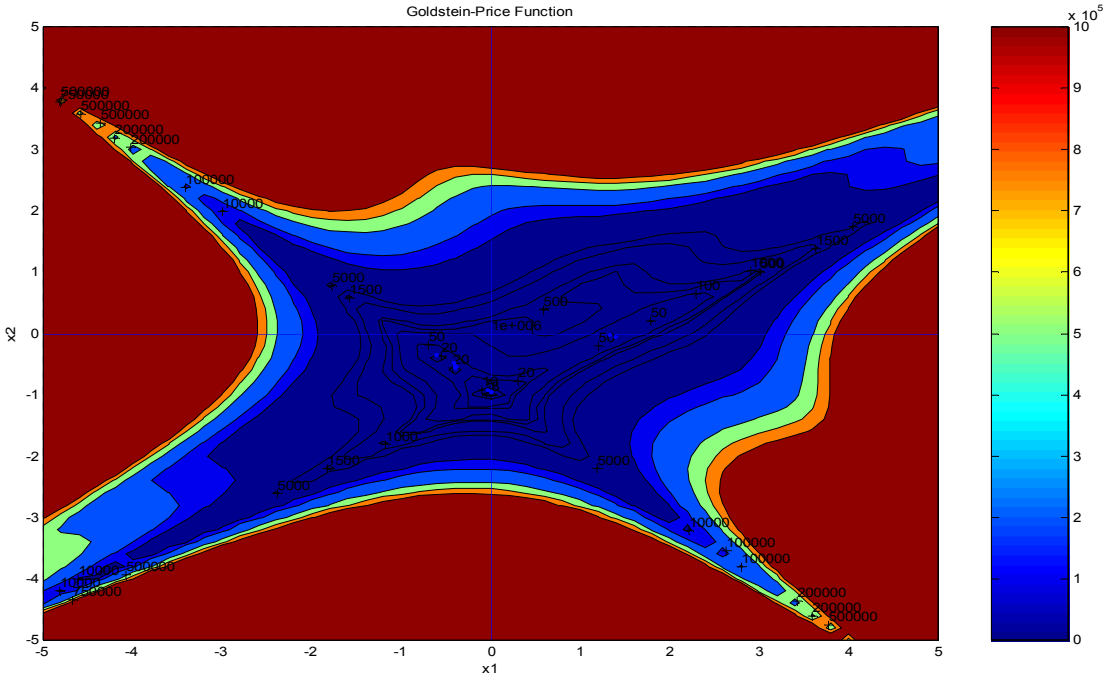
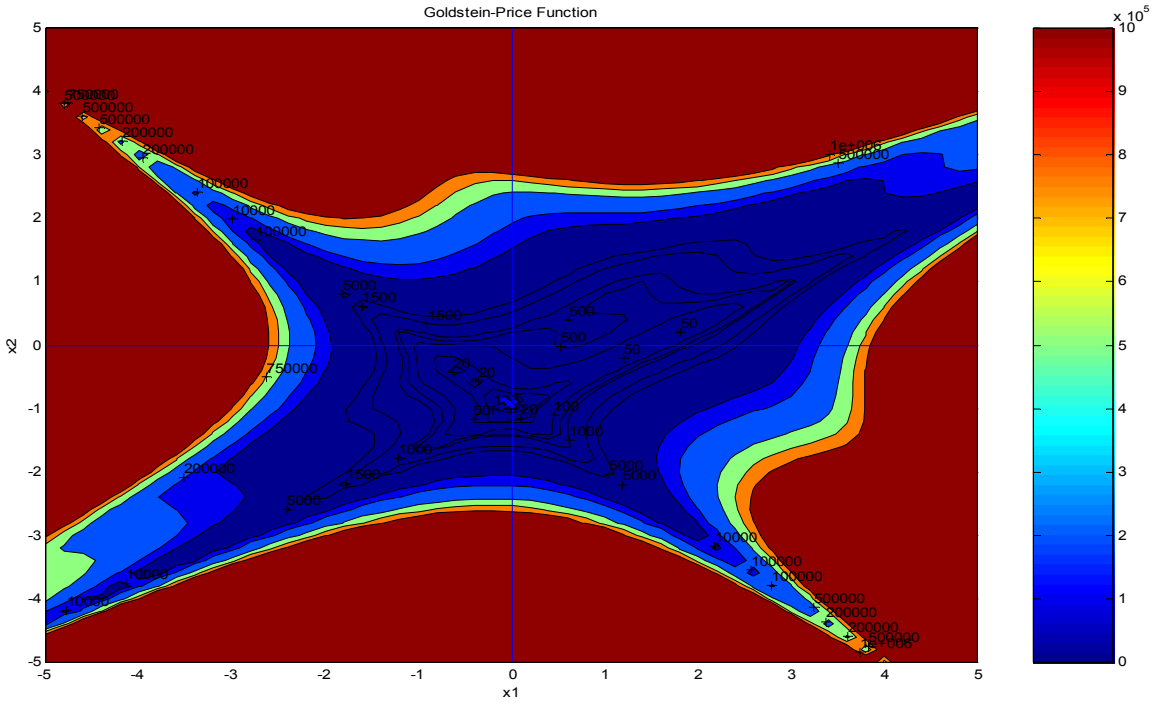


Figura 13 Ubicación de la hormigas en los últimos ciclos



c. Ubicar las hormigas aleatoriamente: ver literal a.

d. Crear mensajes: aunque esta parte del programa no aparece en el algoritmo, es importante crear los mensajes antes de comenzar los ciclos, esto para no hacer dos funciones diferentes para almacenar los mensajes.

Los mensajes son simplemente una matriz que contiene información acumulada de la aptitud y coordenadas de las hormigas a través de los ciclos. Solo almacena los diez mejores y se actualiza llamando la función ***menssage*** y se almacena en la matriz mensajes. **mensajes(3,nmens)**

Es importante tener mucho cuidado al diseñar el almacenamiento de los mensajes ya que son el 50% de la opción de destino, van a emular la persistencia de feromona en los mejores destinos y la evaporación de resultados intrascendentes.

Para el programa de Goldstein – Price la actualización de los mensajes se hace al terminar cada CICLO de iteraciones. La función ***message*** se lleva una matriz filtrada del mejor resultado que encontró cada hormiga (con sus coordenadas) y los mensajes actualmente almacenados, para ordenarlos y almacenar los diez mejores. Este filtro consiste en comparar el resultado que encontró la hormiga j y compararlo con el que obtuvo en el ciclo anterior, sólo almacenará su resultado si el que obtuvo es mejor que el anterior. Así se evita convergencia hacia los mensajes y se le da mayor acción a la estimergia.

e. Comienza el "***for***" del número de ciclos (CICLO) que se va a ejecutar el programa

f. Establecer rangos de movimiento: se estableció a partir del área total del espacio de búsqueda. Debe ser inversamente proporcional al número de hormigas(preferiblemente) por lo tanto se debe actualizar en cada ciclo. No es necesario hacerlo inmediatamente después de mover cada hormiga. Se determina con la función ***distancia*** y es almacenada en rmov: **rmov(1,m)**

La función ***distancia*** le asigna a cada hormiga un porcentaje máximo de exploración que disminuye a medida que va encontrando un mejor resultado; luego, lo multiplica por el área total del espacio de búsqueda y de esa área por hormiga calcula el radio máximo de desplazamiento.

g. En teoría comienza el ciclo por hormiga, pero como requiere información de la colonia se calcula primero información preliminar como el centro de gravedad para cada hormiga.

Para calcular los centros de gravedad, se crean puntos cercanos a cada hormiga(spots), a los que se les calcula y asigna un valor de feromona que depende del valor de la función en sus coordenadas, posteriormente se evalúa la siguiente fórmula

$$G_j = \sum_{i=1}^{n} \left(\frac{x_i \cdot \omega_{ij}}{\sum_{i=1}^{n} (\omega_{ij})} \right) \qquad \text{(F.5.1)}$$

G_j= Centro de gravedad para la hormiga $_j$
χ_i= coordenadas del punto en evaluación(spot)
ϖ_{ij}= Deseabilidad de la hormiga $_j$ por el punto $_i$
n= Número de puntos(nspo) (asociados a la hormiga $_j$)

Con

$$\omega_{ij} = \frac{\overline{\delta}}{2} \cdot e^{(-\theta_i \cdot \delta_{ij})} \qquad \text{(F.5.2)}$$

$\overline{\delta}$ = Distancia media entre individuos
θ_i= Valor de feromona asociado al punto $_i$
δ_{ij}= Distancia entre el la hormiga $_j$ y el punto $_i$

Figura 14. Calculo de centro de gravedad

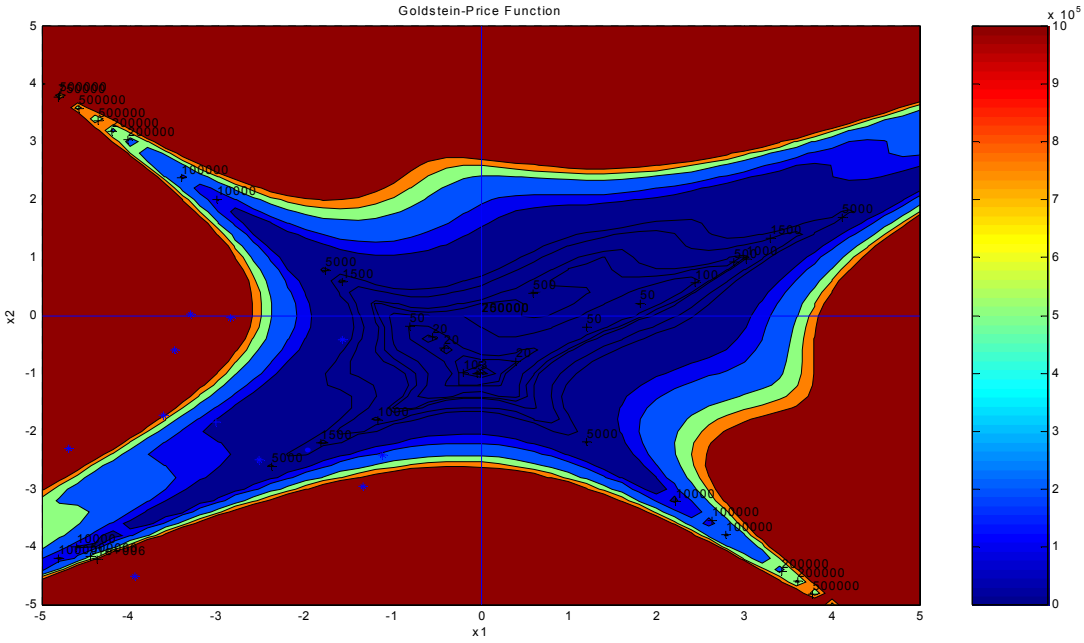
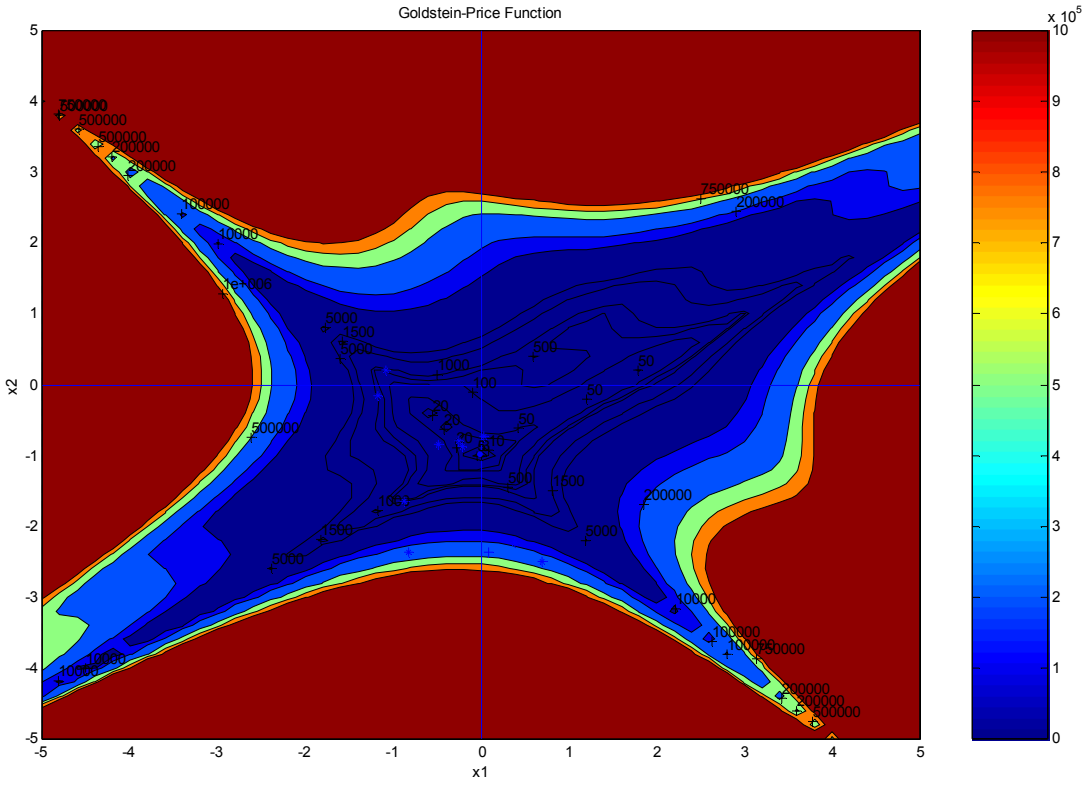


Figura 15 Variación de los spots para un ciclo al final de la corrida



Las Figuras 14 y 15 muestran las coordenadas de los diez spots(asteriscos) donde se evalúa la función para calcular el centro de gravedad(cruz) de la hormiga j(punto) en los primeros y los últimos ciclos respectivamente. Recuérdese que este centro de gravedad no se calcula en el plano sino con los valores de la función evaluada aplicando la ecuación (F.5.1).

h. Una vez calculados los centros de gravedad de las ***m*** hormigas, se puede comenzar con el ciclo para mover la hormiga *for j=1:m.*

Otra diferencia con el algoritmo original es que la hormiga no va a hacer los dos movimientos (estimergia y mensaje), comparará desde el comienzo cual le ofrece mejor aptitud y se moverá hacia ese.

Evaluación del canal de estimergia

i. Ubicar el centro de gravedad de la hormiga *j*, evaluar su valor de función y aptitud (**aptitud(goldsteinprice**(Gj1,Gj2))

j. Lectura de mensajes:

- Se genera un número aleatorio que es el índice del vector de mensajes.
- Se lee la información del mensaje almacenada en dicha posición, se guarda en nm(3,1)
- Comparar la aptitud del mensaje con la del centro de gravedad. El desplazamiento finalmente se hará hacia el valor de mayor aptitud.
- Calcular la distancia de la hormiga *j* al destino. Evaluada a partir de las coordenadas cartesianas, pero almacenada en coordenadas polares en **dire(2,1)**

k. Movimiento hacia el destino: para este punto se debe tener en cuenta que se trata de una función continua, así que una forma de discretizarla es dividiendo el radio de avance en fracciones según el grado de precisión deseado, y evaluar la función a cada incremento del delta de radio. Si se encuentra una mejor aptitud se almacena con sus coordenadas en el vector **bevap(3,1)**

Al terminar el avance el mejor resultado encontrado por la hormiga *j* se almacenará en el vector **bsol(3,1)**
Y en la matriz **tbevap(:,j)**
que al terminar el ciclo **CICLO** tendrá almacenado el mejor resultado de la **m** hormigas.

i. Ahora se debe actualizar el vector de mensajes pero de manera óptima, una forma es como se dijo anteriormente evitar almacenar resultados repetidos o inferiores generados por la misma hormiga, para lo cual se compara su mejor resultado **tbevap(1,j)>apt(1,j,CICLO)** con el resultado obtenido por la misma hormiga en el ciclo anterior[♦]; si el resultado es menor al anterior, se almacenará un valor de -1 en la nueva matriz, de lo contrario el valor obtenido.

tbevap2(3,j)

j. Fin del ciclo ***j.***

k. luego se llama la función **mensaje** que concatena, ordena y almacena los diez mejores mensajes existentes entre **menjases** y **tbevap2**

l. Almacenamiento en la matriz de consolidados: finalmente se almacena en **apt** los resultados del ciclo, es decir los mejores resultados obtenidos por las **m** hormigas.

m. Fin del ciclo **CICLO.**

5.1 ALGORITMO "CACO" PARA MODELO DE THOMAS

El algoritmo Hormiga desarrollado para calibrar cuencas usando el modelo de Thomas parte del mismo algoritmo utilizado para calibrar la función Goldstein-Price, es decir la función principal Thomfant con llamado de las funciones:

[♦] Obsérvese que se está comparando con el valor de **apt** en **CICLO** y no en **CICLO-1**, esto es porque los valores de arranque se almacenaron como ciclo uno, por lo tanto el resultado de cada ciclo(que se hace al final), se está almacenando en **CICLO+1.**

hormiguero
message
distancia
aptitud
thomas

La secuencia de trabajo del programa es como se muestra a continuación:

Entrada de datos del problema
Vector de precipitaciones
Vector de evapotranspiraciones potenciales
Vector de caudales observados (m^3/s)
Área de la cuenca
Para cálculos internos, se crea una matriz con los valores de cada variable *MVval*(valor mínimo, valor máximo, precisión, número de posibles valores)

Tabla 12 Almacenamiento en Matriz con los punteros deequivalencia(azul)

Variable/puntero	1	2	3	…	200	…	340	…	500	…	899	…	999
A	0,8	0,801	0,802	…	1	…							
b	10	11	12	…	210	…	350						
c	0,001	0,002	0,003	…	0,201	…	0,341	…	0,501	…	0,9	…	
d	0,001	0,002	0,003	…	0,201	…	0,341	…	0,501	…	0,9	…	1
Sw	0	1	2	…	200	…	340	…	500				
Sg	0	1	2	…	200	…	340	…	500				

Posición inicial de cada hormiga (Matriz pos y matriz punter)[20]
En la matriz aptitud se almacena la aptitud y los punteros de posición de cada hormiga a través de los ciclos.
Mensajes: se llama la función mensajes por primera vez, la cual ordena y almacena los *nmens* mejores mensajes según su valor de aptitud.
Comienza el ciclo por iteraciones.
Llamado de la función distancia: Se encarga de optimizar el espacio de búsqueda a medida que se va obteniendo un mejor resultado de aptitud. Es una matriz que almacena el porcentaje de avance de cada variable para cada hormiga.

APLICACIÓN CANAL DE ESTIMERGIA*

[20] Para facilitar la programación del algoritmo y el cálculo de operaciones, se creó una matriz paralela a la matriz pos (me guarda los valores de cada variable para cada hormiga (6,m), la cual va a almacenar el número (puntero) del matriz de valores posibles de cada variable que contiene el valor almacenado en pos.
* Para Thomas ver programa desde la línea 352.

Se ubican los spots en los que se evalúa la función para determinar el centro de gravedad, estos spots estarán cad vez más cerca de medida que la hormiga vaya mejorando su función objetivo gracias a la función distancia que a su vez depende de la aptitud actual de la hormiga. Para el cálculo de los centros de gravedad se resuelven las ecuaciones (F.5.1 y F.5.2).

MOVIMIENTO DE HORMIGAS
Leer mensaje aleatoriamente.
Comparar aptitud del mensaje con aptitud del centro de gravedad.
Moverse hacia el mejor resultado. Este movimiento se hace a través del cálculo de una hiper-diagonal de seis dimensiones con dominio en los parámetros de Thomas(a,b,c,d,S_{w0},S_{g0}) expresados como punteros. Posteriormente se calculan los cosenos directores del espacio para realizar el movimiento en cada dimensión como fracción de la diagonal.
A fin de evaluar todos los posibles valores desde la posición de la hormiga hasta su destino *(i-1,i)*, se determina la variable con mayor número de valores posibles a tomar, así se garantiza una menor variación en las demás variables y se evalúan todos los posibles valores de la diagonal.
Almacenar mejor resultado a lo largo del avance. Se almacena aptitud y el valor de variable (*bsol*) y los punteros en (*tbevap).*
Filtro y almacenamiento de mensajes. En un vector temporal se almacenan los mejores resultados de las hormigas si fueron mejor que su aptitud del ciclo anterior, esto para no saturar el vector de menajes con tendencia a un óptimo local.
Fin de los ciclos.

Tabla 13 Principales Diferencias de los programas

Parámetro	Goldstein-Price	Thomas
Porcentaje Max. de Exploración	0,35	0,41
Exploracion	Directa de la variable	Por vector paralelo de punteros
Movimiento de Hormigas	Diferencia de coordenadas con avance en coord. Polares	Difrerencia de coordenadas con avance por cosenos
Almacenamiento de Mensajes	Entran todos los valores a comparación	Se filtra de acuerdo a la aptitud del ciclo anterior
Valor máximo de feromona (Q)	50	0,01
Tiempos de ejecución promedio (s)	12	4800

5.2 VALIDACIÓN DEL PROGRAMA Y RESULTADOS

5.2.1 Descripción del caso de prueba

Como primer caso a usar en la validación del aplicativo de colonia de hormigas para calibrar cuencas hidrográficas a través del modelo de Thomas, se tomó un ejemplo del libro Hydrology for Engineers, Geologist, and Enviromental Profesionals" de Sergio Serrano. Resolviéndolo de forma inversa así:

Tabla 14 Información conocida del ejemplo

MES	P(i) (mm)	PE(i) (mm)	Qi (mm)
ENERO	55	32	186,56
FEBRERO	72	45	69,56
MARZO	101	60	92,79
ABRIL	110	85	99,13
MAYO	220	122	197,59
JUNIO	135	146	120,49
JULIO	43	166	37,34
AGOSTO	25	201	21,54
SEPT.	78	154	66,07
OCT.	97	101	82,54
NOV.	140	55	121,12
DIC.	99	43	84,96

Como se observa en la tabla 12, del ejemplo anterior se conoce la precipitación en milímetros, la Evapotranspiración potencial, y el caudal final para cada uno de los meses del año. Como el ejemplo original pretendía calcular el caudal a partir de valores conocidos de calibración de la cuenca por el modelo de Thomas (a, b, c, d, S_{w0}, S_{g0}), entonces se puede decir que se conocen los valores esperados, por lo tanto se sabrá si se consiguieron los parámetros de calibración ó por lo menos que tan cerca se estuvo.

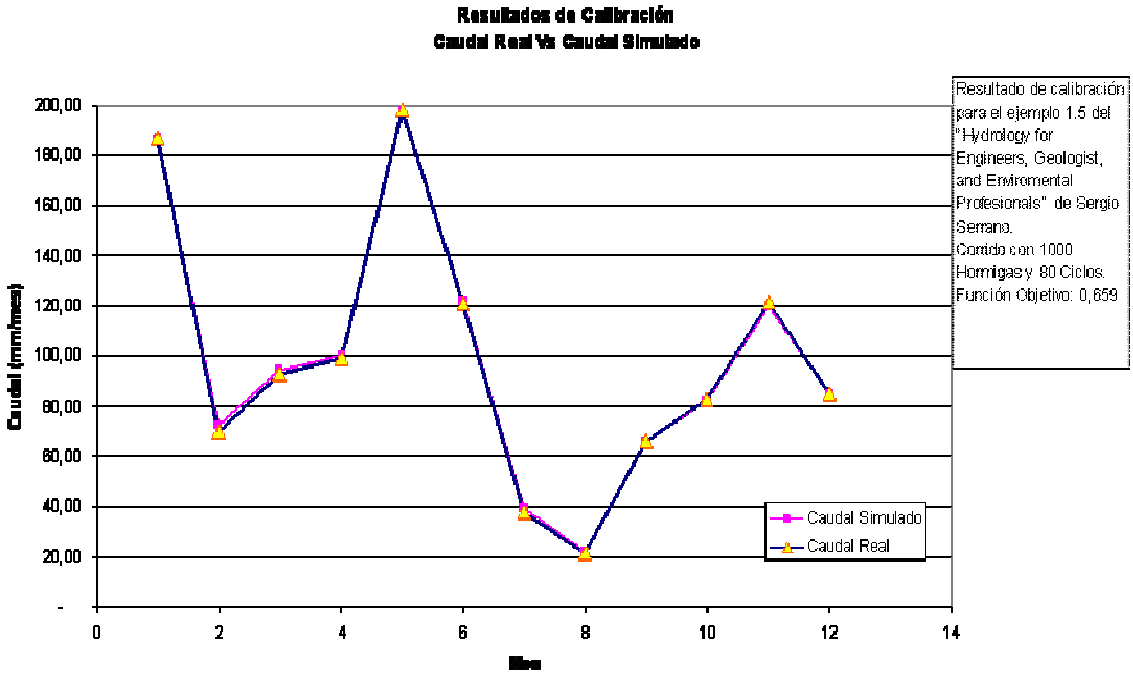
Tabla 15 Valores de la solución óptima

a	b	c	d	Sw0	Sg0
0,9	20	0,1	0,1	150	200

Tabla 16 Cálculo y resultados, Caso 1

MES	ENE.	FEB.	MAR.	ABR.	MAY.	JUN.	JUL.	AGO.	SEPT.	OCT.	NOV.	DIC.
P(i) (mm)	66	72	101	110	223	135	43	25	78	97	140	99
PE(i) (mm)	32	45	60	86	122	146	166	201	164	101	66	43
T A B L A D E C A L C U L O												
Sw	3,021	1,446	0,632	0,168	0,020	0,005	0,002	0,000	0,000	0,005	0,338	1,626
Sg	131,720	116,975	107,392	99,947	104,827	100,332	97,285	74,575	69,940	80,257	68,327	65,975
W(mm)	202,000	75,021	102,445	110,832	223,158	135,223	43,005	25,002	78,800	97,003	140,305	99,839
Y(mm)	17,813	17,836	17,729	17,752	17,885	17,303	17,107	16,940	17,824	17,711	17,311	17,721
Rb (mm)	162,031	50,525	74,250	81,736	173,001	103,152	22,737	7,997	53,131	59,777	107,564	72,264
Rg (mm)	22,095	6,930	10,166	11,146	24,273	14,065	3,101	1,086	7,246	9,515	14,571	9,854
Qg (mm)	24,369	21,640	19,549	18,490	19,393	18,561	16,148	13,795	12,774	12,265	12,548	12,206
Q (mm)	188,40	72,17	94,40	100,22	197,39	121,71	38,89	21,76	65,90	82,04	120,22	84,47
Qreal (mm)	188,56	69,56	92,79	99,13	197,59	120,49	37,34	21,54	66,07	82,54	121,12	84,36
Q - Qi (mm)	0,02	6,77	2,59	1,21	0,04	1,49	2,38	0,05	0,03	0,25	0,81	0,24
a	b(mm)	c	d	Sw0(mm)	Sg0(mm)							
0,527	18	0,12	0,185	147,00	134,00							

Figura 16 Resultado Caudales observados Vs. Caudales simulados Ejemplo Serrano



Obsérvese en la tabla de datos de la **figura 16** los parámetros de ejecución del programa*, en la **tabla 16** se puede ver el caudal observado, caudal simulado, la diferencia de los dos y los parámetros finalmente obtenidos. Estos datos permiten comprobar la eficacia del programa.

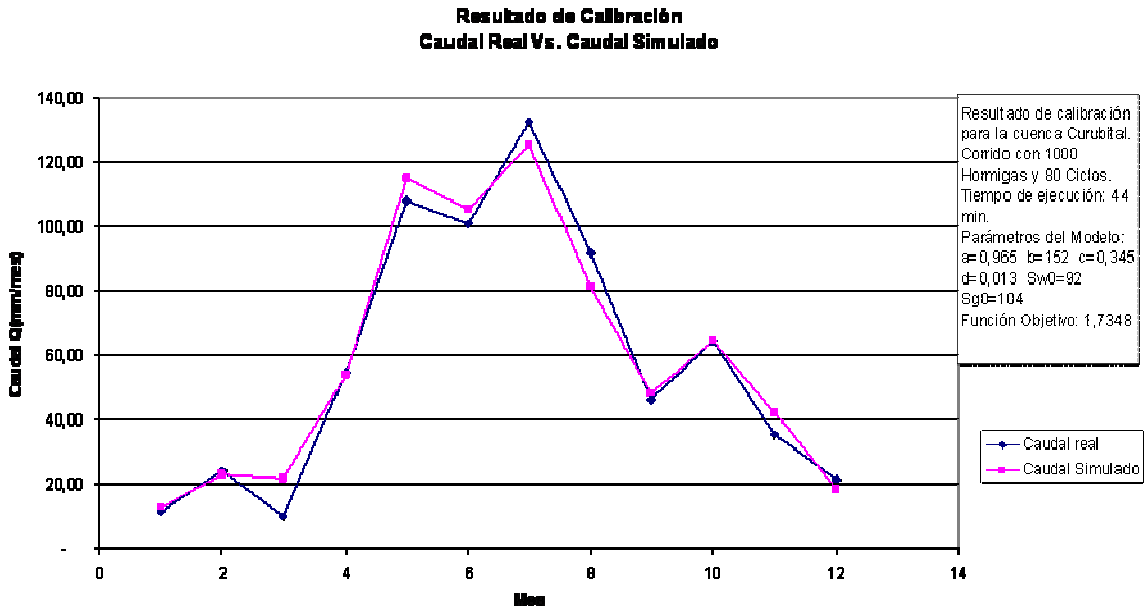
* Recuérdese que la función objetivo es la minimización de diferencia de caudales(Norma Cuadrática Normalizada (F.4.12))

El segundo caso requiere calibrar el modelo de Thomas aplicados a dos cuencas: una ubicada en la parte alta del río Subachoque y otra ubicada en la parte alta del río Tunjuelo cuenca de Curibital). Dichas cuencas se encuentran en la sabana de Bogotá.

Figura 17 Cálculo y resultados, Caso 2 Curubital

MES	ENE.	FEB.	MAR.	ABR.	MAY.	JUN.	JUL.	AGO.	SEPT.	OCT.	NOV.	DIC.
P(i) (mm)	47,78	86,98	76,08	139,15	229,01	210,03	243,56	173,34	121,47	146,5	105,42	52,29
PE(i) (mm)	66,43	65,02	70,87	70,46	71,06	76,09	77,01	79,33	71,57	69,77	63,62	67,88
	T A B L A D E C A L C U L O											
Sw	78,949	86,781	82,887	89,925	92,452	89,182	89,088	86,423	88,336	91,000	92,045	79,690
Sg	108,645	118,429	127,378	152,676	209,085	259,313	318,950	354,631	372,706	398,856	412,992	414,429
W (mm)	139,780	165,929	162,861	222,037	318,935	302,482	332,742	262,428	207,893	234,836	196,420	144,335
Y (mm)	122,222	133,107	132,123	142,955	147,554	147,124	147,861	145,643	141,458	144,009	139,886	124,553
Ro (mm)	11,500	21,498	20,133	51,799	112,255	101,760	121,097	76,494	43,515	59,492	37,030	12,957
Rg (mm)	6,057	11,324	10,605	27,283	59,127	53,599	63,784	40,291	22,920	31,335	19,504	6,825
Qg (mm)	1,412	1,540	1,656	1,985	2,718	3,371	4,146	4,610	4,845	5,185	5,369	5,388
Qi (mm)	12,91	23,04	21,79	53,78	114,97	105,13	125,24	81,10	48,36	64,68	42,40	18,34
Q real (mm)	11,86	24,22	10,25	54,83	107,86	100,83	132,59	91,69	46,23	64,39	35,53	21,46
Qr- Qi (mm)	-1,05	1,18	-11,54	1,05	-7,11	-4,30	7,35	10,59	-2,13	-0,29	-6,87	3,12
a	b(mm)	c	d	Sw0(mm)	Sg0(mm)							
0,965	152	0,345	0,013	92,00	104,00							

Figura 18 Resultado Caudales observados Vs. Caudales simulados Cuenca Curubital



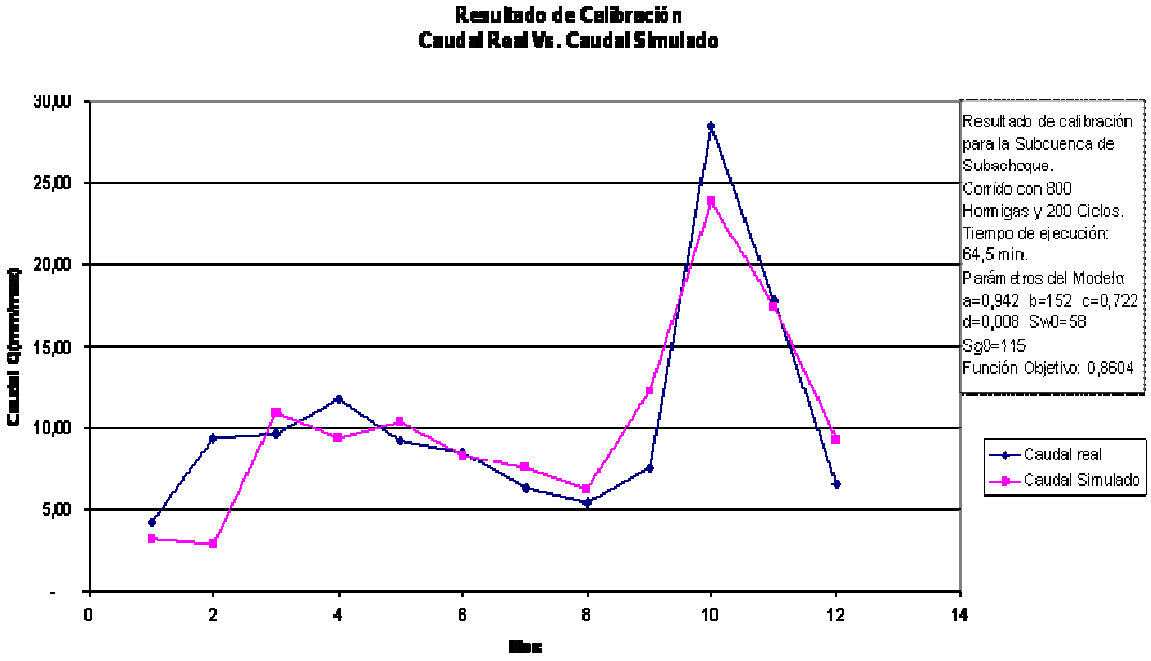
Los parámetros obtenidos arrojan valores finales semejantes a los medidos.

Figura 19 Cálculo y resultados, Caso 3 Subachoque

MES	ENE.	FEB.	MAR.	ABR.	MAY.	JUN.	JUL.	AGO.	SEPT.	OCT.	NOV.	DIC.
P(i) (mm)	40,62	27,17	92,96	53,4	70,94	51,8	42,85	39,29	77,56	114,7	89,93	48,55
PF(i) (mm)	49,4	35,94	38,87	55,03	47,59	36,45	45,7	35,06	35,92	55,21	56,73	46,54
TABLA DE CALCULO												
Sw	85,61	88,07	97,47	84,97	90,88	93,11	85,23	88,30	100,13	95,78	91,98	88,45
Sg	119,86	123,43	147,84	107,11	188,03	204,42	217,72	227,33	252,00	305,28	340,82	364,07
W (mm)	98,62	92,78	161,03	150,87	155,91	142,48	135,96	124,52	163,86	214,83	185,71	140,53
V (mm)	90,90	96,22	125,87	122,04	124,02	118,35	115,12	108,80	128,92	137,73	132,72	117,41
Ro (mm)	2,17	1,82	9,77	8,01	8,86	6,71	5,79	4,40	10,30	21,43	14,73	6,43
Rg (mm)	5,64	4,73	25,39	20,81	23,02	17,43	15,05	11,42	26,75	55,66	38,26	16,69
Qg (mm)	0,96	0,99	1,18	1,34	1,51	1,64	1,74	1,82	2,02	2,44	2,73	2,84
Qi (mm)	3,13	2,81	10,96	9,35	10,37	8,35	7,51	6,22	12,31	23,88	17,16	9,26
Q real (mm)	4,22	9,35	9,64	11,76	9,25	8,52	6,31	5,42	7,49	28,46	17,79	6,57
Qr- Qi (mm)	1,09	6,54	-1,32	2,41	-1,12	0,17	-1,23	-0,90	-4,82	4,58	0,33	-2,69

a	b(mm)	c	d	Sw0(mm)	Sg0(mm)
0,942	152	0,722	0,008	58,00	115,00

Figura 20 Resultado Caudales observados Vs. Caudales simulados Cuenca Subachoque



Aunque en los resultados obtenidos se puede notar una diferencia marcada en algunos puntos, la función objetivo no es tan alta, debe tenerse en cuenta que los rangos de los caudales son pequeños, por lo tanto para la función objetivo la varianza es significativa, pero en realidad no lo es tanto.

Sin embargo es importante resaltar que estas dos últimas cuentas se trabajan con información de campo que puede tener algún margen de error que no permite mejorar los resultados.

5.2 OTROS RESULTADOS

RESULTADOS GOLDSTEIN PRICE

Figura 21 Variación de llamados en función de otros parámetros

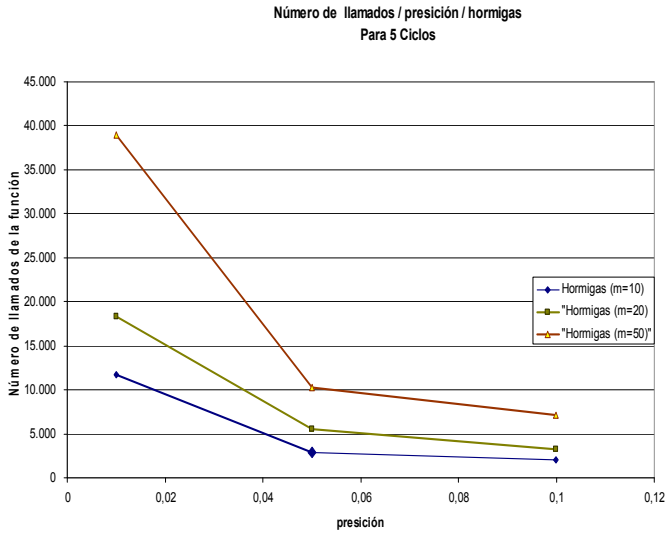
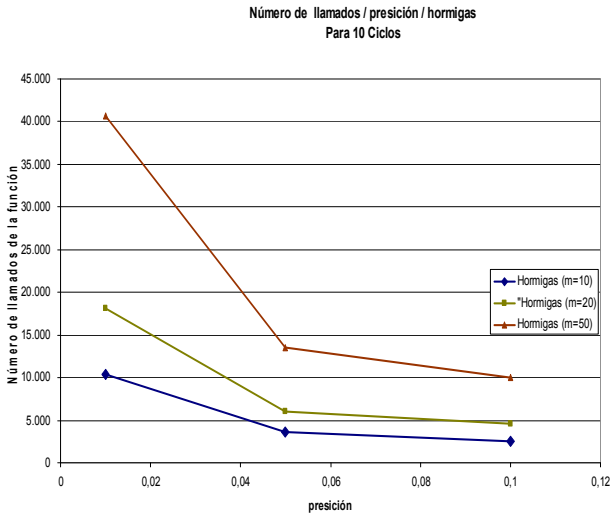


Figura 22 Figura 21 Variación de llamados en función de otros parámetros



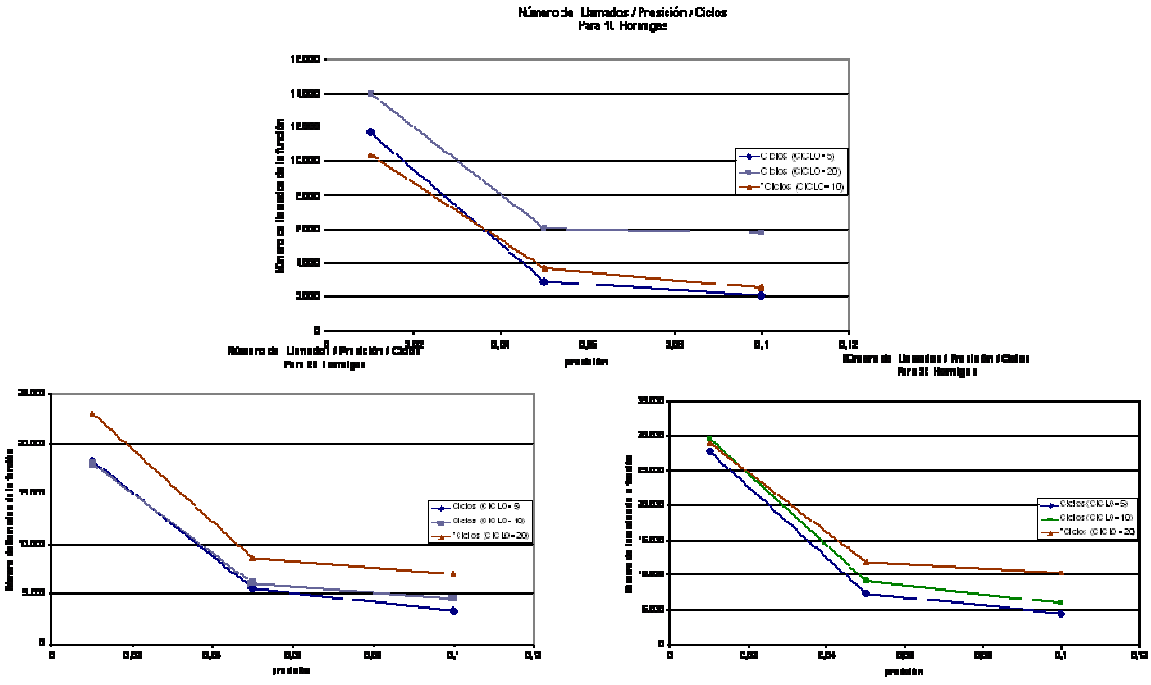


Figura 23. Evolución para 4 ciclos y 20 hormigas, tiempo de ejecución 2 s.

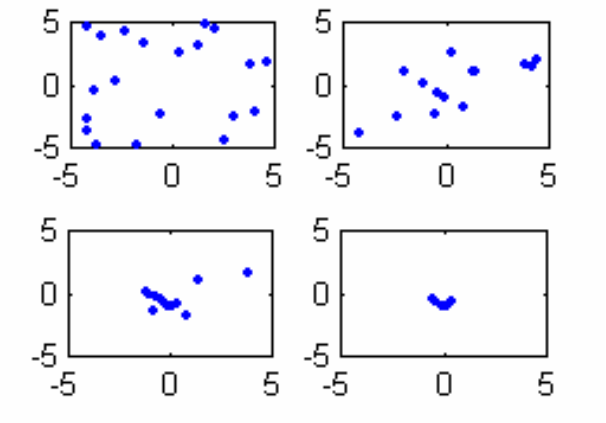


Figura 24. Evolución para 80 ciclos(grafico cada 4)
2 hormigas, tiempo ejecución 0.98s.

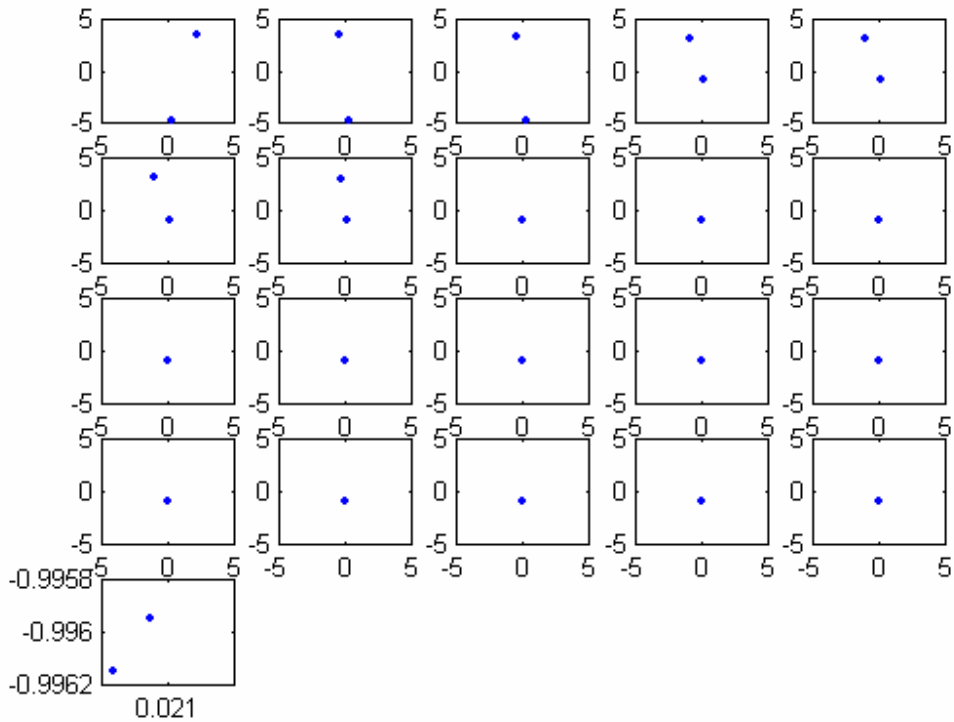


Figura 25. Evaluación para 5 ciclos y 18 hormigas, tiempo de ejecución 1.15s.

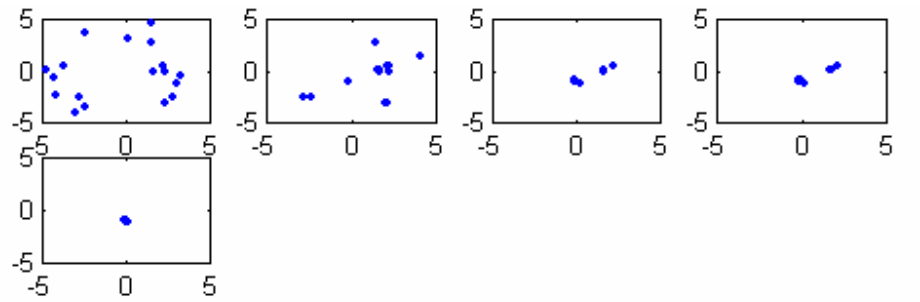
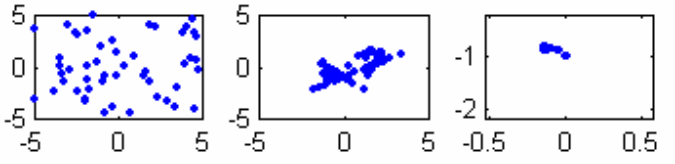


Figura 26. Evaluación para 3 ciclos y 50 hormigas, tiempo de ejecución 2.89s.



6. CONCLUCIONES

- La optimización de Colonia de Hormigas, demuestra ser una potente herramienta para aplicar el problemas de ingeniería.

- Aunque en principio parezca complicado, resulta realmente sencilla su implementación.

- Queda demostrada la versatilidad de aplicación del método de optimización con los tres programas implementados.

- Se deja el camino trazado para la implementación de nuevas aplicaciones, que como se demostró no se restringe a un solo campo de uso.

Del aplicativo al modelo de Thomas se puede concluir:

- Que los resultados obtenidos en la calibración del modelo dejan ver la eficacia del programa.

- Que el mayor incremento en el número de evaluaciones y por ende en el tiempo de ejecución se presenta cuando se aumenta el número de hormigas, mas que aumentando el número de ciclos, sin embargo también se mejoran los resultados al tener un mayor número de hormigas.

- Por lo anterior es importante hacer varias pruebas para determinar un número de ambos que optimice los resultados y el tiempo de ejecución.

- Dada la magnitud del campo de exploración y los relativamente cortos tiempos de ejecución, se puede aumentar el número de iteraciones a fin de mejorar los resultados.

7. RECOMENDACIONES

- Al asignar valores aleatorios, verificar no cruzar las condiciones de frontera.

- Determinar una adecuada función Objetivo que permita ser planteada en términos de minimización o maximización o que permita establecer criterios medibles de optimización.

- Reducir la desviación a medida que va mejorando la función objetivo.

- Almacenar un número de mensajes entre el 10% - 30% del número de hormigas.

- Posibles mejoras aplicables al programa de Thomas:

- Mejorar los criterios iniciales de las variables ya sea reduciendo los rangos de exploración por variables si se conocen características de la cuenca, o por medio de acciones de optimización adicionales como criterios puntuales para cada variable en función de su comportamiento. Se observó que algunos meses (los de mayor o menor precipitación y caudal final pueden ser asociados con determinadas variables facilitando predecir los resultados).

- Optimizar el almacenamiento de mensajes para estimular esta ruta de óptimización.

8. REFERENCIAS

[1] Dorigo M. Maniezzo V. and Colorni A. The ant system: Optimization by a colony of cooperating agents. Université libre de Bruxelles. 1996.

[2] Dorigo M. and Di Caro G. The Ant Colony Optimization Meta-Heuristic. New ideas in optimization. McGraw Hill. 1999.

[3] G. Brassard y P. Bratley. Fundamentals of Algorithmics. Prentice Hall, Englewood Cliffs, NJ, 1996.

[4] I. H. Osman and J. P. Kelly, editores. Meta-Heuristics: Theory & Applications. Kluwer Academic Publishers, Boston, MA, 1996.

[5] Z. Michalewicz y D. B. Fogel. How to Solve It: Modern Heuristics. Springer Verlag, Berlín, Alemania, 2000.

[6] E. Bonabeau, M. Dorigo, y G. Theraulaz. Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York, NY, 1999.

[7] Alonso S., Cordón O., Fernández de Viana I, Herrera F. La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques. Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S. Ingeniería Informática, C/ Periodista Daniel Saucedo Aranda s/n, 18071 Granada (España)

[8] M. Dorigo y L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1: 1, páginas 53-66, 1997.

[9] Colorni A., Dorigo M., Maniezzo V. Distributed Optimization by Ant Colonies. Dipartimento di Elettronica, Politecnico di Milano Piazza Leonardo da Vinci 32, 20133 Milano, Italy

[10] Dréo J., Siarry P. A New Ant Colony Algorithm Using the Heterarchical Concept Aimed at Optimization of Multiminima Continuous Functions. Université de Paris XII Val-de-Marne, Laboratoire d'Étude et de Recherche en Instrumentation Signaux et Systèmes (L.E.R.I.S.S.), 61 avenue du Général de Gaulle, 94010 Créteil, France.

[11] OLARTE VALDIVIESO. Rafael E. Herramientas para la implementación de algoritmos genéticos en ingeniería civil con énfasis en hidroinformática. Bogotá D.C. 2003.Trabajo de grado (Ingeniero Civil). Pontificia Universidad Javeriana. Facultad de Ingeniería Civil. Línea de Aguas.

[12] CARO CAMARGO, Carlos Andrés y FLECHAS PARRA, Franklin Paolo. Análisis del balance hídrico de la sabana de Bogotá mediante el método de Thomas. Bogotá, 2002, 139 p. Trabajo de Grado (Ingeniero Civil). Pontificia Universidad Javeriana. Facultad de Ingeniería Civil. Línea de Aguas.

[13] SERRANO, Sergio E. Hydrology for engineers, geologists, and environmental professionals. Lexington. HydroScience Inc.. 1997. Capítulo 1.

[14] ALLEY, William M. On the treatment of evapotranspiration, soil moisture accounting, and aquifer recharge in monthly water balance models. En: Water resources research. Vol. 20, No. 8 (Ago. 1984); pag. 1137-1149.

ANEXO A

CODIGO DE PROGRAMACIÓN PARA LOS ALGORITMOS DE GOLDSTEIN-PRICE Y THOMAS

Como se comentó anteriormente, el código del programa está programado en lenguaje Matlab versión 5.3. Está compuesto por el archivo principal ***thomfant***, y las funciones que se describen a continuación, cada línea del programa que lo requiera se encuentra claramente comentada:

aptitud: asigna el valor de feromona; diferenciando la función a optimizar (Price o Thomas).

direccion: solo aplica para la función Price. Indica el camino a tomar para el movimiento de las hormigas en términos de un ángulo en radianes y un radio de avance.

distancia: devuelve el porcentaje o fracción máxima de avance de cada variable; se optimiza con el valor de la función objetivo.

goldsteinprice: calcula el valor de la función Goldstein-Price para los parámetros de entrada.

grafique: solo se usa cuando se quieren graficar resultados de Price, dibuja las curvas de nivel.

hormiguero: Se encarga de almacenar en una matriz los movimientos de las hormigas con el valor de aptitud.

message: recibe los mensajes almacenados y los mejores resultados de las hormigas en el presente ciclo y los organiza para almacenar solo los ***nmens*** mejores mensajes.

thomas: evalúa las ecuaciones del Modelo de Thomas y la función objetivo. Devuelve este último valor.

Programa Principal thomfant

```
clc
disp('EL SIGUIENTE ALGORITMO PERMITE SOLUCIONAR LA FUNCION DE GOLDSTEIN
PRICE O ')
disp('CALIBRAR UNA CUENCA HIDROLOGICA CON EL MODELO DE THOMAS')
disp('DISEÑADO Y PROGRAMADO POR: CARLOS ALFREDO COY CALIXTO')
disp('PONTIFICIA UNIVERSIDAD JAVERIANA')
disp('FACULTAD DE INGENIERIA')
disp('DEPARTAMENTO DE CIVIL')
disp('2005')
disp('')
%-------------------------------------------------------------------------
--
%diary resulthomas4.txt
%diary resulprice.txt
% Las  siguientes líneas determinan cuantas veces se quiere repetir la
evaluación con los mismos parámetros
% Se usa para observar la persistencia del programa. para usarse en forma
semiautomática, se debe habilitar
% la asignación previa de valores a las variables de entrada, y
deshabilitar las entradas.
persis=1;
for verr=1:persis
%Variables de inicialización del algoritmo
clear
global CICLO PM;
 porbusq = .5;
CICLO=1;
%nspo es el número de puntos para calcular centros de gravedad, porsbuq
porcentaje de búsqueda
%obfunci=2; iter=5
%m=250
%obfunci=1; iter=10; m=10;

obfunci=input('Que función quiere resolver ( 1 / 2 )?: 1 Goldstein-Price
2 Thomas  :_');
 %---------------------SELECCIONO LA FUNCION GOLDSTEIN - PRICE O THOMAS--
-----------------------------------------------------
if obfunci==1
%------------------SE SELECCIONO GOLDSTEIN PRICE -----------------------
---------------------------------
iter=input('Introduzca el Número de iteraciones deseado? (6 óptimo)
:_');
m=input('Introduzca el Número de hormigas deseado? (20 óptimo)  :_');
fpa=input('Grado de precisión de busqueda deseado? (0.01 óptimo)  :_');
tic;                  %comando activa reloj para contar tiempos de
iteración
nspo = 10;                  %definición del número de spot para cálculo
del centro de gravedad
```

```
Q=50;                       % Valor max de feromona para determinar la
aptitud
nmens = 10;                 %nmens es el número de mensajes
mensajes=-ones(3,nmens);    %Matriz de 3 por nmens(10) donde se guardaran
los nmens mejores mensajes, sus filas contienen
                            %F1= la aptitud como feromona, F2= coordenada
x1, F3= coordenada x2
bsol=ones(3,1);             %Vector columna de 3 por 1 que guarda la mejor
solución
temp = 10*rand(2,m)-5;      %Matriz 2 por m (numero de hormigas, Ubicación
aleatoria de las hormigas x1; x2
pos(:,:,1) = temp;          %Matriz multidimensional de 2 por m. Guarda la
posicion de las hormigas en el ciclo presente para
calculostbevap(1:2,1:m)=10;         %Matriz (2,m) que almacena la mejor
solución de cada hormiga en el ciclo terminado
MGSPO(1:nspo,1:2,1:m,1:CICLO)=10;% Matriz de coordenadas de los spots de
todas las hormigas en todos los ciclos(para graficar)
format long;
%-------------------------------------------------------------------------

%              COMIENZO DE ITERACIÓN

%-------------------------------------------------------------------------
%Matriz de 3 por m. apt es la aptitud y coordenadas de cada hormiga por
iteración (feromona, x1, x2)
apt = hormiguero(Q, m, pos,obfunci);
%-------------------------------------------------------------------------
Aterreno(:,:) = [-5 5; -5 5];       %Parámetros que definen al terreno y
calculo de area del terreno
Area = (abs(Aterreno(1,2)-Aterreno(1,1)))*(abs(Aterreno(2,2)-
Aterreno(2,1)));

%-------------------------------------------------------------------------
%Matriz 3 por nmens. guarda los Mensajes
%Contiene el valor de aptitud y las coordenadas x1 y x2 en el domino
%de la función, los valores salen de los resultados de las mejores
(nmens) hormigas en el acumulado de las iteraciones.
mensajes = message(apt(:,:,CICLO),nmens,m, mensajes);
%-------------------------------------------------------------------------

for CICLO=1:iter

%-------------------------------------------------------------------------
--
%Función: (1,m).  Calcula de la distancia de movimiento permitida para
cada hormiga (avance máximo)
rmov = distancia(Area,apt(:,:,CICLO),m,1);
%-------------------------------------------------------------------------

%EL PROGRAMA TRABAJA CON DOS CANALES DE COMUNICACION
% ------------------------CANAL DE ESTIMERGIA----------------------------
```

```
%Cálculo del centro de Gravedad
% para calcular el centro de gravedad debo calcular algunos elementos
 cont=0; rdis=0; acum=0; vgj=zeros(3,1);  %vgj Matriz de centros de
gravedad para las hormigas
 mspot=zeros(nspo,3,m);       %matriz que guarda las coordenadas de los
spot(10) de cada ant(30)
   %Calculo De La Distancia Promedio Entre Individuos para j, m hormigas
    for j=1:m
       for i=1:m-1
            if i~=j
               rdis=sqrt((pos(1,j,1)-pos(1,i,1))^2+(pos(2,j,1)-
pos(2,i,1))^2);
            acum=rdis+acum;
            cont=cont+1;
            end
       end

    end
    dprom=acum/cont;       % el ciclo anterior calcula la distancia
promedio entre 2 agentes (delta medio)
 for j=1:m

 % CREACION Y UBICACION DE SPOTS
%Los spots se utilizan para calcular el centro de gravedad
%ver ecuación de Gj en documento
wijacum=0;
      for n=1:nspo
         xtemp=0; ytemp=0;             %variables temporales de cálculo
%------------- PARA EVITAR QUE SE PASE DE LOS BORDES DEL RANGO, SOBRETODO
QUE SE SALGA ---
[xtemp,ytemp]=pol2cart(rand*2*pi,rand*rmov(1,j));
%coordenadas aleatorias a partir del radio de movimiento de cada ant y en
cualquier dirección
         provix=xtemp+pos(1,j,1);
         proviy=ytemp+pos(2,j,1);   %a las coordenadas calculadas le suma
las actuales de la hormiga j
         if provix<-5
            mspot(n,1,j)=-5;
         elseif provix>5
            mspot(n,1,j)=5;
         else
            mspot(n,1,j)=provix;       %Almacenamiento de coordenadas de
puntos, verificando no salir de los limites
         end                           % de rango
         if proviy<-5
            mspot(n,2,j)=-5;
         elseif proviy>5
            mspot(n,2,j)=5;
         else
            mspot(n,2,j)=proviy;
         end
         mspot(n,3,j)=goldsteinprice(mspot(n,1,j),mspot(n,2,j));
```

```
         % mspot matriz multidimensional de cada fila es un spot, las
colunas son x1, x2, valor de la función
         % para cada hormiga (n,3,j)
         mtemp=mspot(:,:,j);           %Extracción matriz bidimensional
para cálculo
         mspot(n,4,j)=aptitud(Q,mspot(n,3,j),1);      % almacenamiento de
feromona s/n valor de función
         rdisp(j,n)=sqrt((pos(1,j,1)-mspot(n,1,j))^2+(pos(2,j,1)-
mspot(n,2,j))^2);
         % Cálculo de distancia entre el spot i y la hormiga  j.
         wij(n)=(dprom/2)*exp(-mspot(n,4,j)*rdisp(j,n)); %calculo de wij
         wijacum=wij(n)+wijacum;  %Sumatoria de wij
      end
      Gj=zeros(2,1);
      for n=1:nspo                  %Cálculo del centro de gravedad Gj
para cada hormiga
         Gj=Gj+[mspot(n,1,j); mspot(n,2,j)]*(wij(n)/wijacum);
      end
      vgj(1,j)=Gj(1,1);    % almacenamiento en un vector de las
coordenadas de los centros de gravedad
      vgj(2,j)=Gj(2,1);    % Guarda x1, x2, aptitud(feromona) como filas,
para m hormigas (3,m)
      vgj(3,j)=aptitud(Q,goldsteinprice(Gj(1,1),Gj(2,1)),1);
 end        %end del for j line73
 MGSPO(:,:,:,CICLO)=mspot(:,1:2,:); %Almacenamiento de spot para
graficarlos
 Mvgj(:,:,CICLO)=vgj;               %Almacenamiento de centros de
gravedad para graficarlos
 %-------------------------------------------------------------------------

%-------MOVIMIENTO ANTS-------------
for j=1:m
   bevap=apt(:,j,CICLO); % (3,m) Almacenamiento temporal de la mejor
aptitud por hormiga(valor inicializado en el punto de arranque

   %----------------LECTURA DE MENSAJE---------------
     PM=ceil(10*rand); % (1,1)Selección aleatoria de una posición para el
vector mensajes
     nm(:,:)=mensajes(:,PM); %(3,1) Obtención de la información del
mensaje selecionado

     %--------------------------COMPARO SI ME MUEVO HACIA EL MENSAJE O
HACIA EL CENTRO DE GRAVEDAD----------
     if      nm(1,1)>vgj(3,j)       %Compara Aptitudes del mensaje y del
centro de gravedad  para la hormiga j y elige la mayor
          %En las siguientes lineas se Calcula dirección y avance para
moverse hacia el mensaje, son coordenadas polares
          % dadas como coordenadas locales desde la Ubicación de la
hormiga (angulo y radio)
         antj=pos(:,j);
         mnact(1,1)=nm(2,1);    mnact(2,1)=nm(3,1);
         local(:,:)=mnact-antj;
```

```
        [dire(1,1),dire(2,1)]=cart2pol(local(1,1),local(2,1));
      else
         % Aquí Calcula dirección y avance para moverse hacia el Centro
de Gravedad, son coordenadas polares dadas como
         % coordenadas locales desde la Ubicación de la hormiga, son
(angulo y radio)
        antj=pos(:,j);
        gjact(1,1)=vgj(1,j);
          gjact(2,1)=vgj(2,j);
        local(:,:)=gjact-antj;
        [dire(1,1),dire(2,1)]=cart2pol(local(1,1),local(2,1));
    end
    Vdesp=0:fpa:dire(2,1);      % Divido el radio de avance en el grado
de precisión que da el usuario
    for  p=1:length(Vdesp)
        [dx1,dx2]=pol2cart(dire(1,1),Vdesp(p)); % Cálculo de coordenadas
de avance a partir del origen de la hormiga
        x1=pos(1,j)+dx1;      x2=pos(2,j)+dx2;  % Se le suman las
coordenadas de la hormiga
        tevap=aptitud(Q,goldsteinprice(x1,x2),1); % Evaluación de la
función y la aptitud(feromona)
        if tevap>bevap(1,1)
           bevap=[tevap;x1;x2];             %almacenamiento del mejor
resultado por hormiga en el ciclo
        end
    end
      if bsol(1,1)<bevap(1,1)
        bsol=bevap;                         % Almacenamiento del mejor
resultado global
      %   display([CICLO,j,bsol(2),bsol(3),bsol(1,1)]);
      end
     pos(1,j)=x1; pos(2,j)=x2 ;         %Almacenamiento de las
coordenadas finales de la hormiga j
     tbevap(1,j)=bevap(2);  tbevap(2,j)=bevap(3); %(2,m) Vector que
almacena las coordenadas de la mejor solución
                                                  %que encuentra cada
hormiga en la iteración que acaba de terminar
    % ---------------------------OPTIMIZACION DE ALMACENAMIENTO DE
MENSAJES----------------------
    if bevap(1)>apt(1,j,CICLO)      % Compara con su resultado del ciclo
anterior y almacena el resultado solo si la hormiga encontro un mejor
solución
       tbevap2(2,j)=bevap(2);  tbevap2(3,j)=bevap(3);

tbevap2(1,j)=aptitud(Q,goldsteinprice(tbevap2(2,j),tbevap2(3,j)),1);
    else
       tbevap2(1:3,j)=-1;
    end
 end   %--------fin for j
apt(:,:,CICLO+1) = hormiguero(Q, m, tbevap,obfunci); %Evaluación de la
aptitud en la ubicación actual de todas las hormigas
     %display([CICLO,apt(1,j),apt(2,j),apt(3,j)]);
```

```
     mensajes = message(tbevap2,nmens,m, mensajes);
     %------------
%fin for ciclos
%pos
end
toc
disp(toc)
bsol
disp('X1=')
disp(bsol(2,1))
disp('X2=')
disp(bsol(3,1))

hold on
%------------ GRAFICA LAS HORMIGAS EN EL TRANSCURSO DE LOS CICLOS--------
p=1;
%grafique
  if CICLO>20;
   ra=ceil(CICLO/20);
   for c=1:ra:CICLO
      subplot(5,5,p), grafique, plot(apt(2,:,c),apt(3,:,c),'.')
      %title('Iteración No. ')          axis([-5,5,-5,5]);          p=p+1;
   end
     subplot(5,5,p), grafique,
plot(apt(2,:,CICLO+1),apt(3,:,CICLO+1),'.')
  else
     for c=1:CICLO
        %subplot(5,4,p),
        grafique
        plot(apt(2,:,c),apt(3,:,c),'.')
      %title('Iteración No. ')
      axis([-5,5,-5,5]);
        p=p+1;
     end
 end
 %----------------- graf
 figure
for I=1:CICLO
   hold on
   grafique
  plot(apt(2,1,I),apt(3,1,I),'.')
  plot(Mvgj(1,1,I),Mvgj(2,1,I),'+')
   for s=1:nspo
      plot(MGSPO(s,1,1,I),MGSPO(s,2,1,I),'*')
   end
   axis([-5,5,-5,5]);
 hold off
 pause(0.25)
  %clf;
end

%-----------------------------  GRAFICA UNA HORMIGA CON SUS SPOTS
```

```
else

%-------------------------S I   L A   F U N C I O N   S E L E C C I O N
A D A   F U E   T H O M A S : --------

%              COMIENZO DE ITERACIÓN
iter=input('Introduzca el número de iteraciones deseado? (100 óptimo)
:_');
m=input('Introduzca el número de hormigas deseado? (50 óptimo)  :_');
%-------------------VAR GLOBALES

global P PE MESES Qrm punter tbevap Vrest Cthom;
%--------------- CONSTANTES
Cthom=0;
nspo = 5;                   %definición del número de spot para cálculo
del centro de gravedad
Q=0.1;
nmens = ceil(m*.15);                %nmens es el número de mensajes
mensajes=-ones(7,nmens);
cuentmens=0;  cuentcentg=0;
tic;
%--------VARIABLES POR SOLICITAR
format long;
%RANGOS DE VARIABLES Y PRESICION PARA MAYOR ODTIMIZACION (VAR(MAX, MIN,
PRESI))

a=[.8 1 .001]; b=[10 350 1]; c=[0.001 .9 .001]; d=[.001 1 .001]; Sw0=[0
500 1]; Sg0=[0 500 1];
%-------PARA EJEMPLO LIBRO HIDROLOGIA DE SERGIO SERRANO---------

% Ingresar aquí las precipitaciones (en mm)
P=input('Ingrese el vector de precipitaciones en milimetros entre
corchetes y separado de espacios. ejmp [4 5 6 7]: ');
%P = [55 72 101 110 220 135 43   25 78 97 140 99];
% Ingresar aquí las evapotranspiraciones potenciales (en mm)
PE=input('Ingrese el vector de evapotranspiraciones potenciales en
milimetros entre corchetes y separado de espacios. ejmp [4 5 6 7] ');
%PE = [32 45 60 85 122 146 166 201 154 101 55 43];
% Ingresar aquí los caudales observados (en m3/s) Qrm=input('Ingrese el
vector de caudales observados en mm/mes entre corchetes y separado de
espacios. ejmp [4 5 6 7] ');
%Qrm= [186.556 69.565 92.791 99.125 197.588 120.491 37.343 21.541 66.070
82.540 121.125 84.957];

%Manteniendo el mismo principio explicado en el modelo para
GoldsteinPrice
%-------------------
MESES=length(P);   %numero de datos por calcular, igual al ingresado
pos(:,:)=zeros(6,1);   %Inicialización de la matriz de almacenamiento en
ceros
```

```
MGSPO(1:6,1:nspo,1:m,1:CICLO)=-10;% Matriz de coordenadas de los spots de
todas las hormigas en todos los ciclos(para graficar)
bsol=ones(7,1)*-1;
%------------------------crear vectores de Optimizacion------------------
%Cada vector contiene los valores que puede tomar cada variable
Va=(a(1):a(3):a(2));  Vb=(b(1):b(3):b(2));  Vc=(c(1):c(3):c(2));
Vd=(d(1):d(3):d(2)); VSw0=(Sw0(1):Sw0(3):Sw0(2));
VSg0=(Sg0(1):Sg0(3):Sg0(2));
%Se añade la longitud de cada vector al vector de información de la
variable
a=[a length(Va)];   b=[b length(Vb)];  c=[c length(Vc)];  d=[d
length(Vd)];  Sw0=[Sw0 length(VSw0)]; Sg0=[Sg0 length(VSg0)];
MVval(:,:)=[a;b;c;d;Sw0;Sg0];       %Se crea una matriz con la
información de las variables(6,4)
% temporal para calibrar Qrm=QRM.*2592/area;       %Convierte Caudal de
M3/s  a (mm/año)
%-------------------------------------------------------------------------
%pos es la matriz de posiciones(6,m) de las hormigas, las cuales son
ubicadas aleatoriamente en el terreno,
% Solo para calculos del ciclo en ejecución.
punter=round(rand(6,m).*1000);
for j=1:m-3
    while punter(1,j)>=a(4)
      punter(1,j)=round(rand*1000*.4);
   end
   while punter(2,j)>=b(4)
      punter(2,j)=round(rand*1000*.6);
   end
   while punter(3,j)>=c(4)
      punter(3,j)=round(rand*1000);
   end
   while punter(4,j)>=d(4)
      punter(4,j)=round(rand*1000);    end
   while punter(5,j)>=Sw0(4)
      punter(5,j)=round(rand*1000*.7);
   end
   while punter(6,j)>=Sg0(4)
      punter(6,j)=round(rand*1000*.7);
   end
    for vi=1:6
       if punter(vi,j)<=0 punter(vi,j)=1;
       end
    end        %Las líneas anteriores evitan que las pociciones del
vector para cada variable esten fuera de los respectivos rangos
   pos(:,j)=[Va(punter(1,j)); Vb(punter(2,j)); Vc(punter(3,j));
Vd(punter(4,j)); VSw0(punter(5,j)); VSg0(punter(6,j))];
end
   pos(:,m)=[MVval(:,1)];
   pos(:,m-1)=[MVval(:,2)-(MVval(:,3).*round(MVval(:,4)./6))];
   pos(:,m-2)=[MVval(:,1)+(MVval(:,3).*round(MVval(:,4)./2))];
   punter(:,m)=1;
punter(1,m-1)=find(Va==pos(1,m-1));
```

```
punter(2,m-1)=find(Vb==pos(2,m-1));
punter(3,m-1)=find(Vc==pos(3,m-1));
punter(4,m-1)=find(Vd==pos(4,m-1));
punter(5,m-1)=find(VSw0==pos(5,m-1));
punter(6,m-1)=find(VSg0==pos(6,m-1));
punter(1,m-2)=find(Va==pos(1,m-2));
punter(2,m-2)=find(Vb==pos(2,m-2));
punter(3,m-2)=find(Vc==pos(3,m-2));
punter(4,m-2)=find(Vd==pos(4,m-2));
punter(5,m-2)=find(VSw0==pos(5,m-2));
punter(6,m-2)=find(VSg0==pos(6,m-2));
   %-------------------------------------------------------------------------
%apt es la relación del valor de la función con la aptitud por iteración
de
%cada hormiga. Matriz de (7 por m), almacena también los punteros de los
vectores.
apt = hormiguero(Q, m, pos, obfunci);  %Como variable global se le envia
la matriz punter
%-------------------------------------------------------------------------
   %Creación del vector de Mensajes (7,30)
   %Contiene el valor de aptitud de las nmens mejores posiciones del
dominio
%de la función
mensajes = message(apt(:,:,CICLO),nmens,m, mensajes);
%-------------------------------------------------------------------------
 mspot=zeros(7,nspo,m);       %matriz que guarda las coordenadas de los
spot(nspo) de cada hormiga(m)
for CICLO=1:iter
%-------------------------------------------------------------------------
%Calcula de la distancia de movimiento permitida para cada hormiga
%Expresado en terminos de porcentaje del rango de cada variable, es decir
%número de indices que se podrá mover en el vector de cada variable desde
%su posición actual.
rmov = distancia(MVval,apt(:,:,CICLO),m,obfunci);
%-------------------------------------------------------------------------

% CANAL DE ESTIMERGIA
%-------------------------------------------------------------------------
%Calculo del centro de Gravedad
% para calcular el centro de gravedad debe calcular algunos elementos
 cont=0; rdis=0; acum=0; vgj=zeros(7,1); %vgj= Matriz que almacena los
centros de gravedad de la hormigas
 for j=1:m
   %Calculo De La Distancia Promedio Entre Individuos para j
         for i=1:m
            if i~=j
               temp=sqrt(sum((punter(:,j,CICLO)-punter(:,i,CICLO)).^2));
               acum=temp+acum;
               cont=cont+1;
            end
         end
```

```
      dprom=acum/cont;                              % el ciclo anterior
calcula la distancia promedio entre 2 agentes (delta medio)
% CREACION Y UBICACION DE SPOTS
% Los spots se usan para calcular el centro de gravedad de cada hormiga.
%Ver la Ecuación de centro de gravevdad en el documento
      wijacum=0;
      for n=1:nspo
         %------------- DEBO EVITAR QUE SE PASE DE LOS BORDES DEL RANGO,
SOBRE TODO QUE SE SALGA ---
         vtemp(:,j)=round(rand*rmov(2:7,j));    % coordenadas aleatorias
a partir del radio de movimiento de cada hormiga
                                                % Recupera rmov de la
fila 2 a la 7 (2:7), porque la fila 1 guarda la desviación máxima en
porcentaje
         if round(rand)>0.5                  % Para que explore mejor,
entonces es aleatorio sumar o restar coordenadas
            vtemp(:,j)=vtemp(:,j)+punter(:,j,CICLO);  % suma de deltas a
los punteros de cada variable de coordenada para todas las hormigasant
         else
            vtemp(:,j)=punter(:,j,CICLO)-vtemp(:,j);  % resta de deltas a
los punteros de cada variable de coordenada para todas las hormigasant
         end
         evalimu= vtemp(:,j)>MVval(:,4);  % Vector binario que evalua que
no se salga del rango por arriba
         evalimd= vtemp(:,j)<1;  % Vector binario que evalua que no se
salga del rango por abajo
         i=1;
         while i<=6
            if evalimu(i,1)==1
               mspot(i,n,j,CICLO)=MVval(i,4);
            elseif evalimd(i,1)==1              % Evalúo que no se salga
de los rangos según el resultado binario
               while vtemp(i,j)>MVval(i,4) | vtemp(i,j)<1
                  vtemp(i,j)=round(rand*rmov(i+1,j));
                  mspot(i,n,j,CICLO)=vtemp(i,j)+punter(i,j,CICLO);
               end
            else
               mspot(i,n,j,CICLO)=vtemp(i,j);   %Almacenamiento de nuevos
punteros cercanos a cad hormiga para calcular el centro de gravedad.
            end
            i=i+1;
         end
         mspotth=[Va(mspot(1,n,j,CICLO)); Vb(mspot(2,n,j,CICLO));
Vc(mspot(3,n,j,CICLO)); Vd(mspot(4,n,j,CICLO)); VSw0(mspot(5,n,j,CICLO));
VSg0(mspot(6,n,j,CICLO))];
                                       % (6,1) Matriz con valores por
variable
         mspot(7,n,j,CICLO)=thomas(P,PE,mspotth,1);
                                       % (8,10,m,CICLO) Guarda punteros,
Evaluación de la función en los spots y aptitud
          mtemp=mspot(:,:,j,CICLO);    %Extracción matriz bidimensional
para calculo
```

```
         mspot(8,n,j,CICLO)=aptitud(Q,mspot(7,n,j,CICLO),2);
         % almacenamiento de feromona s/n valor de función
         temp2=0;
         for sp=1:6
            temp2=(punter(sp,j,CICLO)-mspot(sp,n,j,CICLO))^2+temp2;
         end
         rdisp(j,n)=sqrt(temp2)/100;
         %calculo de distancia entre el spot i y la ant j.
         wij(n)=(dprom/2)*exp(-mspot(8,n,j,CICLO)*rdisp(j,n));
         %calculo de wij
         wijacum=wij(n)+wijacum;  %Sumatoria de wij
      end
      Gj=zeros(6,1);
      for n=1:nspo                  %Cálculo de Gj, es acumulado por cada
hormiga
         for sp=1:6

Gj(sp,1)=round(Gj(sp,1)+mspot(sp,n,j,CICLO)*(wij(n)/wijacum));
            if Gj(sp,1)<=0
               Gj(sp,1)=abs(round(rand*100-50));
            elseif Gj(sp,1)>=MVval(sp,4)
               Gj(sp,1)=MVval(sp,4)-1;
            end
         end
      end
      for sp=1:6
         vgj(sp,j)=Gj(sp,1);     % Guarda los punteros de los centros de
gravedad de cada hormiga
      end
      vgjv=[Va(Gj(1,1)); Vb(Gj(2,1)); Vc(Gj(3,1)); Vd(Gj(4,1));
VSw0(Gj(5,1)); VSg0(Gj(6,1))];
      vgj(7,j)=aptitud(Q,thomas(P,PE,vgjv,1),2);
      %----------------------- OPTIMIZACION Sw0
      %if abs(Vrest(1))>20
       %  vgjv(5,1)=(VSw0(Gj(5,1))+round(Vrest(1)*1.5));
       %  vgj(5,j)=find(VSw0==vgjv(5,1));
       %  vgj(7,j)=aptitud(Q,thomas(P,PE,vgjv,1),2);
      %end
      %-----------------------
   end         %end del for j linea 325
   MGSPO(:,:,:,CICLO)=mspot(1:6,:,:,CICLO); %Almacenamiento de spot para
graficarlos
   Mvgj(:,:,CICLO)=vgj;             %Almacenamiento de centros de
gravedad para graficarlos
   %-------------------------------------------------------------------
%-------MOVIMIENTO ANTS-------------
for j=1:m
   bevap=apt(:,j,CICLO); % (7,m) Almacenamiento temporal de la mejor
aptitud por hormiga(valor inicializado en el punto de arranque
   antj=punter(:,j,CICLO);
   %----------------LECTURA DE MENSAJE---------------
```

```
     PM=ceil(10*rand); % (1,1)Selección aleatoria de una posición para el
vector mensajes
     nm(:,1)=mensajes(:,PM); %(7,1) Obtención de la información del
mensaje selecionado
     while nm(2:7,1)-antj==0
        PM=ceil(10*rand); % (1,1)Selección aleatoria de una posición para
el vector mensajes
        nm(:,1)=mensajes(:,PM); %(7,1) Obtención de la información del
mensaje selecionado
     end
     mensajes(:,PM)=-1;
     %--------------------------COMPARO SI ME MUEVO HACIA EL MENSAJE O
HACIA EL CENTRO DE GRAVEDAD----------
     if      nm(1,1)>vgj(7,j)       %Compara Aptitudes del mensaje y del
centro de gravedad  para la hormiga j y elige la mayor          %En las
siguientes lineas se Calcula dirección y avance para moverse hacia el
mensaje, son coordenadas afectadas
          % por los cosenos directores dadas como coordenadas locales
desde la Ubicación de la hormiga (angulo y avance)
          cuentmens=cuentmens+1;
         % sprintf('mens %d ',cuentmens)
          for sp=1:6
           local(sp,1)=nm(sp+1,1)-antj(sp,1);   % Guarda el número de
punteros diferencia entre centro y la ubicación de la hormiga
          end
       else
          % Aquí Calcula dirección y avance para moverse hacia el Centro
de Gravedad, son coordenadas calculadas con los cosenos directores
          % como coordenadas locales desde la Ubicación de la hormiga,
son (angulo y avance)
          cuentcentg=cuentcentg+1;
          %sprintf('cent %d ',cuentcentg);
          for sp=1:6
           local(sp,1)=vgj(sp,j)-antj(sp,1);   % Guarda el n{umero de
punteros diferencia entre centro y la ubicación de la hormiga
        end
     end
     ldesp(7,1)=sqrt(sum(local.^2));       % hiper diagonal del
movimiento
     ldesp(1:6,1)=local(:,1)/ldesp(7,1);   % (7,1) las posiciones 1:6
guardan los cosenos directores de cada dimensión o variable
     Vdesp=ldesp(7,1)/max(abs(local));     % calculo el valor de avance
en función de la variable de mayor rango de movimiento
    for  p=1:max(abs(local))-1
         dm=floor(ldesp(1:6,1)*p*Vdesp);   % Cálculo de punteros de
avance a partir del origen de la hormiga
         xpunt=punter(:,j,CICLO)+dm;               % Se le suman los
punteros de la hormiga
         xpos=[Va(xpunt(1,1)); Vb(xpunt(2,1)); Vc(xpunt(3,1));
Vd(xpunt(4,1)); VSw0(xpunt(5,1)); VSg0(xpunt(6,1))];
         tevap=aptitud(Q,thomas(P,PE,xpos,1),2); % Evaluación de la
función y la aptitud(feromona)
```

```
        if tevap>bevap(1,1)

           bevap=[tevap;xpunt];           % Almacenamiento del mejor
resultado por hormiga en el ciclo
        end
    end
      if bsol(1,1)<bevap(1,1)
         bsol=bevap;                          % Almacenamiento del mejor
resultado global, guarda punteros
         bsolval=[Va(bsol(2,1)); Vb(bsol(3,1)); Vc(bsol(4,1));
Vd(bsol(5,1)); VSw0(bsol(6,1)); VSg0(bsol(7,1))];
      end
      pos(:,j)=xpos;          %Almacenamiento de las coordenadas finales
de la hormiga j
      punter(:,j,CICLO+1)=xpunt;
     tbevap(1:6,j)=bevap(2:7,1);    %(6,m) Vector que almacena las
coordenadas de la mejor solución
                                     %que encuentra cada hormiga en la
iteración que acaba de terminar
     tbevapv(1:6,j)=[Va(tbevap(1,j)); Vb(tbevap(2,j)); Vc(tbevap(3,j));
Vd(tbevap(4,j)); VSw0(tbevap(5,j)); VSg0(tbevap(6,j))];
    % ----------------------------OPTIMIZACION DE ALMACENAMIENTO DE
MENSAJES----------------------
    if bevap(1)>apt(1,j,CICLO)       % Compara con su resultado del ciclo
anterior y almacena el resultado solo si la hormiga encontro un mejor
solución
       tbevap2(1:7,j)=bevap(1:7);
       valtem(:,j)=[Va(tbevap2(2,j)); Vb(tbevap2(3,j)); Vc(tbevap2(4,j));
Vd(tbevap2(5,j)); VSw0(tbevap2(6,j)); VSg0(tbevap2(7,j))];
       %tbevap2(1,j)=aptitud(Q,thomas(P,PE,valtem,j),2); % Evaluación de
la función y la aptitud(feromona)
    else
       tbevap2(1:7,j)=-1;
    end
 end   %--------fin for j
apt(:,:, CICLO+1) = hormiguero(Q, m, tbevapv,obfunci); %Evaluación de la
aptitud en la ubicación actual de todas las hormigas
%display([CICLO,apt(1,j),apt(2,j),apt(3,j)]);
     mensajes = message(tbevap2,nmens,m, mensajes);
     %------------
       %fin for ciclos
  end
toc
disp(toc)
disp('')
disp('R E S U L T A D O S     D E    L A    C A L I B R A C I Ó N =')
disp('')
bsol
disp('Aptitud de la Función (Máximo valor 0.01) =')
disp(bsol(1,1))
bsolval
disp('a=')
```

```
disp(bsolval(1,1))
disp('b=')
disp(bsolval(2,1))
disp('c=')
disp(bsolval(3,1))
disp('d=')
disp(bsolval(4,1))
disp('Sw0=')
disp(bsolval(5,1))
disp('Sg0=')
disp(bsolval(6,1))
cuentmens
cuentcentg
end %fin seleccion tempral
end % fin for numero de corridas
toc
disp(toc)
%------------fin para el if de seleccion de función
%end
```

Función aptitud

```
function [apt] = aptitud(Q,val,num)
global Cllam;
switch num
    case 1,          %Caso Golsdtein - Price
       apt=Q*exp(-val/10000);
       Cllam=Cllam+1;
    otherwise
       apt=Q*exp(-val/100);   % CAso Thomas, A mayor error entre caudal
simulado y caudal real (val) menor será
  end                                          % la aptitud (apt). El valor
máximo ideal es Q=0.01
```

Función dirección

```
function dire=direccion(pos, vgj)

   antj=pos(:,j);
   gjact=vgj(:,j);
   [polarant(1,1),polarant(2,1)]=cart2pol(antj(1,1),antj(2,1))
   [polargrav(1,1),polargrav(2,1)]=cart2pol(gjact(1,1),gjact(2,1))
   lado=sqrt(polarant(2,1)^2+polargrav(2,1)^2-
2*polarant(2,1)*polargrav(2,1)*cos(abs(polarant(1,1)-polargrav(1,1))))
   if polarant(1,1)<(pi/2)
      dire=(asin((sin(abs(polarant(1,1)-
polargrav(1,1))))*polargrav(2,1)/lado))-(pi-polarant(1,1))
   elseif polarant(1,1)<pi
```

```
            dire=(asin((sin(abs(polarant(1,1)-
polargrav(1,1))))*polargrav(2,1)/lado))-(pi-polarant(1,1))
   elseif polarant(1,1)<(3*pi/2)
      dire=(asin((sin(abs(polarant(1,1)-
polargrav(1,1))))*polargrav(2,1)/lado))-(pi-polarant(1,1))
   else
      dire=(asin((sin(abs(polarant(1,1)-
polargrav(1,1))))*polargrav(2,1)/lado))-(pi-polarant(1,1))
   end
```

Función distancia

```
function[dist] = distancia(area,porbusq,m,fun)
if (fun==1)
   %----------------PARA FUNCION GOLDSTEIN - PRICE
 for j=1:m
    dist(1,j) = .35-0.0043*porbusq(1,j); % Se calcula el porcentaje
máximo de exploración teniendo en cuenta la aptitud inicial o del ciclo
anterior
 end
    dist(1,:) = dist(1,:)*area;         % se convierte en área local
    dist(1,:) = sqrt(dist(1,:)/pi);     % se convierte en radio, asumiendo
área circular
 else
     %----------------PARA FUNCION THOMAS
     %para el caso 2 la variable area no es el area, contiene la Matris
MVval con información de a,b,c,d,Sw0,Sg0
     % que en la columna 4 almacena el número máximo de posibles valores
de cada variable. y porbusq almacena apt
   for j=1:m
      dist(1,j) = .41-4*porbusq(1,j);           % Desviación máxima
permitida deacuerdo a la aptitud optenida
 end
    dist(2,:) = dist(1,:)*area(1,4);
    dist(3,:) = dist(1,:)*area(2,4);
    dist(4,:) = dist(1,:)*area(3,4);            %Conjunto de variables
que almacena la desviación máxixma
    dist(5,:) = dist(1,:)*area(4,4);            %que puede tener cada
variable
    dist(6,:) = dist(1,:)*area(5,4);
    dist(7,:) = dist(1,:)*area(6,4);
    %Criterio de actualización de distancia para cada iteración
end
```

Función goldsteiprice

```
function [gol] = goldsteinprice(x1,x2)
   gol=[1+(x1+x2+1)^2*(19-14*x1+3*(x1)^2-
14*x2+6*x1*x2+3*(x2)^2)]*[30+(2*x1-3*x2)^2*(18-32*x1+12*(x1)^2+48*x2-
36*x1*x2+27*(x2)^2)];
```

Función grafique

```
function grafique
n=51;
x=zeros(n,n);
y=zeros(n,n);
z=zeros(n,n);
%
xmin=-5; xmax=5; deltax=(xmax-xmin)/(n-1);
ymin=-5; ymax=5; deltay=(ymax-ymin)/(n-1);
%
cont=0;
for iy=1:n,
   for ix=1:n,
      y(iy,ix)=ymin+deltay*(iy-1);
      x(iy,ix)=xmin+deltax*(ix-1);
      x1=x(iy,ix); x2=y(iy,ix);
      z(iy,ix)=(1+(x1+x2+1)^2*(19-14*x1+3*x1^2-
14*x2+6*x1*x2+3*x2^2))*(30+(2*x1-3*x2)^2*(18-32*x1+12*x1^2+48*x2-
36*x1*x2+27*x2^2));
   cont=cont+1;
   end      % evalua la función en todo el espacio por filas
end
zmin=3;zmax=1000000;  %curvas minima y máxima
ipmax=1000000;v=[2 3 5 10 20 50 100 500 1000, 1500 5000 10000 100000
200000 500000 750000 ipmax];
% Definicion de cusvas de nivel
figh1 = figure('Position',[0 30 1000 680]); %deficicion de ventana
ancho=1000 h=680
subplot('Position',[0.05 0.05 0.90 0.90]) %define dimension del plano
cartesiano COMO PORCENTAJE DE LA VENTANA figh1
[c,h] = contourf(x,y,z,v); %dibuja las lineas de curva de nivel
especificadas en v
axis([xmin xmax ymin ymax])
line([0,0],[ymin,ymax],'LineStyle','-') %bibuja las lineas del centro
(ejes)
line([xmin,xmax],[0,0],'LineStyle','-')
xlabel('x1');ylabel('x2');
title('Goldstein-Price Function');
   caxis([0 ipmax]);
   clabel(c,v);
   colorbar
   grid on
   hold on
%ntheta = 2; % # incognitas (para G-P function son dos)
%ntheta1=ntheta+1;
%p=zeros(ntheta1,ntheta);
%p(1,1)=-2.0; p(1,2)= 1.0;
%p(2,1)= 0.0; p(2,2)=-2.0;
```

```
%p(3,1)= 1.0; p(3,2)=-1.0;
%p1=[p;p(1,1:ntheta)];
```

%line(p1(1:ntheta1+1,1), p1(1:ntheta1+1,2))

Función hormiguero

```
function[hor] = hormiguero(Q,m,pos,obfunci)
%Recordar que las los nombres de las variables con que recibe la función
hormiguerono sonlos mismos que le envia
%Esta aclaración se hace principalmente por la matriz pos, dependiendo de
donde se envie puede recibir la matriz pos, o tbevap
global P PE MESES CICLO punter tbevap;
for j=1:m
   if obfunci==1
      %tercera coordenada de pos es la iteración
      hor(1,j) = aptitud(Q,goldsteinprice(pos(1,j,1),pos(2,j,1)),1);
      hor(2,j) = pos(1,j,1);
      hor(3,j) = pos(2,j,1);
      %tercer argumento en aptitud es el tipo de aptitud a realizar
   else
      if CICLO==1
         hor(1,j) = aptitud(Q,thomas(P,PE,pos,j),2);
         hor(2,j) = punter(1,j);
         hor(3,j) = punter(2,j);
         hor(4,j) = punter(3,j);
         hor(5,j) = punter(4,j);
         hor(6,j) = punter(5,j);
         hor(7,j) = punter(6,j);
      else
         hor(1,j) = aptitud(Q,thomas(P,PE,pos,j),2);
         hor(2,j) = tbevap(1,j);
         hor(3,j) = tbevap(2,j);
         hor(4,j) = tbevap(3,j);
         hor(5,j) = tbevap(4,j);
         hor(6,j) = tbevap(5,j);
         hor(7,j) = tbevap(6,j); % Se almacena la aptitud y los punteros
para de ubicación para cada hormiga
         %ultimo argumento en aptitud es el tipo de aptitud a realizar
      end
   end
end
```

Función messagge

```
function[mensajes] = message(apt, nmens, m, temp)
global PM;
tmp = cat(2,apt,temp);           %Concatena el la matriz de mejor
solución tbevap y la matriz mensajes
```

```
[ordenmen,posini]=sort(tmp(1,:));  %ordeno de menor a mayor para
seleccionar las 10 mejores ubicaciones para generarlas como mensajes
i=2;
mensajes(:,1) = tmp(:,posini(nmens+m));
for j=2:nmens
   tmpc=tmp(:,posini(nmens+m+1-i));
   if mensajes(1,j-1)~= tmpc(1)     %Almacena evitando que repita
coordenadas
      mensajes(:,j) = tmp(:,posini(nmens+m+1-i));     %los nmens mejores
mensajes
   else
      while  mensajes(:,j-1)== tmp(:,posini(nmens+m+1-i)) & m+j>i
         i=i+1;
      end
      mensajes(:,j)=-1; %tmp(:,posini(nmens+m+1-i));
   end
 i=i+1;
end
```

Función thomas

```
function thom=thomas(P,PE,pos,j)
global MESES Qrm Vrest Qsim Cthom;
summes=0;
a=pos(1,j); b=pos(2,j); c=pos(3,j); d=pos(4,j); Sw0=pos(5,j);
Sg0=pos(6,j);
%a=.9; b=20; c=.1; d=.1; Sw0=150; Sg0=200;
for l=1:MESES
   W=P(l)+Sw0;
   Y=(W+b)/(2*a)-sqrt(((W+b)/(2*a))^2-W*b/a);
   Sw=Y*exp(-PE(l)/b);
   Ro=(1-c)*(W-Y);
   Rg=c*(W-Y);
   Sg=(Rg+Sg0)/(d+1);
   Qg=d*Sg;
   Qsim(l)=Ro+Qg;
   Vrest(l)=Qrm(l)-Qsim(l);
   summes=summes+(Qrm(l)-Qsim(l))^2;
   Sw0=Sw;
   Sg0=Sg;
end
Cthom=Cthom+1;
thom=sqrt(summes)/12;
% Se Evalua el caudal simulado con los valores de las variables que tiene
cada hormiga.
% Cada hormiga evalúa los 12 meses y finalmente devuelve el error
calculado como norma cuadrática normalizada.
```

ANEXO 2

RESULTADOS DE LA EVALUACIÓN DE THOMAS PARA LA CUENCA DE SUBACHOQUE

Los resultados obtenidos dependen de las variables que se le permita imprimir al programa, en este caso se puede ver el número de iteraciones, de hormigas, los vectores de entrada de precipitaciones, evapotranspiraciones potenciales y caudales reales, el tiempo de ejecución, el número de evaluaciones por mensajes y por centros de gravedad, y en ***bsol*** se almacenan la aptitud y punteros de cada variable en el orden(aptitud, a,b,c,d,S_{w0},S_{g0}), y en vector ***bsolval*** se muestran los valores de cada variable en el orden (a,b,c,d,S_{w0},S_{g0}).

```
iter =   200
m =   800
Ingrese el vector de precipitaciones en milimetros entre corchetes y separado de espacios. ejmp [4 5 6 7]:
[40.62 27.17 92.96 53.4    70.94    51.8      42.85    39.29    77.56    114.7    89.93    48.55]
Ingrese el vector de evapotranspiraciones potenciales en milimetros entre corchetes y separado de espacios.
ejmp [4 5 6 7] [49.4         35.94    38.87    55.03    47.59    36.45    45.7      35.06    35.92    55.21
          55.73    46.54]
Ingrese el vector de caudales observados en mm/mes entre corchetes y separado de espacios. ejmp [4 5 6 7]
[4.22    9.35      9.64      11.76    9.25      8.52      6.31      5.42      7.49      28.46    17.79    6.57]

elapsed_time =    3.872234000000000e+003
                  3.872234000000000e+003
cuentmens =   377
cuentcentg =      159623
elapsed_time =    3.872234000000000e+003
                  3.872234000000000e+003
3872/60
ans =  64.53333333333333
bsol
bsol =
  1.0e+002 *

   0.00099143325163
   1.43000000000000
   1.43000000000000
   7.22000000000000
   0.08000000000000
   0.59000000000000
   1.16000000000000

bsolval
bsolval =
  1.0e+002 *
```

0.00942000000000
1.52000000000000
0.00722000000000
0.00008000000000
0.58000000000000
1.15000000000000