Universidad de las Ciencias Informáticas

Facultad 1

Título: Propuesta de un modelo para la optimización del cronograma de un proyecto utilizando Inteligencia Colectiva

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autor(es): Ismaray Fagundo Guerrero

Yaima Nuñez Sierra

Tutor(es): Lic. Joel Arencibia Ramírez

La Habana, 27 de junio de 2008

Año 50 de la Revolución.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Ismaray Fagundo Guerrero	Yaima Nuñez Sierra	Joel Arencibia Ramírez
_____	_____	_____
Firma del Autor	Firma del Autor	Firma del Tutor

DATOS DE CONTACTO

Joel Arencibia Ramírez
Licenciado en Ciencias de la Computación, Universidad Central de Las Villas, 2002
Graduado del Curso de Entrenamiento “Web Application Specialist for e-Government Promotion (open source)”, Okinawa Internacional Centre, Japón, 2006
Maestrante en Informática Aplicada

OPINIÓN DEL TUTOR DEL TRABAJO DE DIPLOMA

Título: Propuesta de un modelo para la optimización del cronograma de un proyecto utilizando Inteligencia Colectiva
Autores: Ismaray Fagundo Guerrero y Yaima Nuñez Sierra

El tutor del presente Trabajo de Diploma considera que durante su ejecución el estudiante mostró las cualidades que a continuación se detallan.

Durante la ejecución del presente trabajo de diploma, las diplomantes mostraron un nivel de independencia muy alto, teniendo en cuenta que realizaron casi la totalidad del trabajo separadas del tutor, cumpliendo al pie de la letra las orientaciones que este les daba a distancia. El trabajo realizado presenta una alta originalidad, al dar los primeros pasos en la aplicación de algoritmos de Inteligencia Artificial a la reducción del tiempo de ejecución de un proyecto, tema de vital importancia en nuestra universidad. Las estudiantes mostraron un alto nivel de laboriosidad, evidenciado en la extensa bibliografía revisada y su esfuerzo en la implementación de un algoritmo complejo, dando como valor agregado a su aplicación al tema específico tratado en este trabajo de diploma, una implementación genérica del mismo que puede ser utilizada en trabajos de investigación futuros. El hecho de que hayan cumplido en tiempo con todos los objetivos propuestos en el desarrollo de este trabajo, muestran un alto nivel de responsabilidad de las tesistas, que no se dejaron vencer por la falta de compañía directa de su tutor en ocasiones y cumplieron en tiempo todas las tareas programadas. El trabajo realizado evidencia además la creatividad de las autoras, quienes muchas veces se dejaron llevar por sus impulsos para realizar aportes que, luego de introducidos, revisaban con el tutor. A pesar de no contar con experticia precedente en tema de la Inteligencia Artificial, se esforzaron mucho en aras de lograr la investigación de calidad que hoy están presentando.

Sus resultados presentan una excelente calidad y nivel de completitud. El documento muestra de una manera clara el trabajo realizado, tiene una buena estructura, ortografía y redacción. Tanto el trabajo realizado como el documento presentado cuentan con un excelente nivel científico-técnico y se corresponden con lo exigido para una tesis de este tipo.

Por todo lo anteriormente expresado considero que los estudiantes están aptos para ejercer como Ingenieros en Ciencias Informáticas; y propongo que se le otorgue al Trabajo de Diploma la calificación

de **5 puntos.** Además considero que el trabajo tiene condiciones para presentarlo en eventos científicos y ser publicado.

Joel Arencibia Ramírez

Firma

Fecha

AGRADECIMIENTOS

A la UCI y a Fidel por permitirme formar parte de este proyecto.

A mi Tutor Joel por su gran dedicación y ayuda, por la atención que me ha brindado en todo momento, por las noches sin dormir…

A mi familia por darme apoyo en todo momento principalmente a mis padres, sin ustedes hubiera sido imposible llegar hasta aquí.

A mis amigos: la chiqui, bichote, Odeymi, Geiser, a las integrantes de mi segundo apartamento la riqui, ana, yoa a mi peluquera yilian, al súper Dacho, Susana en fin todos aquellos que de una forma u otra me brindaron su cariño todos estos años y me enseñaron el verdadero significado de amistad.

A mi compañera de tesis.

Ismy

A mis padres, Hilda y Carlos, por formarme y apoyar mis decisiones en todo momento, por quererme y ser mis guías.

A toda mi familia, mis abuelos, tíos, primos, ….

A mi novio Michel por su amor incondicional y su paciencia en tantos años.

A mi tutor, por toda su ayuda, paciencia y confianza, a pesar de la distancia.

A mi compañera de tesis, por aguantar todas las largas horas de trabajo.

A mis amigos de aquí que son muchos, en especial: Yoanna, Yilian, Ana, Rives, Yaimara, al grupo 5.

A mis amigas de allá por sentir que en estos años nuestra amistad no ha cambiado.

A los profesores que nos ayudaron con la tesis y a los que en estos 5 años nos han educado.

A la UCI, a Fidel, por permitirme vivir el sueño que es estudiar aquí.

Yaima

DEDICATORIA

A mis padres por guiarme y estar siempre a mi lado.
A mi fideo, A mis abuelos.
A canelote.

Ismy

A mis padres.
A mi familia.
A mi amor Michel.

Yaima

RESUMEN

Todo proyecto de Software en su desarrollo se rige por un cronograma en el que se reflejan las actividades y el tiempo de duración de las mismas. Actualmente existen problemas con su elaboración pues no se confecciona de la mejor forma, debido a que no se cuenta con una herramienta que procese de algún modo la distribución de las actividades con vistas a reducir su duración. En nuestra Universidad existe un notable crecimiento de los proyectos productivos lo que implica un esfuerzo superior en la planificación de los mismos.

En el presente trabajo se implementa un modelo de Inteligencia Colectiva que permite reducir el tiempo de ejecución de los proyectos mediante una planificación óptima del cronograma. Para ello se realiza un análisis de los modelos de Inteligencia Colectiva más representativos: Optimización basada en Colonia de Hormigas, Optimización basada en Nube de Partículas y Optimización basada en Búsqueda por Difusión Estocástica.

Se concluye utilizar la Optimización basada en Colonia de Hormigas, por sus resultados significativos en la solución al Problema del Viajero Vendedor. Con vistas a solucionar el problema de la optimización del cronograma, se define un modelo para expresar el cronograma de un proyecto en forma de un grafo dirigido ponderado, sobre el cual se implementa el modelo de colonia de hormigas propuesto. Además, como valor agregado, se desarrolla una implementación genérica del modelo de colonia de hormigas que puede ser utilizado en problemas futuros.

PALABRAS CLAVE

Inteligencia Artificial, Inteligencia Colectiva, Colonia de Hormigas, Nube de Partículas, Búsqueda por Difusión Estocástica, Cronograma.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

INTRODUCCIÓN

La Universidad de las Ciencias Informáticas es un Centro de Educación Superior con la misión de formar profesionales comprometidos con la Patria, altamente calificados en la rama de la Informática pero además, de producir software y servicios informáticos, a partir de la vinculación estudio-trabajo como modelo de formación. Con el fin de lograr sus metas, cada año se definen un gran número de proyectos productivos, en los que nuestros estudiantes y profesores, a la vez que practican sus habilidades en el proceso de desarrollo de software, producen importantes sistemas tanto para su despliegue en el propio campus universitario como para el país y la exportación.

En muchos de estos proyectos existen problemas con la calidad del proceso de desarrollo de software, siendo los más comunes: mala gestión de requerimientos por parte del equipo de analistas, ausencia o mala calidad de la documentación, falta de gestión de riesgos y fundamentalmente, problemas relacionados con el cronograma de ejecución del proyecto. La ausencia, mala confección o falta de gestión del cronograma del proyecto, provoca en muchos casos que el tiempo de desarrollo se extienda indefinidamente y en algunos casos, que el proyecto fracase.

La manera más usual de representar el cronograma de un proyecto es mediante un Diagrama de Gantt. Sin embargo, se pudiera representar el mismo mediante un grafo, en el que los vértices representarán las diferentes tareas y las aristas, la dependencia que existe entre ellas, cuyo peso asociado, representaría el tiempo de ejecución de la tarea correspondiente al vértice del que parte, o lo que es lo mismo, el tiempo necesario para poder pasar de esa tarea, a la siguiente. Una búsqueda del camino mínimo para visitar todos los nodos del grafo nos conduciría al cronograma óptimo de ejecución del proyecto. Sin embargo, este problema pertenece a la clase NP-completo, que define problemas de difícil solución.

La Inteligencia Artificial es una de las ciencias de la Computación que se basa en dotar de inteligencia a las computadoras. Hay aplicaciones que conjugan la Ciencia de la Computación con la Inteligencia Artificial. Dentro de la Inteligencia Artificial se puede encontrar una técnica llamada Inteligencia Colectiva (Swarm Intelligence) que estudia el comportamiento colectivo en sistemas descentralizados y autoorganizados, que están generalmente compuestos de una población de agentes simples que interactúan con otros y con el entorno. Algunos ejemplos de estos sistemas son: las colonias de hormigas (ant colonies), manadas de pájaros (bird flocking), manadas de animales (animal herding), expansión de bacterias (bacterial growth), bancos de peces (fish schooling), optimización de enjambre

de partículas (particle swarm optimization) y búsqueda por difusión estocástica (stochastic diffusion search). Algunos de estos modelos pudieran ser útiles en la solución del problema de obtener el cronograma óptimo de ejecución del proyecto, o sea, aquel de menor tiempo de ejecución.

Situación Problémica:

Los cronogramas de ejecución de los proyectos, casi en su totalidad, se basan en la experiencia del planificador o líder del proyecto, los cuales definen las tareas a realizar, las dependencias que existen entre estas tareas y el tiempo de duración de cada una, según el proyecto en sí. Actualmente se representa el cronograma de un proyecto mediante un Diagrama de Gantt. Estos cronogramas, en su gran mayoría, no son refinados en función de reducir el tiempo total de ejecución del proyecto, motivado en gran parte porque los encargados de la elaboración de los mismos no cuentan con una herramienta que los asista en la definición de una planificación óptima, o sea, de tiempo de ejecución mínimo. Esto provoca que algunos proyectos sean diseñados para un tiempo mayor que el que en realidad debían tomar, lo cual unido a algunos retrasos que pudieran introducirse, debido a otros problemas, provoca que se dilate la entrega del producto final.

Después de analizar las ideas anteriores y la situación problémica se plantea como problema científico: ¿Cómo definir un cronograma óptimo en los proyectos productivos de la Universidad de las Ciencias Informáticas basándose en los modelos de Inteligencia Colectiva? Como resultado se define a los modelos de Inteligencia Colectiva como objeto de la investigación y al algoritmo de Inteligencia Colectiva seleccionado durante la investigación como campo de acción.

Se define como objetivo general:
Implementar un modelo de Inteligencia Colectiva que permita reducir el tiempo de ejecución de los proyectos mediante una planificación óptima del cronograma.

Para cumplir el objetivo general se definieron los siguientes objetivos específicos:

- Elaborar el marco teórico referencial atendiendo a una revisión de los aspectos teóricos conceptuales sobre modelos de Inteligencia Colectiva.
- Describir los modelos de Inteligencia Colectiva más comunes y sus aplicaciones reportadas.
- Seleccionar el modelo de Inteligencia Colectiva más adecuado a utilizar para la búsqueda de la planificación de tiempo mínimo.
- Desarrollar un modelo para representar el cronograma de un proyecto utilizando la teoría de Grafos a partir del modelo seleccionado.

- Implementar el algoritmo de Inteligencia Colectiva seleccionado.

Como solución al problema planteado surge la siguiente hipótesis: la implementación de un modelo de Inteligencia Colectiva permitirá obtener un cronograma de proyecto con tiempo de ejecución mínimo.

De esta forma se obtienen las siguientes variables de la investigación:
Variable Independiente: Modelo de Inteligencia Colectiva
Variable Dependiente: Cronograma de proyecto con tiempo de ejecución mínimo

Las tareas de la Investigación definidas son: caracterizar los modelos de Inteligencia Colectiva más representativos existentes en el mundo; describir los modelos de Inteligencia Colectiva más representativos, sus algoritmos y aplicaciones reportadas; determinar algunas técnicas de implementación para cada uno de los modelos de Inteligencia Colectiva seleccionados; analizar la existencia de algoritmos de Inteligencia Colectiva ya implementados y su factibilidad de ser utilizados; modelar el cronograma del proyecto utilizando la teoría de Grafos; seleccionar una técnica de implementación del modelo de Inteligencia Colectiva adecuada, que resuelva el problema de optimizar el tiempo de ejecución del proyecto; implementar el algoritmo de Inteligencia Colectiva seleccionado sobre el modelo de cronograma de proyecto definido; así como validar la eficacia del modelo implementado sobre la planificación real para un proyecto determinado.

Métodos Científicos de Investigación:

- Histórico-lógico porque se menciona de qué manera se realiza la planificación de los proyectos así como el surgimiento de los diferentes algoritmos tratados.

- Hipotético-Deductivo porque a través de la hipótesis se puede llegar a la conclusión de que con la implementación de un modelo de Inteligencia Colectiva se logrará una óptima planificación de los proyectos productivos en la Universidad.

- Deductivo porque a través de la investigación que se lleva a cabo se puede determinar cuál es el modelo más conveniente a utilizar en la planificación de proyectos.

- Modelación porque a través de este método se puede visualizar, construir el grafo donde estará representado el cronograma a optimizar.

Estrategia de la investigación:

- Descriptiva ya que nuestro trabajo persigue describir los algoritmos de inteligencia colectiva, seleccionar el más adecuado para resolver nuestro problema e implementarlo.

La estructura del informe está compuesta por tres capítulos:

Capítulo 1. Fundamentación teórica: En este capítulo se brinda una visión general de los conceptos y definiciones más importantes que facilitan la comprensión de los temas tratados en la investigación.

Capítulo 2. Análisis de algunos algoritmos de Inteligencia Colectiva: En este capítulo se realiza un estudio de los algoritmos de inteligencia colectiva, obteniéndose como resultado la selección del algoritmo a implementar.

Capítulo 3. Implementación y Resultados: En este capítulo se realiza la implementación del algoritmo seleccionado y se muestran los resultados que se obtienen a partir de esta implementación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se relacionan algunos conceptos referentes a Inteligencia Artificial, Inteligencia Colectiva, Algoritmos de Inteligencia Colectiva, Optimización y Cronograma de proyecto. Se realiza una breve descripción de los algoritmos de Inteligencia Colectiva así como aplicaciones reportadas. Además se realiza un estudio de las herramientas que se utilizan en la elaboración de cronogramas de proyecto.

1.1. Inteligencia Artificial, Inteligencia Colectiva, Algoritmos de Inteligencia Colectiva

1.1.1. ¿Qué es Inteligencia Artificial?

En 1943 al definirse la neurona como un dispositivo binario con varias entradas y salidas se comienza a utilizar el término de Inteligencia Artificial (IA), pero no fue definido formalmente hasta el año 1956 durante la Conferencia de Darthmounth donde se establecieron las bases de la IA como un campo independiente dentro de la informática. (García, 2004)

Varias han sido las definiciones que se han dado sobre IA:

Farid Fleifel Tapia describe a la IA como: "la rama de la ciencia de la computación que estudia la resolución de problemas no algorítmicos mediante el uso de cualquier técnica de computación disponible, sin tener en cuenta la forma de razonamiento subyacente a los métodos que se apliquen para lograr esa resolución". (Cepeda, et al.)

La IA es el arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren de inteligencia. (Kurzweil, 1990). (Cepeda, et al.)

La IA es el campo de estudio que se enfoca a la explicación y emulación de la conducta inteligente en función de procesos computacionales. (Schalkoff, 1990). (Cepeda, et al.)

La IA es el estudio de cómo lograr que las computadoras realicen tareas que, por el momento los humanos hacen mejor. (Rich, Knight, 1991). (Cepeda, et al.)

La IA es la rama de la ciencia de la computación que se ocupa de la automatización de la conducta inteligente. (Lugar y Stubblefied, 1993). (Cepeda, et al.)

La inteligencia artificial es aquella que trata de explicar el funcionamiento mental basándose en el desarrollo de algoritmos para controlar diferentes cosas. (García, 2004)

Se define como la técnica de software que los programas utilizan para dar solución a algún tipo de problema, pero tratando de asemejar el comportamiento inteligente que se observa en la naturaleza. (García, 2004)

La IA es un conjunto de técnicas, relacionadas o similares a lo que se llama “inteligencia” en humanos, necesaria para resolver en forma más o menos automática un amplio conjunto de problemas. En definitiva es una Ingeniería que se precia de integrar soluciones para resolver problemas complejos. (Torres, 2008)

Después de analizar las definiciones anteriores se puede concluir que:

La IA es aquella rama de la ciencia que estudia el funcionamiento mental para desarrollar algoritmos que resuelvan problemas imitando al ser humano o a sistemas sociales y naturales.

La IA combina varios campos, como la robótica, los sistemas expertos, entre otros. Existen varios estudios y aplicaciones, dentro de las que se encuentran las redes neuronales, el control de procesos o los algoritmos genéticos.

1.1.2. ¿Que es Inteligencia Colectiva?

La expresión "swarm intelligence"[1] fue introducida por Beni & Wang en 1989 en el contexto de los sistemas robóticos celulares. (Saravia, 2007)

La Inteligencia Colectiva (Swarm intelligence (SI)) es una técnica de Inteligencia Artificial basada en el estudio del comportamiento colectivo en sistemas descentralizados, autoorganizados. Estos sistemas están generalmente compuestos de una población de agentes simples que interactúan con otros y con el entorno. Aunque normalmente no hay una estructura de control centralizada dictando cómo deben comportarse los agentes individuales, las interacciones locales entre estos agentes usualmente guían el comportamiento global. (Arencibia, 2007)

Otras definiciones:

[1] Inteligencia Colectiva o Inteligencia de Enjambre.

"Algoritmos o mecanismos distribuidos de resolución de problemas inspirados en el comportamiento colectivo de colonias de insectos sociales u otras sociedades de animales". (Bioinformática, 2006-2007)

Los algoritmos de Swarm Intelligence (SI) son técnicas metaheurísticas de inteligencia artificial basados en el estudio de comportamientos colectivos presentes en sistemas de la naturaleza, generalmente de carácter descentralizado y autorganizativo. Dicho comportamiento social define los movimientos de las variables de decisión en el espacio de búsqueda y las orienta hacia soluciones óptimas. (García, 2007)

Ejemplos de estos Algoritmos de Inteligencia colectiva son: las colonias de hormigas (ant colonies), manadas de pájaros (bird flocking), manadas de animales (animal herding), expansión de bacterias (bacterial growth), bancos de peces (fish schooling), optimización de enjambre de partículas (particle swarm optimization), búsqueda de difusión estocástica (stochastic diffusion search).

1.1.3. Algoritmos de Inteligencia Colectiva

A continuación se describirán algunos algoritmos de Inteligencia Colectiva y sus aplicaciones en diferentes campos.

1.1.3.1. Optimización basada en Colonia de Hormigas

La metaheurística[2] Optimización con Colonia de Hormigas, conocida también por sus siglas en inglés, ACO, es una propuesta relativamente joven, surge como resultado de un cuidadoso análisis de las características de algoritmos que imitaban a las hormigas, sus orígenes estuvieron cuando comenzaron a estudiarse estrategias estocásticas a raíz de los trabajos de J. L. Deneubourg y otros sobre la hormiga argentina *Linepithema humile*. (Delgado, 2004)

El sistema de Colonia de Hormigas se basa en el comportamiento estructurado de una colonia de hormigas donde individuos muy simples de una colonia se comunican entre sí por medio de una sustancia química denominada feromona, estableciendo el camino más adecuado entre su nido y su fuente de alimentos. El método consiste en simular computacionalmente la comunicación indirecta que

[2] Algoritmo aproximado de propósito general consistente en procedimientos iterativos que guían una heurística subordinada combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda.

utilizan las hormigas para establecer el camino más corto, guardando la información aprendida en una matriz de feromonas. (Arencibia, 2007)

Al iniciar la búsqueda de alimento, una hormiga aislada se mueve a ciegas, es decir, sin ninguna señal que pueda guiarla, pero las que le siguen deciden con buena probabilidad seguir el camino con mayor cantidad de feromonas. (Arencibia, 2007)

Esto ocurría debido a que las hormigas que se movían por el camino más corto regresaban al nido con mayor rapidez que aquellas que lo hacían por el más largo, de manera que la concentración de feromona en esta vía crecía rápidamente haciéndose más atractiva para las hormigas cuando salían a buscar el alimento. De este modo, al cabo de cierto tiempo la mayoría de las hormigas se movían por la vía más corta. (Delgado, 2004)

La Figura 1.1 muestra de una manera gráfica cómo ocurre este proceso:

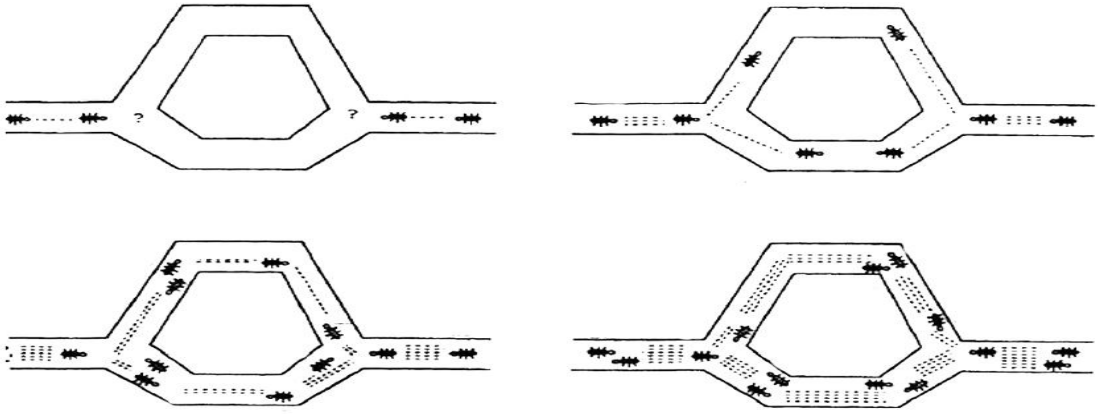


Figura 1.1 Recorrido de las hormigas

Aplicaciones

El primer problema que fue solucionado por un algoritmo de ACO fue el TSP[3], 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. Desde la primera aplicación del AS[4] por Dorigo en su tesis de Doctorado

[3] Traveling Salesman Problem o como se conoce en español, Problema del Viajante de Comercio o Viajante Vendedor.

[4] Ant System o Sistema de Hormigas

en 1991, se convirtió en un problema estándar para realizar pruebas en otros modelos posteriores que ofrecían un mejor rendimiento que el AS. (Alonso, et al.)

De forma cronológica, las dos aplicaciones que le siguieron fueron el problema de la asignación cuadrática (QAP) y el problema de la secuenciación de tareas (job-shop scheduling, JSP) en 1994. Entre las aplicaciones posteriores se encuentran las primeras aplicaciones de enrutamiento en redes en 1996 con el trabajo de Schoonderwoerd y otros y el trabajo sobre AntNet por Di Caro y Dorigo. (Alonso, et al.)

Para el año 1997 comienza a incrementarse el número de aplicaciones de ACO como resultado del primer articulo que se publica sobre ACO en 1996.

Aplicaciones reportadas en el año 1997 aunque fueron publicadas más tardes

- Problemas clásicos de enrutamiento de vehículos. (Bullnheimer, et al., 1999)
- Problemas de ordenación secuencia (Gambardella, et al., 2000) l.
- Problemas de secuenciación (flor shop scheduling, FSS). (Stützle, 1998)
- Problemas de coloras de grafos. (Ant can colour graphs, 1997)

A partir de este momento, muchos autores han utilizado la metaheurística ACO para solucionar un gran número de problemas de optimización combinatoria como la súpersecuencia común más corta, la asignación generalizada, la cobertura de conjuntos y varios problemas de la mochila y de satisfacción de restricciones, entre otros. (Alonso, et al.)

La ACO ha sido utilizada recientemente para aprendizaje automático, exactamente para el diseño de algoritmos de aprendizaje para estructuras de representación del conocimiento como las clásicas reglas lógicas (Parpinelli, et al., 2002), reglas difusas (Alcalá, et al., 2001) y redes bayesianas (Campos, et al., 2002), obteniéndose resultados prometedores.

Aplicaciones específicas

Houston, American Air Liquide, productora de gases industriales y medicinales, utiliza un programa basado en algoritmos de hormigas para planificar diariamente las rutas de distribución de su flota de camiones desde las 100 plantas de producción en EE UU hacia más de 6.000 localizaciones distintas. Otras compañías en Italia (Carnini, el grupo logístico número 1 o Pina Petroli) y Suiza (Migros, la

mayor cadena de supermercados) usan los algoritmos de hormigas para planificar la logística de reparto de alimentos perecederos, como leche y carne, o de gasóleo. (Cordón, 2007)

Otras aplicaciones exitosas son el diseño de planes de asignación de aviones a puertas de embarque en el aeropuerto internacional de Phoenix (Southwest Airlines); el desarrollo de un sistema de recomendaciones para portales de venta en Internet (Rightnow Technologies), o la generación de los caminos más rápidos para transmitir datos en redes de telecomunicaciones, empleado por varias operadoras de telefonía en Inglaterra y Francia. (Cordón, 2007)

El ayuntamiento de Sant Boi del Llobregat (ciudad de 80.000 habitantes de Barcelona) planifica las rutas de recogida de basuras mediante algoritmos de hormigas. (Cordón, 2007)

Herramientas desarrolladas entorno a Colonia de Hormigas

A continuación se presentan tres grupos de herramientas que se han desarrollado en torno a colonia de hormigas: educativas, específicas y de propósito general. (Escenarios para el aprendizaje de tendencias bioinspiradas, 2007)

El grupo de Herramientas educativas representan aplicaciones virtuales desarrolladas teniendo en cuenta los conceptos de la colonia de hormigas, que ayudan a entender e interactuar con dichos conceptos.

- StarLogo (FreeWare) (2007)

  Herramienta basada en la tortuga de Logo, que permite simular el comportamiento de una colonia de hormigas, permitiendo cambiar el número de hormigas, la fuente de comida y el porcentaje de feromona; utiliza su propio lenguaje procedimental (Logo)

- Ants Viewer 1.0 (Rennard, 2007)

  Applet desarrollado bajo Java, que permite ver el comportamiento de la colonia de hormigas en su búsqueda de la fuente de comida.

- Ant Farm Simulator (FreeWare) (Aguirre, 2007)

  Este juego, que simula la capacidad de las hormigas de coordinar en la tarea de reunión de alimentos, fue desarrollado en Visual Basic.

- SimAnt (SimAnt, Maxis Software Inc, 2007)

  Juego que simula el comportamiento de una colonia de hormigas, que se enfrenta a otras colonias de hormigas por la supremacía de la colonia, al estilo Age of Empires.

Las herramientas específicas tienen como tarea fundamental resolver un problema específico por medio de la abstracción de los conceptos básicos de colonia de hormigas.

- ACOTSP.V1.0 (GPL) (Stützle, 2007)

  Está desarrollada en lenguaje Ansi-C para solucionar problemas del TSP. Los algoritmos utilizados son Ant System, Elitist - Ant System, Max-Min.

- Antnet-1.1 (GPL) (Farooq, 2007)

  Permite solucionar el problema de encaminamiento de redes, el cual consiste en dirigir datos de una fuente a unos nodos destino, maximizando el funcionamiento de la red; está desarrollado en Omnet++, con el algoritmo Ant Net Routing.

- Hc-mmas-ubqp (GPL) (Blum, 2007)

  Herramienta exclusiva para la solución del hipercubo cuadrático (UBQP). Consiste en poner en práctica algoritmos de optimización de colonias de hormigas con intervalos de [0,1]. Fue desarrollado en C++ con el algoritmo de Min-Max.

- GUIAnt-Miner (GPL) (Meyer, et al., 2007)

  Herramienta para extraer reglas de clasificación de datos. Está desarrollada sobre Java con el algoritmo Ant-Miner (minería de datos).

- Hamiltonian Cycle Problem (HCP) (Hamiltonian cycle problem., 2007)

  Prueba si un grafo contiene un ciclo hamiltoniano[5] o no.

- Sequential Ordering Problem (SOP) (Sequential ordering problem., 2007)

  Problema de TSP asimétrico, el cual consiste en dar una restricción de visitar un nodo determinado antes de pasar a otro.

[5] Es un ciclo simple que visita todos los vértices.

- Capacitated Vehicle Routing Problem (CVRP) (Capacitated vehicle routing problem., 2007)

  Dado un conjunto de camiones, posibles elementos para cargar los camiones con sus pesos respectivos y las distancias hacia un destino, minimiza el número de viajes de los camiones, respetando la capacidad máxima de los mismos.

- Atsp (Atsp, 2007)

  Se deben visitar todos los nodos desde un nodo inicial a un nodo final y viceversa, teniendo en cuenta que la distancia del nodo i al nodo j puede ser diferente del nodo j al nodo i, minimizando las distancias recorridas.

Una herramienta de propósito general que se desarrolló para tratar diferentes problemas con colonia de hormigas es:

- Operations Research 3.0 (Addlink Software Científico.Operations Research 3.0, 2007)

  Es necesario tener el software Matemática para poder usar esta herramienta, que utiliza el algoritmo "Simulated annealing"[6]. Es una heurística para problemas de optimización global; es decir, encontrar una buena aproximación al óptimo global de una función en un espacio de búsqueda grande.

1.1.3.2. Optimización basada en Nube de Partículas

La Optimización basada en Nubes de Partículas o Particle Swarm Optimization (PSO) es uno de los algoritmos basado en Inteligencia Colectiva desarrollado en 1995.

Los orígenes del PSO como método estocástico de optimización global se remontan a los estudios iniciados por Kennedy y Eberhart , los que tenían como objetivo inicial simular gráficamente el movimiento sincronizado e impredecible de grupos tales como los bancos de peces o las bandadas de aves, intrigados por la capacidad de estos grupos para separarse, reagruparse o encontrar alimento. (Pérez, 2005)

[6] Pertenece a la clase de los algoritmos probabilísticos e iterativos. Pretende simular el proceso de enfriamiento de los metales.

En la terminología utilizada en PSO, Kennedy y Eberhart, introducen el término general partícula o agente para representar a los peces, pájaros, abejas, hormigas o cualquier otro tipo de individuos que exhiban un comportamiento social como grupo, en forma de una colección de agentes que interactúan entre sí. (Pérez, 2005)

De acuerdo con los fundamentos teóricos del método, el movimiento de cada una de estas partículas hacia un objetivo común en dos dimensiones está condicionado por dos factores básicos, la memoria autobiográfica de la partícula o nostalgia y la influencia social de todo el enjambre. (Pérez, 2005)

A nivel computacional, como método de optimización, esta filosofía puede extenderse a un espacio N-dimensional de acuerdo con el problema bajo análisis. La posición instantánea de cada una de las partículas de la población en el espacio N-dimensional representa una solución potencial, siendo N el número de incógnitas del problema original. (Pérez, 2005)

Como un ejemplo de comportamiento de manada de PSO se puede ver a continuación que se describe el comportamiento que exhibe un enjambre de abejas en su movimiento sobre un campo cubierto con diferentes concentraciones de flores.

Sin ningún conocimiento a priori del espacio de búsqueda, las abejas inician su movimiento desde posiciones aleatorias y con velocidades aleatorias. En su desplazamiento, el objetivo del enjambre se centra en encontrar el emplazamiento con la mayor densidad de flores. Cada abeja tiene memoria y puede recordar la posición visitada con mayor densidad de flores y también conoce, por mecanismos de comunicación con sus congéneres, la localización donde otras abejas encontraron una densidad de flores significativa. Esta dupla de información es utilizada por la abeja para modificar continuamente su trayectoria, acelerando en ambas direcciones y volando hacia un punto espacial intermedio que dependerá de su posición actual y de cómo influyan sobre su decisión las así denominadas nostalgia o memoria y cooperación o conocimiento social. De esta forma, las abejas se encuentran permanentemente sobrevolando el campo en busca de posiciones con mayor densidad de flores, redirigiendo en parte la trayectoria del enjambre cada vez que se encuentran configuraciones de mayor calidad. Con el transcurso del tiempo, una vez ha sido explorado el espacio de soluciones en su totalidad, el conjunto del enjambre se encontrará volando alrededor de la zona con la mayor concentración de flores de todo el campo. En esta situación, las abejas, incapaces de encontrar posiciones alternativas mejores, son permanentemente atraídas hacia dicha posición. (Pérez, 2005)

Aplicaciones

PSO ha demostrado ser eficiente en problemas multidimensionales continuos, en problemas de investigación de operaciones tales como: identificación de procesos, proyectos de controladores, localización de bancos de capacitares (Toro, et al., 2006). Otras aplicaciones reportadas son optimización de funciones numéricas, entrenamiento de redes neuronales, aprendizaje de sistemas difusos, registrado de imágenes, viajante de comercio, control de sistemas, ingeniería Química (Optimización Basada en Nubes de Partículas (Particle Swarm), 2006-2007)

Herramientas desarrolladas utilizando PSO (GONZÁLEZ)

- PSO TOOLBOX: Archivos Matlab (.m) que implementan el algoritmo PSO para tareas de optimización.
- JSwarm-PSO Paquete de Java para PSO.

1.1.3.3. Optimización basada en Búsqueda por Difusión Estocástica

La búsqueda por difusión estocástica fue descrita por primera vez en 1989 como un algoritmo de "pattern matching"[7]basado en una población (Stochastic Searching Networks, 1989). A diferencia de la comunicación empleada en la Colonia de Hormigas, la cual se basa en la modificación de las propiedades físicas de un entorno simulado, la búsqueda por difusión estocástica utiliza una forma de comunicación directa (uno a uno) entre agentes. (Inteligencia Colectiva y sus Aplicaciones, 2007) SDS[8] puede definirse como un nuevo método de búsqueda genérico o metaheurística. (Meyer, et al.)

Los agentes ejecutan evaluaciones poco costosas de una hipótesis (candidata a solución del problema de búsqueda). Entonces comparten la información acerca de las hipótesis (difusión de la información) a través de una comunicación directa uno a uno. Como resultado del mecanismo de difusión, pueden identificarse soluciones de alta calidad desde clústeres de agentes con la misma hipótesis. La manera en que opera este algoritmo puede comprenderse a través de una simple analogía con el ejemplo que se describe a continuación: el juego del restaurante (Inteligencia Colectiva y sus Aplicaciones, 2007).

[7] Conjugación de patrones.

[8] Stochastic Diffusion Search ó Búsqueda por Difusión Estocástica.

Juego del restaurante

Un grupo de delegados participa en una extensa conferencia en un pueblo desconocido. Cada noche ellos deben encontrar donde comer. Hay una larga lista de restaurantes de donde seleccionar y cada uno de ellos ofrece una gran variedad de platos. El problema que enfrenta el grupo es seleccionar el mejor restaurante, aquel donde el máximo número de delegados disfrute la comida. Puesto que una búsqueda exhaustiva entre restaurantes y platos pudiera tomar mucho tiempo, los delegados deciden emplear búsqueda por difusión estocástica. (Inteligencia Colectiva y sus Aplicaciones, 2007)

Cada delegado actúa como un agente tomando como hipótesis un supuesto mejor restaurante del pueblo. Cada noche, cada delegado prueba su hipótesis yendo a comer a ese restaurante y seleccionando de manera aleatoria un plato de la oferta. En la mañana siguiente, cada delegado que no disfrutó su comida la noche anterior le pregunta a un colega seleccionado de manera aleatoria sobre sus impresiones acerca de su comida. Si la experiencia del colega fue buena, él adopta el restaurante del colega como su elección. En otro caso, él simplemente selecciona otro restaurante al azar. Utilizando esta estrategia, rápidamente un número significativo de delegados se congrega en el "mejor" restaurante del pueblo. (Inteligencia Colectiva y sus Aplicaciones, 2007)

Aplicaciones

La búsqueda por difusión estocástica ha sido aplicada a diversos problemas tales como búsqueda de textos (Stochastic Searching Networks, 1989), reconocimiento de objetos (Bishop, et al., 1992), rastreo de rasgos (Locating the Mouth Region in Images of Human Faces., 1993), selección de sitios para redes inalámbricas (Whitaker, et al.), estimación de parámetros robustos y la Programación Lógica Inductiva. Además se puede utilizar en problemas de planificación de trayectorias, en los que la meta es construir una trayectoria para alcanzar un blanco especificado (Meyer, et al.).

Ha sido aplicado a una variedad de problemas del mundo real: localizar los ojos en imágenes de caras humanas (Bishop, et al., 1992); rastrear labios en películas de video (Grech-Cini, 1995) y auto-localización de una silla de ruedas autónoma (Self-localisation in the senario autonomous wheel-chair., 1998). Además, fue propuesto un modelo de red neural de SDS usando neuronas hablantes (Nasuto, et al., 1998); (Nasuto, et al., 1999). La sincronización emergente a través de una población grande de neuronas en esta red puede interpretarse como un mecanismo de amplificación atencional (De Meyer, et al., 2000). El análisis de SDS incluye la caracterización de su asignación estable de recursos del

lugar (Nasuto, 1999), la convergencia probada globalmente en la solución óptima (Nasuto, et al., 1999) y la complejidad de tiempo linear (Nasuto, et al., 1998).

1.2. Cronograma de Proyectos

A continuación se describe qué es la Optimización de Cronograma de Proyectos y las herramientas que se utilizan con ese fin.

1.2.1. Optimización de Cronograma de Proyectos

1.2.1.1. Proyecto

Un proyecto se puede definir como el evento o conjunto de eventos con los mismos objetivos con una duración predeterminada que requiere inversión financiera y actividad humana que está compuesto por diferentes etapas (Parodi, 2001):

- Planificación: Etapa de un proyecto en la que se valoran las opciones, tácticas y estrategias a seguir teniendo como indicador principal el objetivo a lograr.
- Ejecución: Etapa de acción, en la que ocurre propiamente el proyecto.
- Evaluación: Etapa final de un proyecto en la que éste es revisado, y se llevan a cabo las valoraciones pertinentes sobre lo planeado y lo ejecutado, así como sus resultados, en consideración al logro de los objetivos planteados.

1.2.1.2. Cronograma

Un cronograma es una secuencia detallada y cronológica de las actividades que se van a ejecutar para alcanzar los resultados propuestos.

Un cronograma de proyecto consiste en una lista de todos los elementos terminales de un proyecto con sus fechas previstas de comienzo y final.

1.2.1.3. Optimización

Un problema de optimización trata de tomar una decisión óptima para maximizar (ganancias, velocidad, eficiencia, etc.) o minimizar (costos, tiempo, riesgo, error, etc.) un criterio determinado. (Optimización (matemática))

De manera general se puede deducir que la optimización de un cronograma de proyecto consiste en minimizar el tiempo y los costos de un proyecto, aunque a los efectos de este trabajo, solo se estará tratando la temática de minimizar el tiempo de ejecución del proyecto.

1.2.2. Herramientas para gestionar Cronogramas de Proyectos

1.2.2.1. Herramientas que se utilizan en el Mundo

En la actualidad en el mundo se utilizan diferentes herramientas para la elaboración de cronogramas de proyectos. La más utilizada es el Microsoft Office Project 2003. Como resultado de la investigación realizada en (Thomas, 2007) se obtienen las siguientes:

Microsoft Office Project 2003: Es la solución que forma parte del paquete Microsoft Office para la Administración de Proyectos. Es una herramienta esencial para todo gerente de proyectos. Con Project 2003 se podrá programar y organizar los recursos y las tareas, a fin de generar proyectos a tiempo y conforme al presupuesto. Adicionalmente, se contará con la Guía Project que ayuda a dominar rápidamente el proceso, y a programarlo paso a paso, a controlar su progreso y reportar información del mismo.

La familia Microsoft Office Project 2003 está formada por dos líneas de producto: Microsoft Project Standard 2003, y Enterprise Project Management (EPM), que es una solución empresarial compuesta por Microsoft Project Server 2003, Microsoft Project Professional 2003 y Microsoft Office Project Web Access.

Microsoft Project Standard 2003: Es una versión del programa central de administración de proyectos de Microsoft. Con herramientas conocidas sencillas de usar, Project Standard permite administrar proyectos de forma independiente en el escritorio. Con Project Standard, puede planear, administrar y comunicar información de los proyectos con más eficacia que antes.

Enterprise Project Management (EPM): Es la solución idónea para las organizaciones que necesiten un alto nivel de coordinación y estandarización entre los proyectos y los administradores de proyectos, administración centralizada de los recursos o un mayor nivel de creación de informes acerca de los proyectos y los recursos. La solución EPM de Microsoft permite que organizaciones enteras, departamentos o equipos trabajen juntos en la administración eficaz de proyectos y procesos.

Microsoft Office Project Professional 2003: Project Professional 2003 es el programa de escritorio de Microsoft para la administración de proyectos empresariales. Project Professional proporciona todas las herramientas centrales de programación de Project Standard 2003, además de eficaces capacidades de administrar recursos si se conecta al Project Server 2003.

Microsoft Office Project Server 2003: Proporciona un control centralizado de todos los proyectos desarrollados por la organización. A través de un repositorio gestionado centralmente en el servidor, permite a los equipos un acceso a la información mediante una interfaz Web. Permitiendo mantener la información convenientemente actualizada.

Microsoft Office Project Web Access: Es el portal Web que permite a los usuarios conectarse al proyecto y a la información de recursos en Project Server. Mediante el acceso Web cualquier miembro del equipo puede fácilmente consultar, analizar o modificar la información sin la necesidad de tener instalado Microsoft Project en su ordenador. Con solamente un navegador, un usuario puede acceder a las utilidades necesarias para gestionar toda la información del proyecto.

Visual Studio 2005 Team System: Descubre cómo realizar la gestión de proyectos directamente desde el entorno integrado de desarrollo, gracias a la última versión de Visual Studio. Brinda una serie de herramientas para la gestión de proyectos, basadas en los softwares ya conocidos: Microsoft Excel, Microsoft Project, Microsoft Word, y Windows Share Point Services. Mediante la integración de Microsoft Office, la gestión del proyecto no necesita tratar los datos de estas aplicaciones para los datos usados por el equipo de desarrollo. La carpeta integra los productos de trabajos en el Visual Studio IDE para el acceso eficaz del equipo.

Microsoft Solutions Framework (MSF): Está diseñado para proporcionar una guía básica sobre cómo desarrollar con éxito un proyecto software, es una flexible e interrelacionada serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos. MSF se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas. Originalmente creado en 1994 para conseguir resolver los problemas a los que se enfrentaban las empresas en sus respectivos proyectos; se ha convertido posteriormente en un modelo práctico que facilita el éxito de los proyectos tecnológico.

Existe una tendencia al Sistema Operativo Linux. El objetivo principal de este sistema es propulsar el software de libre distribución junto con su código fuente para que pueda ser modificado por cualquier

persona, dando rienda suelta a la creatividad. A continuación se relacionan algunas herramientas de uso libre para la gestión de proyectos:

Gantt PV: Es un programa gratuito, de apariencia sencilla y sin grandes complicaciones, para planificación de proyectos, descomposición, representación y seguimiento de tareas sobre diagrama de Gantt.

Gantt Project: Es una herramienta de planificación libre y fácil de usar. Basado en las tareas del Project y en el Diagrama de Gantt. Dentro de sus mejores características incluye:

- Jerarquía de tareas y dependencia.
- Diagrama de Gantt.
- Carga diagrama de Recursos.
- Diagramas generados de PERT.
- Informes HTML y PDF.
- Importa/Exporta a MS Project.

Es una aplicación de escritorio con interfaz similar a MS. Project permite programar y organizar las tareas y asignación de personas y recursos sobre una representación Gantt. Por supuesto es una herramienta mucho más ligera que MS Project, pero esto en el ámbito y dimensión de muchos proyectos es más una ventaja que un inconveniente.

Dotproject: Algo más veterana, ésta solución en entorno Web, ofrece un marco completo para la planificación, gestión y seguimiento de múltiples proyectos para clientes diferentes, quienes pueden disponer también de acceso para monitorizar la evolución del desarrollo.

TeamWork: Impresionante es la apariencia de esta herramienta de entorno Web para registrar y gestionar los tiempos de diferentes equipos de trabajo en sus respectivos proyectos. Realiza la gestión completa de informes de tiempos y costes. Combina gestión de documentos, de equipos y de proyectos.

Planner: Aplicación de escritorio para gestión y seguimiento de proyectos, con descomposición en tareas y sub-tareas, dependencias, identificación de la ruta crítica, diagramas de Gantt. Inicialmente desarrollada para Linux, dispone de versión (beta) para Windows.

HojaExcelparaScrum: Hoja de cálculo para gestionar el trabajo en cada tarea, asignación, estado y tiempo. Genera de forma automática los gráficos para el seguimiento de esfuerzo y tareas.

AgileTrack: Herramienta para planificación y seguimiento de proyectos, de interfaz sencillo. Para desarrollo de software en equipos reducidos con metodologías ágiles, especialmente extreme Programming.

PPTS: Project Planning and Tracking System es una herramienta de gestión ágil de proyectos para equipos que trabajan con Scrum y/o Extreme Programming. Es un sistema Web, accesible con un navegador que puede instalarse sobre servidor Linux o Windows (con php y MySQL) y de uso libre, con licencia GNU (Licencia Pública General en español).

XPWeb: Plataforma Web para gestión de proyectos con Extreme Programming.

TUTOS: (The Ultimate Team Organization Software) es una herramienta Web de código abierto y uso gratuito para la gestión de pequeños grupos de trabajo o departamentos que incluye las siguientes funcionalidades: calendario, gestión de equipos, directorio de personas, gestión de incidencias, registros de tiempo, listas de seguimiento.

Solodox: Servicio de software que permite editar y compartir con el equipo y demás interesados planificaciones Gantt. Su versión alfa solo funciona sobre Explorer. (Para diagrama de flujos y presentaciones 2007)

ToDoList: Es una herramienta gratuita muy simple y efectiva para la gestión de proyectos en entornos ágiles. Escasamente ocupa 1 Megabyte, y al instalarla se puede indicar que emplee un fichero.ini para guardar la información de configuración, de forma que no toca para nada el registro de Windows y se puede llevar incluso en una memoria USB.

ClockingIT: Es un gestor de proyectos y tareas, con control de tiempos, generador de informes, repositorio de ficheros, agenda, chat, y notificaciones.

1.2.2.2. Herramientas que se utilizan en la Universidad

En la Universidad de las Ciencias Informáticas, la herramienta más utilizada para elaborar los cronogramas de proyectos es Microsoft Office Project. Esta herramienta es la que se imparte en la asignatura de Ingeniería del Software. Forma parte del paquete Microsoft Office. Posibilita la

programación y organización de los recursos y las tareas a tiempo, conforme al presupuesto. Se basa fundamentalmente en el Método de la Ruta Critica y la Técnica de Revisión y Evaluación de Proyectos utilizando el Diagrama de GANTT, conjugando las tres técnicas para trabajar (Thomas, 2007).

Conclusiones

Los sistemas de Inteligencia Colectiva que forman parte de la Inteligencia Artificial se utilizan en el mundo para solucionar diferentes problemas, aunque son técnicas relativamente jóvenes cada día son más sus aplicaciones debido a que se han obtenido logros significativos. En este capítulo se describieron brevemente los algoritmos de Inteligencia Colectiva: Optimización basada en Colonia de Hormigas, Optimización basada en Nube de Partículas, Optimización basada en Búsqueda por Difusión Estocástica. En el siguiente capítulo se profundizará en dichos algoritmos.

El cronograma de proyecto constituye una parte importante en la planificación de proyectos. Para su gestión se utilizan en el mundo diferentes herramientas, siendo la más utilizada, el Microsoft Office Project, que es además el que se explota en nuestra Universidad.

CAPÍTULO 2: ANÁLISIS DE ALGUNOS ALGORITMOS DE INTELIGENCIA COLECTIVA

Introducción

En este capítulo se profundizará en los algoritmos de Inteligencia Colectiva tratados en el capítulo anterior. Se describirán sus características, algoritmos en seudocódigos. Además se seleccionará uno de ellos para su posterior implementación.

2.1. Optimización basada en Colonia de Hormigas

Los algoritmos de ACO se basan en el comportamiento de las colonias reales de hormigas para solucionar problemas de optimización combinatorias.

Trabajan en una colonia de hormigas artificiales, compuesta por agentes computacionales simples que trabajan de manera cooperativa y se comunican mediante rastros de feromonas. Estos algoritmos se pueden clasificar en algoritmos constructivos ya que en todas las iteraciones, cada hormiga construye una solución al problema, recorriendo un grafo de construcción. Cada arista del grafo, representa los posibles caminos que puede recorrer la hormiga, así como dos tipos de información que rigen su movimiento (Alonso, et al.):

Información heurística, que mide la preferencia heurística de moverse desde el nodo r hasta el nodo s, o sea, de recorrer la arista a_{rs}. Se nota por η_{rs}. Las hormigas no modifican esta información durante la ejecución del algoritmo.

Información de los rastros de feromonas artificiales, que mide la deseabilidad aprendida del movimiento de r a s. Imita la feromona real. Se denota por τ_{rs}.En las diferentes variantes de algoritmos ACO hacen un uso diferente de esta información pues es en ella que se basa el camino a soluciones satisfactorias.

2.1.1. Modo de funcionamiento y Estructura Genérica de un Algoritmo de ACO

El modo de operación básico de un algoritmo de ACO es como sigue (Alonso, et al.):

Las m hormigas artificiales de la colonia se mueven, concurrentemente y de manera asíncronas, a través de los estados adyacentes del problema los cuales pueden ser representados a través de un grafo con pesos. Este movimiento se hace siguiendo una regla de transición basada en la información local disponible en los componentes (nodos).Esta información local incluye la información heurística y

memorística (rastros de feromona) para guiar las búsqueda. Al moverse por el grafo de construcción, las hormigas construyen incrementalmente soluciones.

Las Hormigas mientras construyen la solución (actualización en línea paso a paso de los rastros de feromona), pueden de forma opcional depositar feromonas cada vez que crucen un arco (conexión).

Una vez que cada hormiga ha generado una solución se evalúa esta y puede depositar una cantidad de feromona que es en 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.

Además, el modo de operación genérico de un algoritmo de ACO incluye dos procedimientos adicionales, la evaporación de los rastros de feromonas y las acciones del demonio. 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 las hormigas busquen y exploren nuevas regiones del espacio. Las acciones del demonio son acciones opcionales para implementar tareas desde una perspectiva global que no pueden llevar a cabo las hormigas por la perspectiva local que ofrecen. Ejemplos: observar la calidad de todas las soluciones generadas y depositar una nueva cantidad de feromonas adicional sólo en las transiciones componentes asociadas a algunas soluciones consideradas como buenas, o aplicar un procedimiento de búsqueda local a las soluciones generadas por las hormigas antes de actualizar los rastros de feromonas. En ambos casos, el demonio reemplaza la actualización en línea a posteriori de feromona y el proceso pasa a llamarse actualización fuera de línea de rastros de feromona.

Estructura General de un ACO:

A continuación se muestra la estructura básica de un ACO (Dorigo, et al., 1999) y se tratarán detalladamente las acciones que de forma general se llevan a cabo.

Procedimiento Metaheuristica_ACO ()
    Inicializacion_de_parametros
        mientras (criterio_de_terminacion_no_satisfecho)
    Programacion_de_actividades
        Generacion_de_Hormigas_y_actividad ()
        Evaporacion_de_Feromona ()
        Acciones_del_demonio (){*opcional* }
    fin Programación_de_actividades

```
        Fin mientras
fin Procedimiento

Procedimiento Generacion_de_Hormigas_y_actividad ()
    repetir en paralelo desde k = 1 hasta m (numero_hormigas)
        Nueva_Hormiga (k)
    fin repetir en paralelo
fin Procedimiento

Procedimiento Nueva_Hormiga (id_Hormiga)
    inicializa_hormiga (id_Hormiga)
    L = actualiza_memoria_hormiga ()
    mientras (estado_actual ≠ estado_objetivo)
        P = calcular_probabilidades_de_transicion(A, L, Ω)
        siguiente_estado = aplicar_política_decision (P, Ω)
        mover_al_siguiente_estado (siguiente_estado)
        si (actualizacion_feromona_en_linea_paso_a_paso)
            depositar_feromona_en_el_arco_vistado ()
        fin si
        L = actualizar_estado_interno ()
    fin mientras
    si (actualizacion_feromona_en_linea_a_posteriori)
        para cada arco visitado
            depositar_feromona_en_el_arco_visitado ()
        fin para
    fin si
    liberar_recursos_hormiga (id_Hormiga)
fin Procedimiento
```

El primer paso incluye la inicialización de los valores de los parámetros que se tienen en consideración en el algoritmo. Entre otros, se debe 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 constructor Programacion_de_actividades, la planificación de las tres componentes siguientes:

- La generación y puesta en funcionamiento de las hormigas artificiales.
- La evaporación de feromona.
- Las acciones del demonio.

La implementación de este constructor determinará la sincronía existente entre cada una de estas tres componentes. Mientras que la aplicación a problemas clásicos NP-Duros (no distribuidos), normalmente usa una planificación secuencial, en problemas distribuidos como el enrutamiento en redes, el paralelismo puede ser explotado de manera sencilla y eficiente.

Varias componentes son, o bien opcionales, como las acciones del demonio, o bien dependientes estrictamente del algoritmo ACO específico, por ejemplo cuando y como 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, el demonio suele actualizar los rastros de feromona).

El procedimiento actualiza_memoria_hormiga () es el encargado 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 cual 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.).

Los procedimientos calcular_probabilidades_de_transicion y aplicar_política_decision tienen en consideración el estado actual de la hormiga, los valores actuales de la feromona visible en dicho nodo y las restricciones del problema Ω para establecer el proceso de transición probabilística hacia otros estados válidos.

2.1.2. Modelos de Optimización basada en Colonia de Hormigas

Existen diversos algoritmos que siguen la metaheurística ACO. Entre los disponibles para problemas de optimización combinatoria NP-duros, se encuentran:

- *Sistema de Hormigas* (SH o Ant System (AS)).
- *Sistema de Colonia de Hormigas* (SCH o Ant Colony System (ACS)).
- *Sistema de Hormigas* Max-Min (SHMM, Max-Min Ant System(MMAS)).
- *SH con ordenación* (Rank-Based Ant System).
- *Sistema de la Mejor-Peor Hormiga* (SMPH o Best-Worst Ant System(BWAS)).

2.1.2.1. El Sistema de Hormigas

El sistema de Hormigas fue el primer algoritmo desarrollado por Dorigo, Maniezzo y Colorni en 1991. Inicialmente se presentaron tres variantes diferentes, AS-densidad, AS-cantidad y AS-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 (aplicaban una actualización en línea paso a paso de feromona), con la diferencia de que la cantidad de feromona depositada en AS-densidad es constante, mientras que la depositada en AS-cantidad dependía directamente de la deseabilidad heurística de la transición η_{rs}. 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 es la que se conoce como AS en la literatura ya que era la que obtenía mejores resultados. (Alonso, et al.)

El AS se caracteriza por el hecho de que la actualización de la feromona se realiza una vez que todas las hormigas han completado sus soluciones y se lleva a cabo como sigue: primero, todos los rastros de feromona se reducen en un factor constante, implementándose de esta manera la evaporación de la feromona. Seguidamente cada hormiga de la colonia deposita una cantidad de feromona que es función de la calidad de su solución. En sus inicios el AS 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 (Alonso, et al.). En cada paso de construcción, una hormiga k escoge ir al siguiente nodo con una probabilidad que se calcula:

$$p_{rs}^{k} = \begin{cases} \dfrac{[\tau_{rs}]^{\alpha} \cdot [\eta_{rs}]^{\beta}}{\sum_{u \in N_r^k} [\tau_{rs}]^{\alpha} \cdot [\eta_{rs}]^{\beta}}, & si\ s \in N_k(r) \\ 0, & en\ otro\ caso \end{cases} \qquad (1)$$

Donde $N_k(r)$ es el vecindario alcanzable por la hormiga k cuando se encuentra en el nodo r, y $\alpha, \beta \in \mathcal{R}$ son dos parámetros que ponderan la importancia relativa de los rastros de feromonas y la información heurística. Cada hormiga k almacena la secuencia que ha seguido hasta el momento y su memoria L_k , tal como se explicó antes, se utiliza para determinar $N_k(r)$ en cada paso de construcción.

En la Figura 2.1 se puede observar la manera en que ocurre:

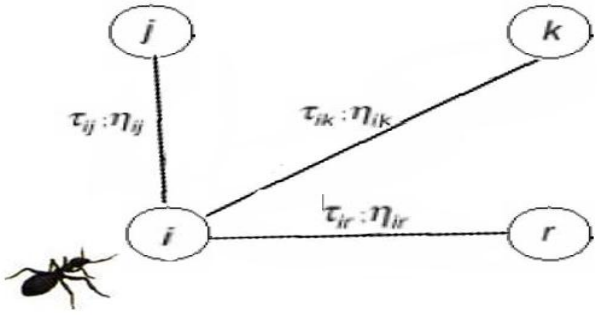


Figura 2. 1 Probabilidad que usan las hormigas para escoger caminos.

La función de los parámetros α y β:

- 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 partidas en caso de que las hormigas estén situadas en nodos distintos al comienzo de cada iteración).

- Si $\beta=0$, solo se tiene 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 superior 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 mencionado anteriormente, la deposición de la 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:

$$\tau_{rs} \leftarrow (1-\rho)\cdot\tau_{rs} \qquad (2)$$

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 está almacenado en su memoria L_k) y deposita una cantidad de feromona $\Delta\tau_{rs}^{k}$ en cada conexión por la que ha viajado:

$$\tau_{rs} \leftarrow \tau_{rs} + \Delta\tau_{rs}^{k}\,,\ \forall a_{rs} \in S_k \qquad (3)$$

donde $\Delta\tau_{rs}^{k} = f\big(C(S_k)\big)$, es decir, la cantidad de feromona que se deposita depende de la calidad $C(S_k)$ de la solución S_k construida por la hormiga k.

A continuación puede observarse el procedimiento Nueva_Hormiga para este algoritmo de ACO:

Procedimiento Nueva_Hormiga (id_Hormiga)
k= id_Hormiga; r = generar_estado_inicial; $S_k = r$
$L_k = r$
mientras (estado_actual ≠ estado_objetivo)
Para cada $s \in N_k(r)$ hacer $p_{rs}^{k} = \frac{[\tau_{rs}]^{\alpha}\cdot[\eta_{rs}]^{\beta}}{\sum_{u\in N_r^k}[\tau_{rs}]^{\alpha}\cdot[\eta_{rs}]^{\beta}}$
siguiente_est =aplicar_politica_de_decision $(P, N_k(r))$
r =siguiente_est; S_k =<S_k, r>

$L_k = L_k \cup r$
fin *mientras*
{Se ejecuta el procedimiento Evaporacion_de_feromona() se lanza y evapora la feromona se lanza y evapora la feromona en cada arco a_{rs}: $\tau_{rs} = \{1-\rho)\cdot\tau_{rs}\}$
Para cada arco $a_{rs} \in S_k$ hacer
$\tau_{rs} = \tau_{rs} + f\big(c(S_k)\big)$
fin *para*

Liberar_recursos_hormiga (id_Hormiga)
fin Procedimiento

Los creadores de AS propusieron una versión extendida de este algoritmo que mejoraba los resultados obtenidos (solucionando el problema de la lentitud de convergencia), llamada AS elitista (Dorigo, et al., 1996). 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 las aristas que pertenecen a la mejor solución encontrada hasta el momento en el proceso de búsqueda. La cantidad de feromona depositada, que depende de la cantidad de la mejor solución global, se incrementa en un factor e, que se corresponde con el número de hormigas elitistas, como se muestra:

$$\tau_{rs} = \tau_{rs} + e.f\left(C\left(S_{mejor-global}\right)\right),\ \forall a_{rs} \in S_{mejor-global} \qquad (4)$$

2.1.2.2. Sistema de Colonia de Hormiga

El ACS es uno de los primeros sucesores del AS que introduce tres modificaciones importantes con respecto a dicho algoritmo de ACO (Alonso, et al.):

1. El ACS usa una regla de transición distinta, denominada regla proporcional seudoaleatoria. Sea k una hormiga situada en el nodo $r, q_0 \in [0,1]$ un parámetro y q un valor aleatorio en $[0,1]$, el siguiente nodo s se elige aleatoriamente mediante la siguiente distribución de probabilidad:

   Si $q \leq q_0$

$$p_{rs}^{k} = \begin{cases} 1, & si\ s = arg \max_{u \in N_k(r)} \left\{\tau_{ru} . \eta_{rs}^{\beta}\right\} \\ 0, & en\ otro\ caso \end{cases} \qquad (5)$$

   Si no ($q > q_0$):

$$p_{rs}^{k} = \begin{cases} \frac{[\tau_{rs}]^{\alpha} \cdot [\eta_{rs}]^{\beta}}{\sum_{u \in N_r^k} [\tau_{rs}]^{\alpha} \cdot [\eta_{rs}]^{\beta}}, & si\ s \in N_k(r) \\ 0, & en\ otro\ caso \end{cases} \qquad (6)$$

Como se observa, la regla tiene dos intensiones si $q \leq q_0$, explota el conocimiento disponible, eligiendo la mejor opción con respecto a la información heurística y los rastros de feromona. Por otro lado si $q > q_0$ se aplica una exploración controlada, de la misma forma que se hacía en el AS. En fin esta regla establece un compromiso entre la exploración de nuevas conexiones y la explotación de la información disponible en ese momento.

2. Solo el demonio (y no las hormigas individualmente) actualizan la feromona, en otras palabras, se realiza una actualización de feromona fuera de línea de los rastros. Para realizar este paso, el ACS solo considera una hormiga concreta, la que generó la mejor solución global, $S_{mejor-global}$ (aunque en algunos trabajos iniciales se consideraba también una actualización basada en la mejor hormiga de la iteración, en ACO casi siempre se aplica la actualización por medio de la mejor global).

La actualización de la feromona se lleva acabo evaporando primeramente los rastros de feromona en todas las conexiones utilizadas por la mejor hormiga global (es importante recalcar que, en el ACS, la evaporación de la feromona solo se aplica a las conexiones de la solución, que es también la usada para depositar feromona) como se muestra a continuación:

$$\tau_{rs} \leftarrow (1-\rho) \cdot \tau_{rs},\ \forall a_{rs} \in S_{mejor-global} \qquad (7)$$

A continuación, el demonio deposita feromona usando la regla:

$$\tau_{rs} \leftarrow \tau_{rs} + \rho \cdot f\left(C\left(S_{mejor-global}\right)\right),\ \forall a_{rs} \in S_{mejor-global} \qquad (8)$$

Adicionalmente, el demonio puede aplicar un algoritmo de búsqueda local para mejorar las soluciones de las hormigas antes de actualizar los rastros de feromonas.

3. Las Hormigas aplican una actualización en línea paso a paso de los rastros de feromona que favorece la generación de soluciones distintas a las ya encontradas.

Cada vez que una hormiga viaja por una arista a_{rs} , aplica la regla:

$$\tau_{rs} \leftarrow (1-\varphi).\tau_{rs} + \varphi.\tau_0 \qquad (9)$$

donde $\varphi \in (0,1]$ es un segundo parámetro de decremento de feromona. Como puede verse, la regla de actualización en línea paso a paso incluye tanto la evaporación de feromona como la

deposición de la misma. Ya que la cantidad de feromona depositada es muy pequeña (de hecho, τ_0 es el valor del rastro de feromona inicial y se escogiese de tal manera que, en la práctica, se corresponda con el límite menor de rastro de feromona, esto es, con la elección de las reglas de actualización de feromona del ACS ningún rastro de feromona puede caer por debajo de τ_0), la aplicación de esta regla hace que los rastros de feromona entre las conexiones recorridas por las hormigas disminuyan. Así, esto lleva a una técnica de exploración adicional del ACS ya que las conexiones atravesadas por un gran número de hormigas son cada vez menos atractivas para el resto de las hormigas que la recorren en la iteración actual, lo que ayuda claramente a que no todas las hormigas sigan el mismo camino.

A continuación se muestra los procedimientos Nueva _ hormiga y Acciones_del_demonio para el ACS:

Procedimiento Nueva_Hormiga (id_hormiga)

k= id_hormiga; r=Generar_estado_Inicial; $S_k = r$

$L_k = r$

mientras (estado_actual≠ estado_objeto)

para cada $s \in N_k(r)$ hacer calcular $b_{rs} = \tau_{rs}.\eta_{rs}^{\beta}$

q=generar_valor_aleatorio_en_ $[0,1]$

si $q \leq q_0$

siguiente_estado= $\max(b_{rs}, N_k{}^{(r)})$

si no

para cada $s \in N_k(r)$ hacer

$$p_{rs}^{k} = \frac{b_{rs}}{\sum_{u \in N_k(r)} b_{ru}}$$

sig_est =aplicar_politica_de_decision $(P, N_k(r))$

fin si

r= sig_est ; S_k=< S_k, r>

$\tau_{rs} \leftarrow (1-\varphi).\tau_{rs} + \varphi.\tau_0$

$L_k = L_k \cup r$

fin mientras

liberar_recursos_hormiga (id_hormiga)

fin Procedimiento

Procedimiento Acciones_del_demonio

Para cada S_k hacer busqueda_local (S_k) {opcional}

$S_{mejor-actual}$ =mejor_solucion (S_k)

Si (mejor $(S_{mejor-actual}, S_{mejor-global})$)

$S_{mejor-global} = S_{mejor-actual}$

fin si

para cada arista $a_{rs} \in S_{mejor-global}$ hacer

{se ejecuta el procedimiento evaporacion_de_feromona () se lanza y evapora feromona en la arista a_{rs}: $\tau_{rs} = (1-\rho) \cdot \tau_{rs}$}

$$\tau_{rs} = \tau_{rs} + \rho . f\left(C\left(S_{mejor-global}\right)\right)$$

fin para

fin Procedimiento

Sistema de Colonia de Hormigas aplicado al problema TSP

La técnica del Sistema de Colonia de Hormigas tiene gran aplicación en la solución del clásico Problema del Viajero Vendedor, en el que un viajero quiere visitar un conjunto de n ciudades exactamente una vez y regresar a su punto de origen de forma tal que se minimice la distancia total recorrida. Este problema puede solucionarse encontrando un circuito Hamiltoniano con distancia total mínima, pero no existe un algoritmo con complejidad temporal polinomial que solucione el mismo. Para explicar el algoritmo consideremos un conjunto de n ciudades que serán visitadas por m hormigas. (Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, 1997)

Para satisfacer la restricción de que una hormiga visite todas las ciudades una sola vez, se asocia a cada hormiga k una pequeña estructura de datos llamada lista tabú, $tabu_k$, que guarda información relativa a las ciudades ya visitadas por dicha hormiga. Una vez que todas las ciudades hayan sido recorridas, el trayecto o tour es completado volviendo a la ciudad origen. La lista tabú se vacía y nuevamente la hormiga está libre para iniciar un nuevo tour, independientemente del estado en que se encuentren las demás hormigas del sistema, lo que sugiere un alto grado de paralelismo asíncrono. En este contexto, se define como $tabu_k(n)$ al elemento n-ésimo de la lista tabú de la hormiga k y como $J_k(i)$ al conjunto de ciudades que aun no visitó la hormiga k ubicada en la ciudad i

Dado el conjunto de n ciudades, se denomina d_{ij} a la longitud del camino entre las ciudades i, j; para el caso del Problema del Viajero Vendedor. El punto de partida para la solución del problema, es la matriz de distancias $D = \{d_{ij}$, distancia entre la ciudad i y la ciudad $j\}$, a partir de la cual se calcula la visibilidad:

$$\eta_{ij} = 1 / d_{ij}$$

Por su parte, se denota como $\tau = \{\tau(i,j)\}$ a la matriz de feromonas a ser utilizada para consolidar la información que va siendo recogida por las hormigas. En otras palabras, $\tau(i,j)$ representa la cantidad de feromona que se va almacenando entre cada par de ciudades $(i\ j)$. Esta es inicializada con un valor τ_0 definida como:

$$\tau_0 = (n \times L_{nn})^{-1} \qquad (1)$$

donde L_{nn} es a la longitud de un tour típico, obtenido inicialmente por alguna otra heurística o por pruebas aleatorias y que puede tratarse simplemente de una mala aproximación a la longitud óptima del tour, pues el algoritmo final no es muy sensible a la elección de este parámetro inicial (Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, 1997)

La intensidad de las feromonas del arco $(i\ j)$, denotada $\tau(i,j)$, es actualizada localmente mientras las hormigas construyen su tour, esto es, al moverse de la ciudad i a la ciudad j, cada hormiga deposita una cantidad de feromonas en el arco correspondiente, calculado conforme:

$$\tau(i,j) = (1-\rho) \times \tau(i,j) + \rho \times \tau_0 \qquad (2)$$

donde $0 < \rho < 1$ es un parámetro que puede entenderse como de evaporación de las feromonas de τ (pudiera tomarse $\rho = 0.1$). Además, se procede a una actualización global de τ según (3), la cual se realiza cuando todas las hormigas de una colonia terminaron su tour y se puede establecer la mejor solución del ciclo.

$$\tau(i,j) = (1-\alpha) \times \tau(i,j) + \alpha \times \Delta\tau(i,j) \qquad (3)$$

donde α es el coeficiente de evaporación de las feromonas, que determina el grado de influencia de una buena solución en la actualización de la matriz de feromonas, mientras que la cantidad de feromona depositada en un arco $(i\ j)$, está dada por:

$$\Delta\tau(i,j) = \begin{cases} (L_{gb})^{-1} & si\ (i,j) \in al\ mejor\ tour\ global \\ 0 & de\ otra\ manera \end{cases} \qquad (4)$$

donde L_{gb} es la longitud del mejor tour global, hallado desde el mismo inicio de la corrida.

Durante la ejecución del algoritmo Sistema de Colonia de Hormigas, una hormiga ubicada en la ciudad i debe elegir la próxima ciudad j a visitar, para lo cual elige de entre las ciudades no visitadas, con una probabilidad q_0 la ciudad con mayor cantidad de feromonas en τ (explotación), o en forma aleatoria conforme se explica a continuación (exploración). Esto puede ser expresado como:

$$j = \begin{cases} \arg max_{u \in J_k(i)}\{[\tau(i,u)] \times [\eta(i,u)]^{\beta}\}, & si\ q \leq q_0\ (explotación) \\ R, & de\ otra\ manera\ (basado\ en\ exploración) \end{cases} \qquad (5)$$

donde q es un número aleatorio uniformemente distribuido entre $[0 \ldots 1]$ obtenido al momento de la decisión, q_0 es un parámetro $0 < q_0 < 1$ que representa una probabilidad y R es una variable aleatoria seleccionada de acuerdo con la probabilidad dada por la ecuación (6).

$$p_{rs}^{k} = \begin{cases} \dfrac{[\tau(i,j)\,] \times [\eta(i,j)\,]^{\beta}}{\sum_{u \in J_k(i)} [\tau\,(i,u)] \times [\eta(i,u)]^{\beta}}, & si\ j \in J_k(i) \\ 0, & de\ otra\ manera \end{cases} \qquad (6)$$

donde β es un parámetro que determina la importancia relativa de las feromonas respecto a la distancia ($\beta > 0$). Una vez analizados estos elementos, el algoritmo secuencial planteado en (Barán, et al., 2002) puede expresarse como:

1. Fase de inicialización
   - Inicializar contador de ciclos *NC*
   - Para cada arco (i,j):
     - valor inicial de $\tau_0(t) = \tau_0$
   - Para cada hormiga
     - Elegir ciudad origen

2. Repetir hasta llenar $tabu_k$
   - Si es la ultima posición de $tabu_k$
     - Para cada hormiga:
       - Insertar ciudad origen en $tabu_k$

Sino

Para cada hormiga:

Elegir próxima ciudad a ser visitada según ecuaciones (5) y (6)

Mover la hormiga a la próxima ciudad

Insertar ciudad seleccionada en $tabu_k$

Para cada hormiga:

Actualizar feromonas según ecuación (2)

3. Repetir para cada hormiga k

Calcular la longitud de L_k del ciclo

Actualizar feromonas según ecuación (3)

4. Si (Condicion_fin=Verdadero)

Imprimir camino más corto L_k

Sino

Ir a la Fase 2

En resumen, Colonia de Hormigas se inicia ubicando m hormigas en n ciudades de acuerdo a una regla de inicialización. Para nuestros experimentos se eligen las ciudades origen de manera aleatoria. Cada hormiga construye su propio tour, eligiendo la próxima ciudad a visitar aplicando las ecuaciones (5) y (6). Mientras construyen su tour, las hormigas actualizan feromonas al moverse de la ciudad i a la ciudad j según la ecuación (2). Cuando todas las hormigas volvieron a su ciudad origen, se calcula la mejor distancia hallada en el ciclo, y nuevamente se realiza una actualización de las feromonas, esta vez teniendo en cuenta, solo la mejor solución global L_{gb} según la ecuación (3).

El proceso se repite iterativamente hasta que se cumpla algún criterio de parada. Por simplicidad, el proceso termina si el contador de ciclos alcanza un número máximo de ciclos NCMAX (definido por el usuario), pudiera ser igual a 10.000. Debido a la forma en que se calculan las probabilidades en (5) y (6), las hormigas son guiadas en la búsqueda de soluciones por información heurística, prefiriendo elegir los arcos con menor distancia y mayor cantidad de feromonas, lo que a su vez sirve para incrementar la cantidad de feromonas en los caminos que van resultando óptimos a medida que avanza la corrida del algoritmo.

2.1.2.3. El sistema de Hormiga Max-Min

El MMAS fue desarrollado por Stützle y Hoos en 1996, y constituye uno de las extensiones del AS con mejores rendimientos reportados. Los siguientes aspectos constituyen las extensiones del AS (Alonso, et al.):

1. Se aplica una actualización de los rastros de feromona fuera de línea, de manera similar a como se hace en el ACS. Después de que todas las hormigas hayan construido su solución cada rastro de feromona sufre una evaporación:

$$\tau_{rs} = (1 - \rho) \cdot \tau_{rs} \qquad (10)$$

y a continuación la feromona se deposita siguiendo las siguiente fórmula:

$$\tau_{rs} = \tau_{rs} + f\left(C\left(S_{mejor}\right)\right),\ \forall a_{rs} \in S_{mejor} \qquad (11)$$

La *mejor hormiga* a la que se le permite añadir feromona puede ser la que tiene mejor solución *mejor de la iteración* o la solución *mejor global*. Experimentos demuestran que el mejor rendimiento se obtiene incrementando la frecuencia de escoger la mejor global para la actualización de la feromona.

En el MMAS las soluciones que se obtienen a través de las hormigas son mejoradas utilizando optimizadores locales previamente a la actualización de la feromona.

2. Los valores posibles para los rastros de feromona están limitados al rango$[\tau_{min}, \tau_{max}]$.Esto disminuye la probabilidad de un estancamiento del algoritmo, al darle a cada conexión existente una pequeña probabilidad de ser escogido. En la práctica, existen heurísticas para fijar los valores τ_{min} y τ_{max} .Se puede observar como resultado de la evaporación de la feromona que el nivel máximo de feromona en los rastros está limitado a $\tau^*_{max} = 1/\left(\rho.C(S^*)\right)$, donde S^* es la solución óptima. Basándonos en este resultado, la mejor solución global puede usarse para estimar τ_{max} sustituyendo S^* en la ecuación de τ^*_{max}. Para τ_{min}, normalmente solo se escoge su valor de tal manera que sea un factor constante menos que τ_{max}.

3. En vez de inicializar los rastros de feromona a una cantidad pequeña , el MMAS los inicializa a una estimación del máximo permitido para un rastro (la estimación puede obtenerse generando una solución S' con una heurística voraz y reemplazando dicha solución S' en la ecuación de τ^*_{max}).Esto lleva a una componente adicional de diversificación en el algoritmo, ya que al comienzo las diferencias relativas entre los rastros de feromona no serán muy acusadas, lo que no ocurre cuando los rastros se inicializan a un valor muy pequeño.

La estructura del procedimiento Acciones_del_demonio- en el MMAS se muestra a continuación:

Procedimiento Acciones_del_demonio
Para cada S_k hacer busqueda_local (S_k)
$S_{mejor-actual}$ = mejor_solucion (S_k)
Si (mejor $(S_{mejor-actual}, S_{mejor-global})$)
$S_{mejor-global} = S_{mejor-actual}$
fin si
S_{mejor}= decision($S_{mejor-global}, S_{mejor-actual}$)
Para cada arista $a_{rs} \in S_{mejor}$ hacer
$\tau_{rs} = \tau_{rs} + f\left(C\left(S_{mejor}\right)\right)$
si $(\tau_{rs} < \tau_{min})$ $\tau_{rs} = \tau_{min}$
fin para
si (condicion_de_estancamiento)
para cada arista a_{rs} hacer $\tau_{rs} = \tau_{max}$
fin si
fin procedimiento

2.1.2.4. Sistema de Hormigas con Ordenación

El sistema de Hormigas con ordenación fue propuesto como otra extensión de AS en 1997 por Bullnheimer, Hartl y Strauss. Introduce la idea de ordenar las hormigas para realizar la actualización de feromona, que el demonio realiza, de nuevo, fuera de línea, tal como se muestra a continuación (Alonso, et al.):

1. Las m hormigas se ordenan de mejor a peor según la calidad de sus soluciones: $(S'_1, \ldots, S'_m)$, siendo S'_1 la mejor solución construida en la iteración actual.

2. El demonio deposita feromona en las conexiones por las que han pasado las $\sigma - 1$ mejores hormigas (hormigas elitistas). La cantidad de feromona depositada depende directamente del orden de la hormiga y de la calidad de su solución.

3. Las conexiones por las que ha pasado la mejor hormiga global reciben una cantidad adicional de feromona que depende únicamente de la cantidad de dicha solución. Esta deposición de feromona se considera la más importante, de hecho recibe un peso σ.

Esta metodología de operación tiene efecto al utilizar la siguiente regla de actualización de feromona, la que es aplicada a cada arista una vez que todos los rastros de feromona han sido evaporados:

$$\tau_{rs} \leftarrow \tau_{rs} + \sigma.\Delta\tau_{rs}^{mg} + \Delta\tau_{rs}^{orden} \qquad (12)$$

donde

$$\Delta\tau_{rs}^{mg} = \begin{cases} f\left(C\left(S_{mejor-global}\right)\right), & si\ \ a_{rs} \in S_{mejor-global} \\ 0, & en\ otro\ caso \end{cases} \qquad (13)$$

$$\Delta\tau_{rs}^{orden} = \begin{cases} \sum_{\mu=1}^{\sigma-1}(\sigma-\mu) f\left(C\left(\mathrm{S}'_{\mu}\right)\right), & si\ \ a_{rs} \in \mathrm{S}'_{\mu} \\ 0, & en\ otro\ caso \end{cases} \qquad (14)$$

Finalmente el procedimiento Acciones _del _demonio presenta la siguiente estructura:

Procedimiento Acciones_del_demonio

Para cada $S_k$hacer busqueda_local (S_k) {opcional}

ordenar $(S_1, \ldots, S_m)$ en orden decreciente según la calidad de la solución: $(S'_1, \ldots, S'_m)$

si (mejor $\left(S'_1, S_{mejor-global}\right)$)

$S_{mejor-global} = S'_1$

fin si

desde $\mu = 1$ hasta $(\sigma - 1)$ hacer

para cada arista $a_{rs} \in \mathrm{S}'_{\mu}$ hacer

$$\tau_{rs} = \tau_{rs} + (\sigma - \mu).f\left(C\left(\mathrm{S}'_{\mu}\right)\right)$$

fin para

fin desde

para cada arista $a_{rs} \in S_{mejor-global}$ hacer

$$\tau_{rs} = \tau_{rs} + \sigma . f\left(C\left(S'_{\mu mejor-global}\right)\right)$$

fin para

fin procedimiento

2.1.2.5. El Sistema de Mejor Peor- Hormiga

El BWAS es un algoritmo que fue propuesto por Cordón y otros en 1999, incorpora conceptos de computación evolutiva. Este algoritmo constituye una extensión del AS. Utiliza la misma regla de transición de estados que el AS, así como la misma regla de evaporación de feromona que en el $AS_{Ordenación}$ y el MMAS, se aplica a todas las transiciones. Al igual que el MMAS, siempre considera la explotación sistemática de optimizadores locales para mejorar las soluciones de las hormigas.

A continuación se muestran las tres acciones que se pueden encontrar del demonio (Alonso, et al.):

La *regla mejor-peor de actualización de rastros de feromona*, basada en la regla de actualización del vector de probabilidad del PBIL, refuerza las aristas que se encuentran en la mejor solución global. Además, penaliza cada conexión de la peor solución generada hasta el momento,$S_{peor-actual}$, que no se encuentre en la mejor global realizando una evaporación de feromona adicional de esos rastros. Por tanto, la regla de actualización de feromona en el BWAS se convierte en:

$$\tau_{rs} \leftarrow \tau_{rs} + \rho . f\left(C\left(S_{mejor-global}\right)\right),\ \forall a_{rs} \in S_{mejor-global} \qquad (15)$$

$$\tau_{rs} \leftarrow (1-\rho).\tau_{rs},\ \forall a_{rs} \in S_{peor-actual}\ y\ \ a_{rs} \notin S_{mejor-global} \qquad (16)$$

Se realiza una mutación de los rastros de feromona para introducir diversidad en el proceso de búsqueda. Para llevarla a cabo, el rastro de feromona asociado a cada una de las transiciones desde cada nodo (ejemplo, cada fila de la matriz de rastros de feromona) se muta con una probabilidad P_m utilizando operador de mutación con codificación real.

La propuesta original del BWAS aplicaba un operador que alteraba los rastros de feromona de cada transición mutada añadiendo o restando la misma cantidad en cada iteración. El rango de mutación $mut(it, \tau_{umbral})$ que depende de la media de los rastros de feromona en las transiciones de la mejor

solución global, τ_{umbral} , es más suave en las primeras etapas del algoritmo (donde no hay riesgo de estancamiento) y más fuerte en las últimas etapas, donde el peligro de estancamiento es más fuerte:

$$\tau'_{rs} \leftarrow \begin{cases} \tau_{rs} + mut(it, \tau_{umbral}), & si\ a = 0 \\ \tau_{rs} - mut(it, \tau_{umbral}), & si\ a = 1 \end{cases} \qquad (17)$$

Como en otros modelos de ACO, el BWAS considera la *reinicialización* de los rastros de feromona cuando se estanca la búsqueda, lo que se lleva a cabo fijando cada rastro de feromona a. τ_0 En las primeras versiones del algoritmo se comprobaba si el porcentaje de arcos distintos entre la mejor solución y la peor de la población actual era menor que un cierto valor umbral. En versiones más recientes se utiliza un concepto diferente para evaluar si el algoritmo se ha estancado o no: se comprueba el porcentaje de iteraciones del algoritmo sin haber conseguido una mejora en la mejor solución.

El procedimiento Acciones_del_demonio es como sigue:

Procedimiento Acciones_del_demonio

para cada S_k hacer busqueda_local (S_k)

$S_{mejor-actual}$ =mejor_solucion (S_k)

si(mejor ($S_{mejor-actual}, S_{mejor-global}$))

$S_{mejor-global} = S_{mejor-actual}$

fin si

para cada arista $a_{rs} \in S_{mejor-global}$ hacer

$\tau_{rs} = \tau_{rs} + \rho . f\left(C\left(S_{mejor-global}\right)\right)$

Suma=suma+τ_{rs}

fin para

$\tau_{umbral=sum/|S_{mejor-global}|}$

$S_{peor-actual}$=peor_solucion (S_k)

Para cada arista $a_{rs} \in S_{peor-global}$ y hacer $a_{rs} \notin S_{mejor-global}$

$\tau_{rs} = (1 - \rho) \cdot \tau_{rs}$

fin para

$mut = mut(it, \tau_{umbral})$

Para cada nodo/componente $r \in \{1, \dots, 1\}$ hacer

z = generar_valor_aleatorio_en_$[0,1]$
Si($z \leq P_m$)
s=generar_valor_aleatorio_en$[1, \dots, 1]$
a=generar_valor_aleatorio_en$[0,1]$
Si $(a = 0)$ $\tau_{rs} = \tau_{rs} + mut$
Si no $\tau_{rs} = \tau_{rs} - mut$
fin si
fin para
si (condicion_de_estacionamiento)
para cada arista a_{rs} hacer $\tau_{rs} = \tau_0$
fin si
fin Procedimiento

Por otro lado, en (Analysis of the Best-Worst Ant System and its variants on the QAP., 2002) O. Cordón, I. Fernández de Viana y F. Herrera propusieron el Sistema de Colonias de la Mejor-Peor Hormiga ("Best-Worst Ant Colony System", BWACS), un nuevo modelo de ACO de buen rendimiento donde se incorporan las ideas presentadas anteriormente al ACS aplicadas al problema de la asignación cuadrática (QAP).

2.2. Optimización basada en Nube de Partículas

La técnica de optimización conocida como Nube de Partículas (Particle Swarm Optimization - PSO) se basa en una población de soluciones guiándose por la inteligencia colectiva de dicha población. Está técnica está inspirada en el comportamiento social de los animales tales como una bandada de pájaros, un cardumen de peces o un enjambre de abejas (Toro, et al., 2006). A continuación se describen aspectos importantes de este algoritmo tratados en (Optimización Basada en Nubes de Partículas (Particle Swarm), 2006-2007).

2.2.1. Funcionamiento del Algoritmo PSO

A continuación se describe el funcionamiento básico del PSO. Este algoritmo simula el comportamiento de las bandadas de aves. Supongamos que una de estas bandadas busca comida en un área y que solamente hay una pieza de comida en dicha área. Los pájaros no saben donde está la comida pero sí conocen su distancia a la misma. La estrategia más eficaz para hallar la comida es seguir al ave que se encuentre más cerca de ella. PSO emula este escenario para resolver problemas de optimización.

Cada solución (partícula) es un “ave” en el espacio de búsqueda que está siempre en continuo movimiento y que nunca muere. La nube de partículas es un sistema multiagente. Las partículas son agentes simples que se mueven por el espacio de búsqueda y que guardan la mejor solución que han encontrado. Cada partícula tiene un fitness (aptitud), una posición y un vector velocidad que dirige su “vuelo”. El movimiento de las partículas por el espacio está guiado por las partículas óptimas en el momento actual.

La técnica PSO tiene una forma de resolver los problemas similar a como lo realizan los algoritmos de Computación Evolutiva[9] (COMPUTACIÓN EVOLUTIVA Y ALGORITMOS BIOINSPIRADOS), codificando las soluciones del problema para que puedan ser procesadas por una computadora de tres formas distintas: la primera como cadenas binarias, secuencias de 0 y 1 donde el dígito de cada posición representa algún aspecto de la solución; otro método consiste en codificar las soluciones como cadenas de enteros o números decimales donde cada posición nuevamente representa algún aspecto particular de la solución y por último un tercer método que consiste en representar a los individuos como cadenas de letras donde cada letra representa algún aspecto específico de la solución. (Marczyk, 2004)

Anatomía de una partícula

Una partícula está compuesta por tres vectores. El vector X almacena la posición actual (localización) de la partícula en el espacio de búsqueda, el vector pBest almacena la localización de la mejor solución encontrada por la partícula hasta el momento, y el vector V almacena el gradiente (dirección) según el cuál se moverá la partícula. Contiene dos valores de fitness: el x_fitness dónde se almacena el fitness de la solución actual (vector X), y el p_fitness almacenando el fitness de la mejor solución local (vector pBest).

$$p_i$$
$$X_i =< X_{i1}, \dots, X_i n >$$
$$pBest_{\ i} =< p_{i1}, \dots, p_i n >$$
$$v_i =< v_{i1}, \dots, v_{in} >$$
$$x_{fitness} =?$$
$$pBest_{fitness} =?$$

[9] Es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas.

La Figura 2.2 muestra gráficamente lo antes mencionado:

Figura 2. 2 Posición de la partícula en el espacio.

Inicialización de la Nube de Partículas

La nube se inicializa generando las posiciones y las velocidades iniciales de las partículas. Las posiciones se pueden generar aleatoriamente en el espacio de búsqueda, de forma regular, o con una combinación de ambas. Las velocidades se generan aleatoriamente, con cada componente en el intervalo $[-\mathrm{V}_{\max}, \mathrm{V}_{\max}]$. No es conveniente fijarlas a cero, no se obtienen buenos resultados. $\mathrm{V}_{\max}$ será la velocidad máxima que pueda tomar una partícula en cada movimiento.

Movimiento de las Partículas

El movimiento de una partícula de una posición del espacio de búsqueda a otra se hace añadiendo el vector velocidad V_i al vector posición X_i para obtener un nuevo vector posición: $X_i \leftarrow X_i + V_i$. Una vez calculada la nueva posición de la partícula, se evalúa ésta. Si el nuevo fitness es mejor que el que la partícula tenía hasta ahora, pBest_fitness, entonces: $pBest_i \leftarrow X_i;\ pBest_{fitness} \leftarrow x_{fitness}$. De este modo, el primer paso es ajustar el vector velocidad, para después sumárselo al vector posición. Las fórmulas empleadas son las siguientes:

$$v_{id} = \omega \cdot v_{id} + \underbrace{\varphi_1 \cdot rnd() \cdot (pBest_{id} - x_{id})}_{\text{cognitivo}} + \underbrace{\varphi_2 . rnd() \cdot (g_{id} - x_{id})}_{\text{social}}$$

$$x_{id} \leftarrow x_{id} + v_{id}$$

donde:

p_i es la partícula, φ_1, φ_2 son ratios de aprendizaje (pesos) que controlan los componentes cognitivo y social, g representa el índice de la partícula con el mejor pBest_fitness del entorno de p_i (lBest) o de toda la nube (gBest), los $rnd()$ son números aleatorios generados en $[0,1]$, y d es la d-ésima dimensión del vector.

Kennedy identifica cuatro tipos de algoritmos de PSO en función de los valores de φ_1 y φ_2:

- Modelo completo: $\varphi_1, \varphi_2 > 0$.
- Sólo Cognitivo: $\varphi_1 > 0,\ \varphi_2 = 0$.
- Sólo Social: $\varphi_1 = 0,\ \varphi_2 > 0$.
- Sólo Social exclusivo: $\varphi_1 = 0,\ \varphi_2 > 0\ y\ g \neq i$ (la partícula en sí no puede ser la mejor de su entorno).

A continuación se muestran los seudocódigoss de algoritmos PSO Local y PSO global:

Seudocódigos PSO Local

$t = 0$;
Para $i = 1$ hasta Numero_particulas
    Inicializar X_i y V_i;
Mientras (no se cumpla la condición de parada) hacer
    $t \leftarrow t + 1$
    Para $i = 1$ hasta Numero_particulas
        evaluar X_i;
        Si $F(X_i)$ es mejor que $F(pBest)$ entonces
            $pBest_i \leftarrow X_i; F(pBest_i) \leftarrow F(X_i)$
    Para i=1 hasta Numero_particulas
        Escoger $lBest_i$, la partícula con mejor fitness del entorno de X_i
        Calcular X_i, la velocidad deX_i, de acuerdo $pBest_i$ y $lBest_i$
        Calcular la nueva posición X_i, de acuerdo a X_i y V_i.
Devolver la mejor solución encontrada

Seudocódigos PSO Global

$t = 0$;
Para $i = 1$ hasta Numero_particulas
    Inicializar X_i y V_i;
Mientras (no se cumpla la condición de parada) hacer
    $t \leftarrow t + 1$
    Para $i = 1$ hasta Numero_particulas
    evaluar X_i;
        Si $F(X_i)$ es mejor que $F(pBest)$ entonces
            $pBest_i \leftarrow X_i; F(pBest_i) \leftarrow F(X_i)$
        Si $F(pBest)$ es mejor que $F(pBest)$ entonces
            $gBest \leftarrow pBest_i, F(gBest_i) \leftarrow F(pBest_i)$
    Para $i = 1$ hasta Numero_particulas
        Calcular V_i, la velocidad de X_i, de acuerdo $pBest_i$ y $gBest_i$
        Calcular la nueva posición X_i, de acuerdo a X_i y V_i
Devolver la mejor solución encontrada

Valores de los parámetros

Los valores de los parámetros, primeramente la nube puede tener un tamaño entre 20 y 40 partículas donde se puede definir problemas simples aquellas que tengan 10 partículas y problemas muy complejos de 100-200 partículas. La Velocidad máxima suele definirse a partir del intervalo de cada variable: los ratios de aprendizaje: Habitualmente, $\varphi_1 = \varphi_2 = 2$.

Estableciendo una comparación entre el PSO Global y el PSO Local se puede decir que la versión global converge más rápido pero cae más fácilmente en óptimos locales y viceversa.

Topologías de la Nube de Partículas

Las topologías definen el entorno de cada partícula individual. La propia partícula siempre pertenece a su entorno. Los entornos pueden ser de dos tipos:

- Geográficos: se calcula la distancia de la partícula actual al resto y se toman las más cercanas para componer su entorno.

- Sociales: se define a priori una lista de vecinas para partícula, independientemente de su posición en el espacio. Este es el más empleado.

Una vez decidido el entorno, es necesario definir su tamaño. Cuando este abarca toda la nube de partículas, el entorno es a la vez geográfico y social, y se tiene la PSO global.

2.2.2. Aspectos avanzados

Control de la Velocidad de las Partículas

Un problema habitual de los algoritmos de PSO es que la magnitud de la velocidad suele llegar a ser muy grande durante la ejecución, con lo que las partículas se mueven demasiado rápido por el espacio. El rendimiento puede disminuir si no se fija adecuadamente el valor de la velocidad máxima inicial de cada componente del vector velocidad. Se han propuesto dos métodos para controlar el excesivo crecimiento de las velocidades:

- Un factor de inercia, ajustado dinámicamente.
- Un coeficiente de constricción.

Factor de Inercia

En este método la ecuación de adaptación de la velocidad quedaría:

$$v_{id} = \omega \cdot v_{id} + \varphi_1 \cdot rnd(\,) \cdot (pBest_{id} - x_{id}) + \varphi_2 \cdot rnd(\,) \cdot (lBest_{id} - x_{id})$$

donde ω se inicializa en 1.0 y se va reduciendo gradualmente a lo largo del tiempo. ω debe mantenerse entre 0.9 y 1.2. Los valores altos provocan la búsqueda global que consiste en más diversificación y los valores bajos inducen una búsqueda más localizada, en otras palabras, más intensificación.

Coeficiente de Constricción

En este método se realiza una nueva modificación a la ecuación de adaptación de la velocidad resultando:

$$v_{id} = K \cdot [v_{id} + \varphi_1 \cdot rnd(\,) \cdot (pBest_{id} - x_{id}) + \varphi_2 \cdot rnd(\,) \cdot (lBest_{id} - x_{id})]$$

donde:

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}$$

$$\varphi = \varphi_1 + \varphi_2$$

$$\varphi > 4$$

Tamaño de la Nube de Partículas

El tamaño de la nube de partículas determina el equilibro entre la calidad de las soluciones obtenidas y el coste computacional (número de evaluaciones necesarias). Hace poco, se han propuesto algunas variantes que adaptan heurísticamente el tamaño de la nube. Si la calidad del entorno de la partícula ha mejorado pero la partícula es la peor de su entorno, se elimina la partícula. Si la partícula es la mejor de su entorno pero no hay mejora en el mismo, se crea una nueva partícula a partir de ella. Las decisiones se toman de forma probabilística en función del tamaño actual de la nube.

Actualización de las Partículas

La actualización de las partículas se puede efectuar de dos formas distintas ya sea asíncrona como síncrona. La actualización asíncrona permite considerar las soluciones nuevas más rápidamente.

En el modelo síncrono todas las partículas se mueven en paralelo. Todas las partículas comparten la misma información acerca de la mejor solución de partida. En el PSO asíncrono cada partícula aprovecha al desplazarse la información actualizada por sus inmediatos predecesores. Al actualizar la información partícula a partícula, el modelo asíncrono acelera la optimización, aunque la naturaleza del modelo síncrono lo hace susceptible de ser ejecutado en paralelo, sobre múltiples procesadores. (Parallel global optimization with the particle swarm algorithm)

2.2.3. Versiones de PSO

Se han trabajado en diferentes versiones de PSO, primeramente con una versión hibridizada del PSO básico al que se lo denominó HPSO. En este algoritmo las partículas se representaron como vectores de números reales para mantener la eficiencia demostrada del algoritmo cuando trabaja sobre espacios continuos y se utilizó un operador de mutación dinámica para mantener la diversidad en la población (Esquivel, et al.).

La versión posterior de HPSO, denominada HPSOvecin, mejoró la performance del primero, a través de la utilización de vecindarios lógicos, logró solucionar una falencia que presentaba HPSO: la convergencia prematura hacia óptimos locales. Esto se logró gracias a que cada partícula además de estar influenciada por el mejor valor encontrado por ella misma, está influenciada por el mejor valor alcanzado por alguno de sus vecinos. Esto produce que cada partícula sea afectada por la mejor performance de un grupo más pequeño de partículas (una del vecindario que comparte), y no por la mejor performance de la mejor partícula de la población como sucedía en HPSO (Esquivel, et al.).

Trabajos posteriores condujeron a una nueva mejora: la inserción de conocimiento específico del problema permitiendo de este modo que el algoritmo ya no realice una búsqueda ciega dentro del espacio de búsqueda sino que guía a las partículas hacia las regiones más promisorias del mismo. Este algoritmo HPSO-kn es el que revela la mejor acción. La inserción de conocimiento se realiza a través de la inclusión de 3 semillas provistas por tres buenas heurísticas. Las mismas son introducidas en la población inicial de partículas y de esta manera las influencian. Los resultados obtenidos han sido enviados para su publicación y actualmente está en proceso de revisión (Esquivel, et al.).

2.3. Optimización basada en Búsqueda por Difusión Estocástica

La Búsqueda por Difusión Estocástica, SDS, nueva metaheurística de la inteligencia colectiva que tienen muchas similitudes con los algoritmos de hormigas y evolutivos (Meyer, et al.). Usa una vía de comunicación directa entre agentes (similar al mecanismo de llamada de tándem empleado por una especie de hormigas, Leptothorax Acervorum, (Tandem calling: a new kind of signal in ant communication., 1974).

SDS utiliza una población de agentes dónde cada agente propone una hipótesis sobre la posible solución y lo evalúa parcialmente. Los agentes exitosos prueban repetidamente su hipótesis mientras que los agentes no exitosos son reclutados por comunicación directa. Esto crea un mecanismo de retroalimentación positivo que asegura la convergencia rápida de agentes hacia las soluciones prometedoras en el espacio de todas las soluciones. Las regiones del espacio de solución calificado por la presencia de grupos de agentes pueden interpretarse como buenas soluciones candidatas. Una solución global se construye así de la interacción de muchos agentes simples localmente operativos que forman el grupo más grande. Semejante grupo es dinámico en la naturaleza, todavía estable, similar a "un bosque cuyos contornos no cambian pero cuyos árboles individuales sí. (Meyer, et al.)

La estocasticidad se puede ver de forma discreta, en forma de árboles de escenarios. Este enfoque considera que las distribuciones de probabilidad de las variables aleatorias están concentradas en valores discretos a lo largo del tiempo. Cada una de estas trayectorias, o evoluciones a lo largo del tiempo, forma un escenario. La fracción de un escenario que corresponde a una etapa se denomina nodo. Los escenarios comparten nodos en las primeras etapas y se ramifican después, para reflejar la incertidumbre creciente con el paso del tiempo (Latorre).

Utilizando el juego del restaurante (ya visto en el Capítulo 1) que contiene los principios de conducta de SDS se verá a continuación proceso algorítmico. Al usar esta estrategia se comprueba que muy rápidamente un gran número de delegados se congrega alrededor del mejor restaurante del pueblo (Meyer, et al.).

Fase de inicialización

Donde todos los agentes (delegados) generan una hipótesis inicial (restaurante)

Lazo

Fase de prueba

Cada agente evalúa la evidencia para su hipótesis (degustación de la comida). Los agentes se dividen en activos (comensales satisfechos) e inactivo (comensales disgustados).

Fase de difusión

Los agentes inactivos adoptan una nueva hipótesis a través de comunicación con otro agente (delegado) o, si el agente seleccionado también es inactivo, no hay flujo de información entre los agentes; en cambio el agente seleccionando debe adoptar una nueva hipótesis (restaurante) al azar.

Fin del Lazo

Por iteración a través de las fases prueba y difusión, los agentes exploran estocásticamente el espacio de la solución completo. Sin embargo, como las pruebas tienen éxito más a menudo en las buenas soluciones candidatas que en las regiones con información irrelevante, un agente individual gastará más tiempo examinando las buenas regiones, al mismo tiempo reclutando a otros agentes que a su

vez reclutan también más agentes. Las soluciones candidatas son identificadas así por las concentraciones de una población sustancial de agentes. (Meyer, et al.)

Muy importante para el poder de SDS es su habilidad de escapar locales de espacio muy reducido. Esto se logra por el resultado probabilístico de la evaluación de la hipótesis parcial en combinación con la reasignación de recursos (agentes) por vía de los mecanismos de reclutamiento estocásticos. La evaluación de la hipótesis parcial le permite a una agente formar su opinión rápidamente sobre la calidad de la solución investigada sin pruebas exhaustivas (por ejemplo puede encontrar el mejor restaurante en el pueblo sin tener que probar todas las comidas disponible en cada uno). (Meyer, et al.)

2.3.1. Terminología

En la formulación original de SDS una población de agentes busca la mejor solución a un problema de optimización dado. El conjunto de todas las soluciones factibles del problema forma el espacio de solución S. Cada punto en S tiene un valor objetivo asociado. Los valores objetivos tomados acerca del espacio de solución completo forma una función objetiva f. Por razones de simplicidad, es asumido que el objetivo es minimizar la suma de $n\{0,1\}$, funciones componentes evaluadas f_i:

$$\min_{\forall s \in S} f(s) = \min_{\forall s \in S} \sum_{i=1}^{n} f_i(s) \quad f_i : S \leftarrow \{0,1\} \qquad (1)$$

Aunque esto puede parecer como una restricción sería, muchos problemas de optimización pueden ser transformados en (1), como se explica en (Meyer, et al., 2007). Durante la operación, cada agente mantiene una hipótesis sobre la mejor solución al problema; una hipótesis es así una solución candidata, o designa un punto en el espacio de solución. Las suposiciones no a-priori son hecho sobre la representación de la hipótesis: ellas pueden ser condiciones binarias, condiciones simbólicas, números entero, o incluso (por lo menos en teoría) números reales. (Meyer, et al.)

2.3.2. Algoritmo

Los agentes en el algoritmo de SDS original operan síncronamente. Ellos sufren varias etapas de operación que se resumen en el siguiente algoritmo (Meyer, et al.):

Inicializar (Agentes);

repetir

Probar (Agentes);

Difundir (Agentes);

hasta (Criterio de parada);

Inicializar

Como un primer paso, los parámetros de la hipótesis de los agentes necesitan ser inicializados. Existen diferentes métodos de inicialización, pero su especificación no se necesita para el entendimiento básico del algoritmo; una discusión puede encontrarse en. (Meyer, et al., 2007)

Probar

Cada agente selecciona al azar una función de componente única, $f_i,\ i \in \{0, \dots, 1\}$ y lo evalúa para su hipótesis particular $s_{h \in} S$. Basado en el resultado de la evaluación, los agentes son divididos en dos grupos: activo e inactivo. Para los agentes activos $f_i(s_h) = 0$; para los agentes inactivos, $f_i(s_h) = 1$. Note que, si se permite a f_i ser probabilístico, es posible que evaluaciones diferentes de $f_i(s_h)$ tengan un resultado distinto. La fase de prueba se describe en seudocódigos a continuación:

```
for agente = 1 to (Todos los Agentes)
        cf = Escoger-AlAzar-Funcion-Componente ();
        if (cf (agente.hipotesis) == 0)
                agente.actividad = TRUE;
        else
                agente.actividad = FALSE;
        end
end
```

Difundir

Durante la fase de difusión, cada agente inactivo escoge al azar otro agente para la comunicación. Si el agente seleccionado es activo, entonces el agente seleccionando copia su hipótesis: la difusión de información. Si el agente seleccionado también es inactivo, entonces no hay ningún flujo de información entre agentes; en cambio, el agente seleccionando adopta una nueva hipótesis aleatoria.

Los agentes activos, de su lado, no empiezan una sesión de comunicación en SDS estándar. La fase de difusión se resume debajo.

```
For agente = 1 to (Todos los Agentes)
        if (agente.actividad == FALSE)
                agente2 = Escoger-AlAzar-Agente (Agentes);
                if (agente2.actividad == TRUE)
                        agente.hipotesis = agente2.hipotesis;
                else
                        agente.hipotesis = Escoger-AlAzar-hipotesis ();
                end
        end
end
```

Parar

Existen diversos tipos de criterio de parada (Meyer, et al., 2007); su especificación no se necesita para la comprensión del algoritmo. El criterio de parada más simple podría ser basado en alcanzar un umbral prescrito de un número total de agentes activos.

2.4. Selección del algoritmo a utilizar

Una vez estudiados los algoritmos de Inteligencia Colectiva y analizada la propuesta de representar el cronograma de un proyecto mediante un grafo, se puede concluir que la metaheurística ACO es la que permitirá llevar a cabo esta propuesta debido a que resuelve problemas que se pueden representar como rutas/caminos entre nodos de un grafo. Además es el más conveniente para resolver el problema de recorrer todos los vértices del grafo sin pasar dos veces por el mismo vértice, utilizando dentro de esta metaheurística el algoritmo sistema de colonia de hormigas pues ha reportado resultados significativos en la solución de problemas de planificación, además de ser uno de los más fáciles de implementar. La técnica PSO a pesar que se ha utilizado para resolver el problema del TSP, no es seleccionada ya que su forma de resolver los problemas es similar a como lo realizan los algoritmos de Computación Evolutiva[10], codificando las soluciones en forma de cadenas donde cada posición representa algún aspecto de la solución. La Búsqueda por Difusión Estocástica también es

[10] Es una técnica de programación que imita a la evolución biológica como estrategia para resolver problemas.

utilizada en los problemas de planificación pero su meta es trazar una trayectoria para alcanzar un blanco específico no la optimización de este camino. Además su ejecución lleva más tiempo y es más costosa.

Conclusiones

En este capítulo se analizaron los algoritmos de Inteligencia Colectiva así como sus características, seudocódigos que permitió concluir que la metaheurística ACO es la más adecuada a utilizar ya que es el que más conveniente para resolver el problema de recorrer todos los vértices del grafo sin pasar dos veces por el mismo vértice

CAPÍTULO 3: IMPLEMENTACIÓN Y RESULTADOS

Introducción

Una vez seleccionado el modelo de Inteligencia Colectiva más adecuado a utilizar en la búsqueda de la planificación de un cronograma con tiempo de ejecución mínimo, se da paso a la implementación, para ello en este capítulo se desarrolla un modelo matemático para representar el cronograma de un proyecto utilizando la teoría de grafos, así como la implementación del algoritmo de Inteligencia Colectiva seleccionado: Optimización basada en Colonia de Hormigas.

Por último se realiza un análisis de los resultados obtenidos a través de una comparación entre el cronograma de proyecto utilizando la implementación del modelo de Inteligencia Colectiva con el cronograma de proyecto utilizando Microsoft Project.

3.1. Lenguaje y Herramientas utilizadas

Lenguaje C++:

El nombre C++ fue propuesto por Rick Mascitti en el año1983. Antes se había usado el nombre C con clases. En C++, la expresión C++ significa incremento de C y se refiere a que C++ es una extensión de C. Constituye un lenguaje potente ya que permite trabajar tanto a alto como bajo nivel, sin embargo es a su vez uno de los que menos automatismos trae, obliga a hacerlo todo manual.

Compilador C++ Builder 6:

El compilador C++ Builder 6 constituye la vía más rápida para desarrollar aplicaciones en C++.Permite reutilización de código. Las aplicaciones que se desarrollan compilan y se ejecutan con mayor rapidez.

3.2. Algoritmos implementados. Factibilidad de ser utilizados

Durante la investigación se analizaron una serie de algoritmos ya implementados con el objetivo de reutilizar el código, estos son: la tesis de grado Clasificación Automática (Delgado, 2004); Problemas de Programación Lineal (samdani), Algoritmos Genéticos (Kohout, 2006), solución TSP (Stützle, 2007) del alemán Thomas Stuetzle perteneciente a la Universidad Darmstadt. Se determinó que no era conveniente reutilizar estos algoritmos ya que se encontraban muy vinculados a las soluciones específicas para los que fueron implementados, por lo que en este trabajo se desarrolló la implementación del algoritmo de hormigas seleccionado para ser utilizado en el modelo propuesto.

3.3. Algoritmo para construir el Modelo Matemático del Cronograma de Proyecto

Entrada:

1) Lista L de tareas del cronograma en la que cada L_i contiene una tarea, su duración, comienzo, fin, número y la lista de tareas predecesoras.

2) Tarea inicial y final del cronograma.

Salida:

Grafo ponderado en la que los vértices se corresponden con tareas y los arcos ponderados (V_i, V_f) con costo C representan la tarea V_f que es antecedida por la tarea V_i y la duración de la tarea V_i es C.

Procedimiento:

1- Crear un grafo G dirigido y ponderado

2- Para cada tarea T_i

> Adicionar un vértice V_i al grafo. El vértice contiene toda la información de la tarea: nombre, duración, comienzo, fin, número y lista de tareas predecesores.

3- Para cada tarea T_i

> Para cada tarea T_j predecesora de T_i
>
> > Adicionar al grafo la arista (V_j, V_i) de costo C_j (V_j y V_i vértices asociados a las tareas T_j y T_i respectivamente; C_j duración de la tarea T_j)

4- Combinar tareas concurrentes. Estas son las tareas que coinciden en una parte de su tiempo y que no deben ser tratadas como vértices distintos del grafo para no sobredimensionar la duración del algoritmo; pues tratarlas en vértices diferentes, contaría el tiempo común entre las dos tareas, de manera doble. Estas tareas deben combinarse en un solo vértice del grafo. Los casos a tener en cuenta a la hora de combinar las tareas son:

- El más simple: una tarea grande que tiene otras tareas concurrentes que están incluidas dentro de ella. Todas las tareas tienen las mismas precedentes. Se utiliza el vértice de mayor duración.

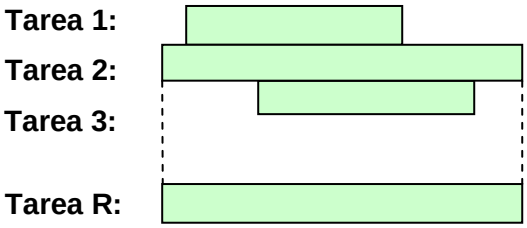


Figura 3. 1 Ejemplo de tareas concurrentes

- Las tareas concurrentes difieren en su inicio y fin, y coinciden en sus predecesoras. Se unifica, teniendo en cuenta el inicio y fin de cada una.

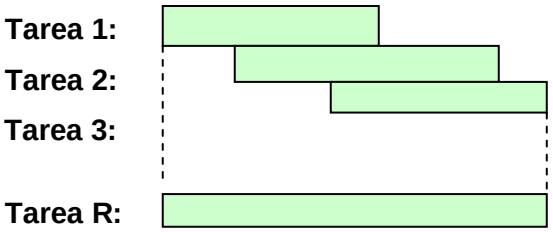


Figura 3. 2 Ejemplo de tareas concurrentes

- Las tareas concurrentes no coinciden en sus predecesoras. Se adoptan las acciones anteriores, según el caso, manteniendo todas las aristas incidentes en las tareas, o sea, las que provienen de sus diferentes predecesoras, es decir, que al vértice de Tarea R incidirían las aristas que incidían sobre Tarea 1, Tarea 2,…, Tarea n.

5- Completar con otras aristas para permitir mayor movilidad de las hormigas en el grafo. Esto es:

Para cada vértice V_i .

Para cada vértice V_j que no tiene a V_i entre sus predecesores

Adicionar la arista (V_j, V_i)

6- Agregarle un nodo final F, ya que la duración de la última tarea necesita una arista con ese peso para que no se pierda el dato de su duración.

En este modelo, queda pendiente que un vértice V_i pudiera tener varios vértices predecesores y por lo tanto no puede visitarse V_i hasta que no se hayan visitado todos sus predecesores. Esto no se logra al construir el grafo, sino con una modificación en el algoritmo de hormigas.

3.4. Diagrama de Clases

Para dar solución a la implementación del modelo de conversión de cronograma a grafo como la implementación del algoritmo de optimización de colonia de hormigas, se desarrollaron las siguientes clases: Fecha, Tarea, TareaCombinada, Cronograma, Hormiga, ColoniaHormigas, ColoniaCronograma como se muestra en la Figura 3.3.

3.4.1. Clase Cronograma

La clase principal es Cronograma, en la cual se almacena la lista de tareas leídas del cronograma a través de un XML, cuya estructura se muestra en el Anexo 1, esta lista desde este momento pasa a llamarse lista de tareas simples; la lista de tareas combinadas, que resulta de combinar las tareas concurrentes y el grafo que se corresponde con el cronograma, cuyos nodos son tareas combinadas y las aristas, va a estar dadas por el algoritmo definido anteriormente.

El constructor de Cronograma recibe como parámetro el XML del cual se lee el cronograma Y crea la lista de tareas simples en el que cada elemento se corresponde con una tarea leída del cronograma representado en el XML.

El método combinarTareas() comienza por la primera tarea buscando entre todas cuales se solapan con esta y las adiciona en un objeto de tipo TareaCombinada, el mismo se va guardando en la lista de tareas combinadas. De la misma manera se repite con el resto de las tareas que no han sido combinadas y las va combinando, de manera que cuando termina de ejecutarse el método, la lista de tareas simples quedó transformada en una lista de tareas combinadas, cuya longitud es menor o igual a la longitud de la primera. En el mejor caso, una tarea combinada solo estaría compuesta por una sola tarea simple si es que esta no se solapa con ninguna otra.

La lista de tareas combinadas es representada en un grafo a través del método tareasAGrafo(), cuya implementación se basa en el modelo matemático mencionado en el epígrafe anterior.

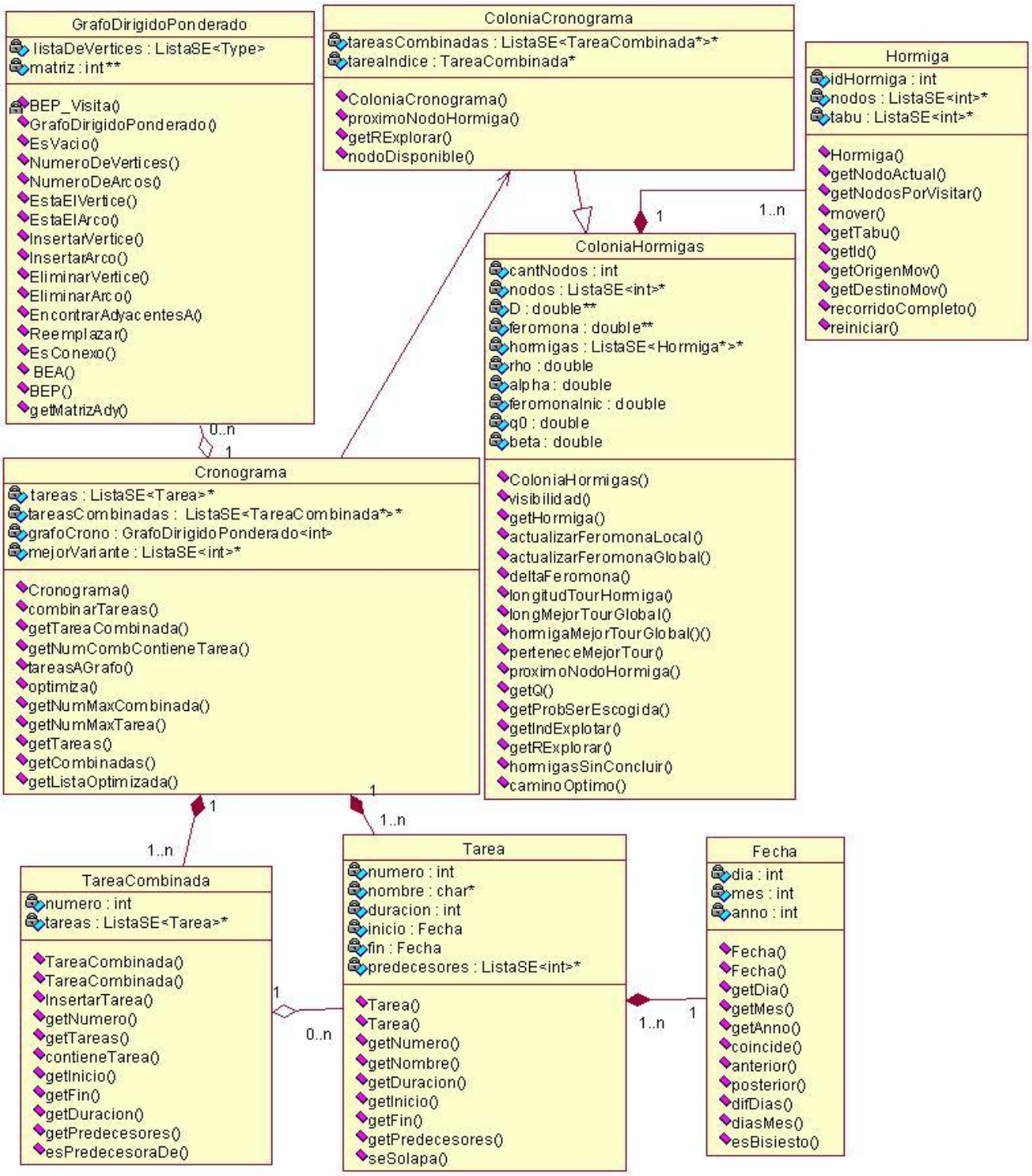


Figura 3. 3 Diagrama de Clases.

3.4.2. Clases Hormiga y ColoniaHormigas

Las clases Hormiga y ColoniaHormiga se corresponden con la implementación de la colonia de hormigas general para usar en cualquier problema futuro.

Entre los principales métodos de la clase Hormiga se encuentran getNodosPorVisitar() que devuelve la lista de nodos que aún no han sido visitados; mover(int destino) que le permite a la hormiga desplazarse al nuevo nodo adicionando este a la lista tabú, que contiene almacenado los nodos ya visitados, esta lista puede ser accedida a través del método getTabu().

La clase ColoniaHormigas está conformada por varios métodos entre los que se destacan proximoNodoHormiga(int id) mediante el cual una hormiga ubicada en el nodo i elige el próximo nodo j a visitar; actualizarFeromonaLocal(int id) permitiendo la actualización de la cantidad de feromona de una arista después del movimiento de la hormiga; actualizarFeromonaGlobal() es en este donde se actualiza la feromona una vez terminado el tour de todas las hormigas; a través del longMejorTourGlobal() se obtiene la longitud del mejor camino encontrado por las hormigas y por último caminoOptimo(int cantCiclos) que devuelve el mejor camino después de haberse ejecutado el algoritmo.

3.4.3. Clase ColoniaCronograma

La clase ColoniaCronograma, es la implementación de hormigas específica para resolver el problema optimización del cronograma, esta clase hereda de ColoniaHormigas y tiene como objetivo redefinir la selección del próximo nodo a visitar para tener en cuenta que una hormiga no puede moverse a un nodo hasta que no se haya visitado todos sus predecesores, lo cual quedaba pendiente del modelo matemático definido en el Epígrafe 3.3, para ello cuenta con el método getRExplorar(int idHormiga) en el cual se redefine la exploración del próximo nodo buscando el de mayor probabilidad que esté disponible y viéndose la disponibilidad a través del método nodoDisponible(int idHormiga,int nodo) .La disponibilidad es la posibilidad de que un nodo sea visitado porque ya todos sus predecesores lo fueron.

3.4.4. Interfaz

Con el objetivo de comprobar la validez del modelo propuesto se desarrolló una interfaz sencilla. A continuación se muestran sus principales elementos.

En la Figura 3.4 se muestra la ventana principal; como se observa los menús desplegables Cronograma, Grafo y Operaciones no están activados debido a que no se ha importado ningún cronograma. Mediante el menú Archivo se importa el XML donde es leído el cronograma.

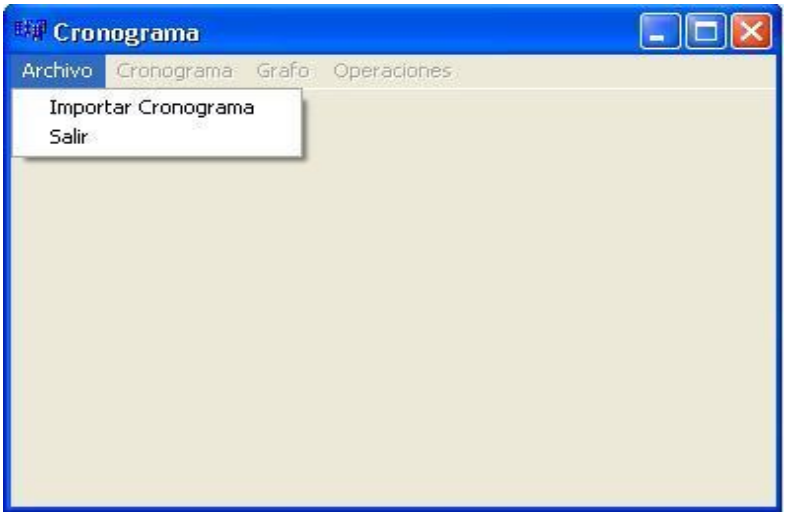


Figura 3. 4 Ventana principal.

Las opciones que permite el menú Cronograma son mostradas a través de la Figura 3.5, en esta no se encuentra activada la opción de mostrar las tareas optimizadas ya que todavía no se ha procedido a la optimización del cronograma.

Figura 3. 5 Menú Cronograma.

La lista de tareas que forman el cronograma puede obtenerse mediante la opción Ver Tareas del menú Cronograma, como se muestra en la Figura 3.6.

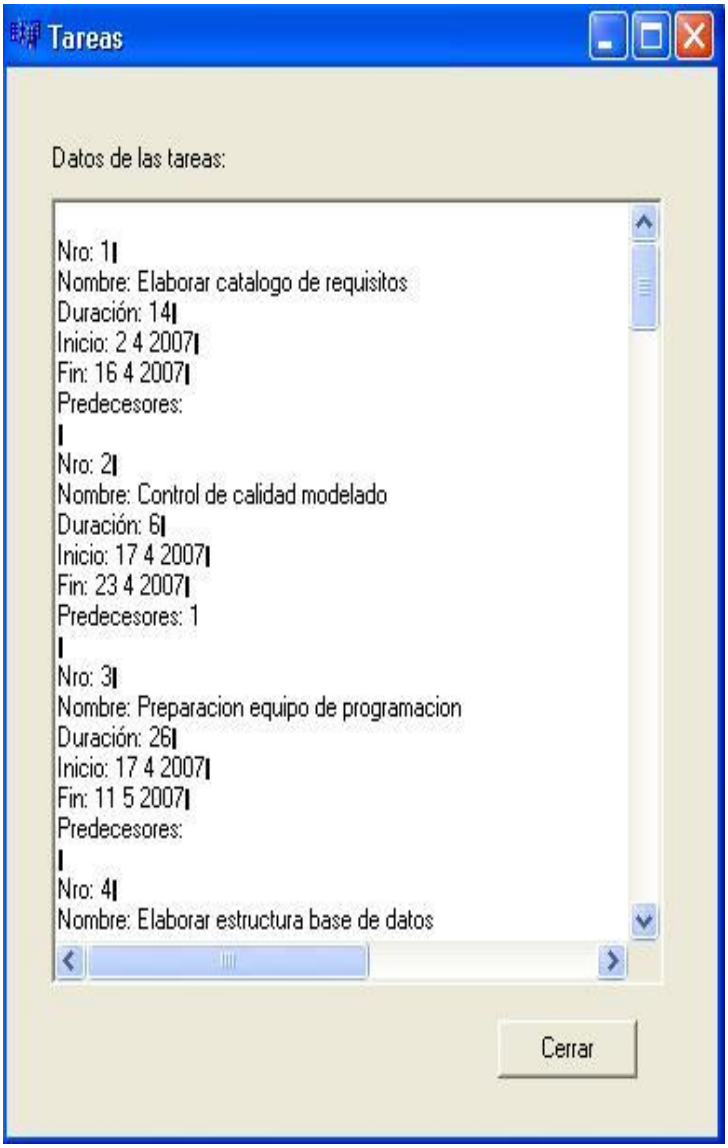


Figura 3. 6 Lista de tareas del cronograma.

La lista de Tareas Combinadas que se obtienen aplicando el modelo matemático es consultada a través del menú Cronograma como se observa en la Figura.3.7

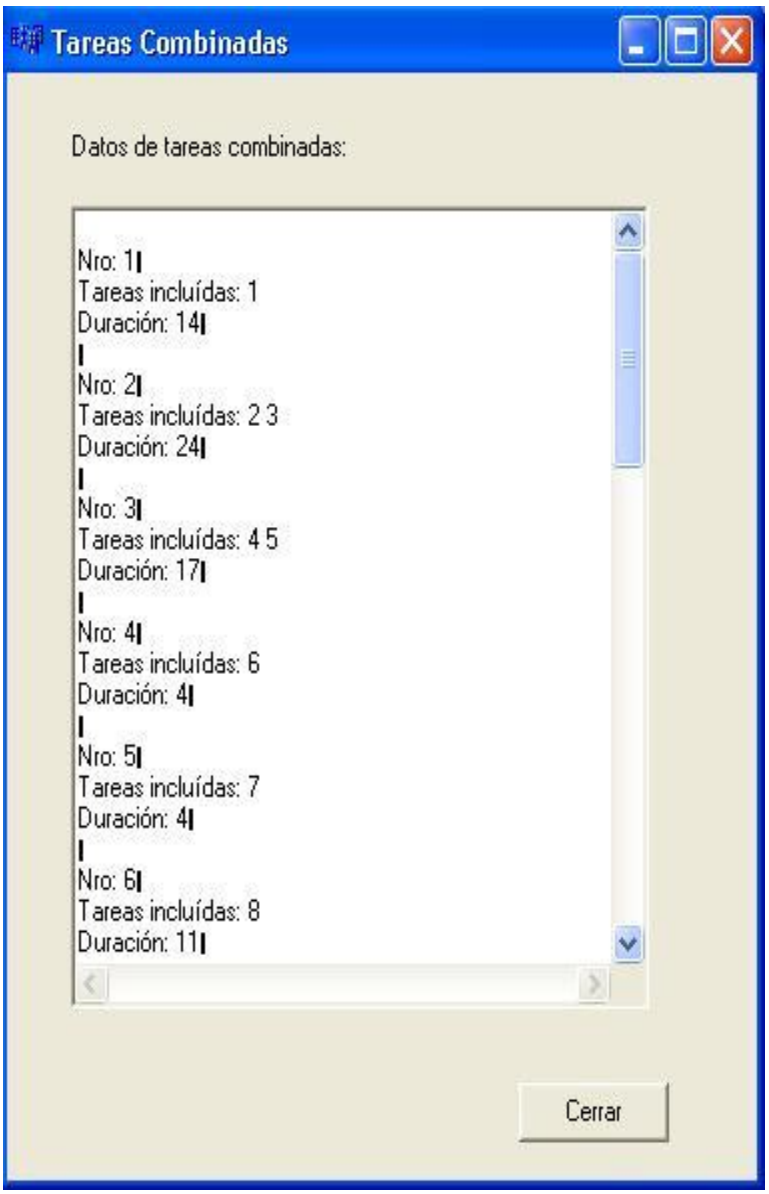
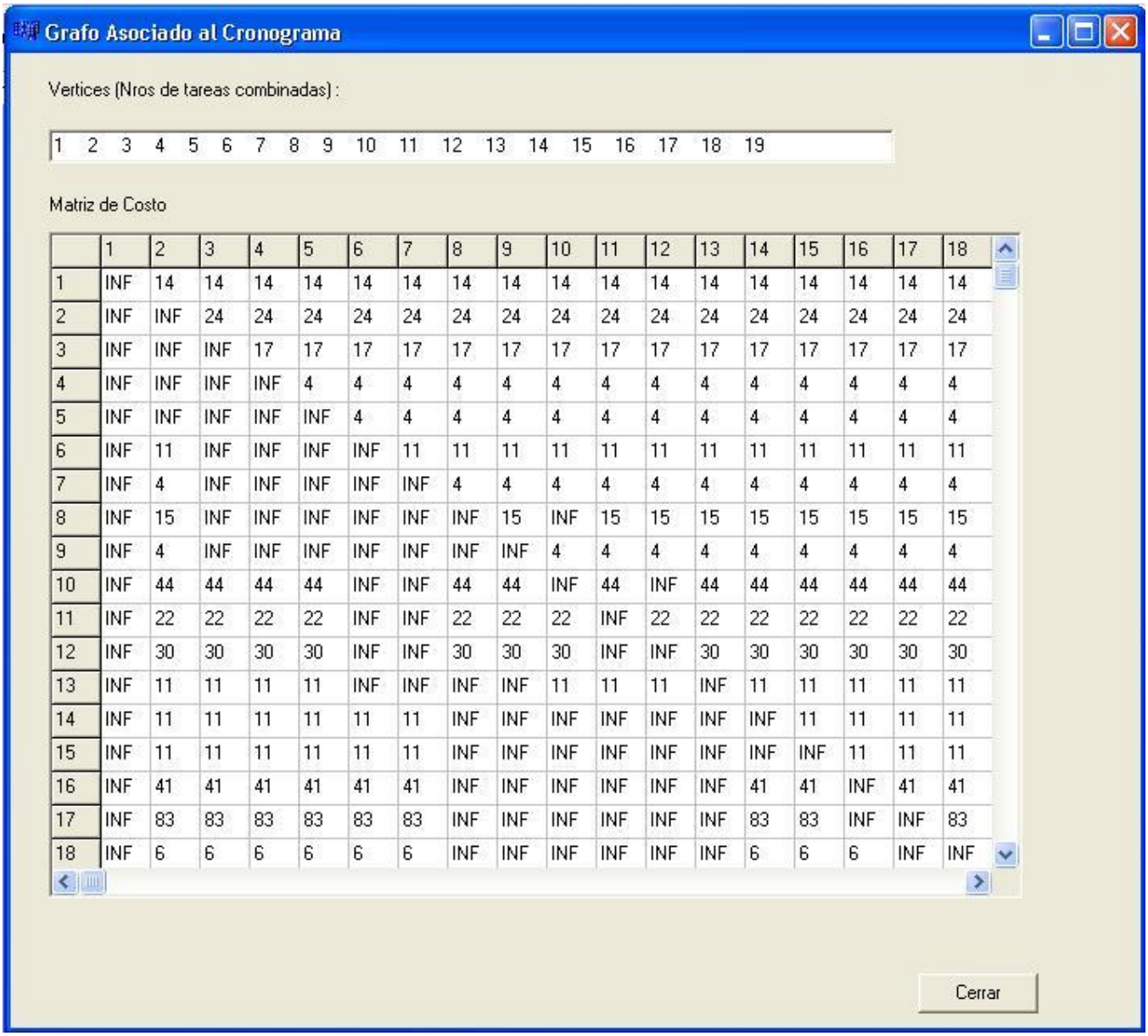


Figura 3. 7 Listado de tareas combinadas.

El menú Grafo muestra el Grafo Asociado al Cronograma, el mismo es mostrado en la Figura 3.8.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	INF	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14	14
2	INF	INF	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24	24
3	INF	INF	INF	17	17	17	17	17	17	17	17	17	17	17	17	17	17	17
4	INF	INF	INF	INF	4	4	4	4	4	4	4	4	4	4	4	4	4	4
5	INF	INF	INF	INF	INF	4	4	4	4	4	4	4	4	4	4	4	4	4
6	INF	11	INF	INF	INF	INF	11	11	11	11	11	11	11	11	11	11	11	11
7	INF	4	INF	INF	INF	INF	INF	4	4	4	4	4	4	4	4	4	4	4
8	INF	15	INF	INF	INF	INF	INF	INF	15	INF	15	15	15	15	15	15	15	15
9	INF	4	INF	INF	INF	INF	INF	INF	INF	4	4	4	4	4	4	4	4	4
10	INF	44	44	44	44	INF	INF	44	44	INF	44	INF	44	44	44	44	44	44
11	INF	22	22	22	22	INF	INF	22	22	22	INF	22	22	22	22	22	22	22
12	INF	30	30	30	30	INF	INF	30	30	30	INF	INF	30	30	30	30	30	30
13	INF	11	11	11	11	INF	INF	INF	INF	11	11	11	INF	11	11	11	11	11
14	INF	11	11	11	11	11	11	INF	INF	INF	INF	INF	INF	INF	11	11	11	11
15	INF	11	11	11	11	11	11	INF	INF	INF	INF	INF	INF	INF	INF	11	11	11
16	INF	41	41	41	41	41	41	INF	INF	INF	INF	INF	INF	41	41	INF	41	41
17	INF	83	83	83	83	83	83	INF	INF	INF	INF	INF	INF	83	83	INF	INF	83
18	INF	6	6	6	6	6	6	INF	INF	INF	INF	INF	INF	6	6	6	INF	INF

Figura 3. 8 Grafo Asociado al Cronograma.

En el menú Operaciones mediante la opción Optimizar cronograma se optimiza el cronograma y cuando concluye muestra el siguiente mensaje, como se observa en la Figura 3.9:

Figura 3. 9 Optimización concluida.

El cronograma optimizado es visualizado mediante el menú Cronograma con la opción Ver Tareas Optimizadas como se muestra en la Figura 3.10

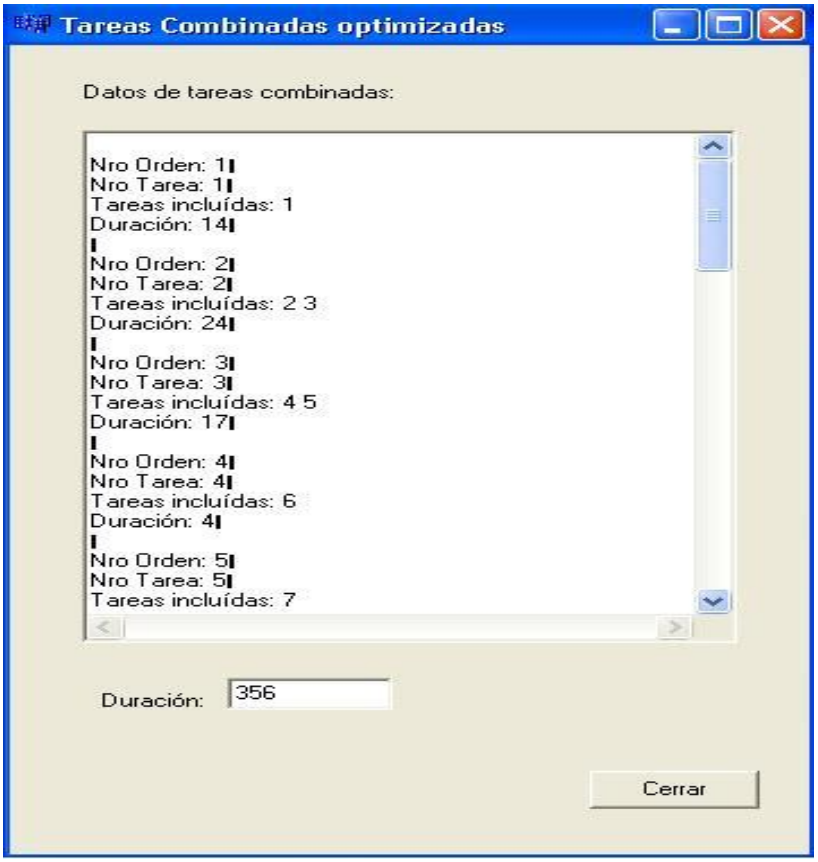


Figura 3. 10 Tareas Combinadas Optimizadas

3.5. Resultados

Para comprobar la validez de la propuesta se trabajó con un cronograma de prueba (Ver Anexo1). Primeramente se representó este cronograma utilizando Microsoft Office Project 2003, el cual tiene una duración de 413 días. La Figura 3.11 muestra el Diagrama de Gantt representado en el Project.

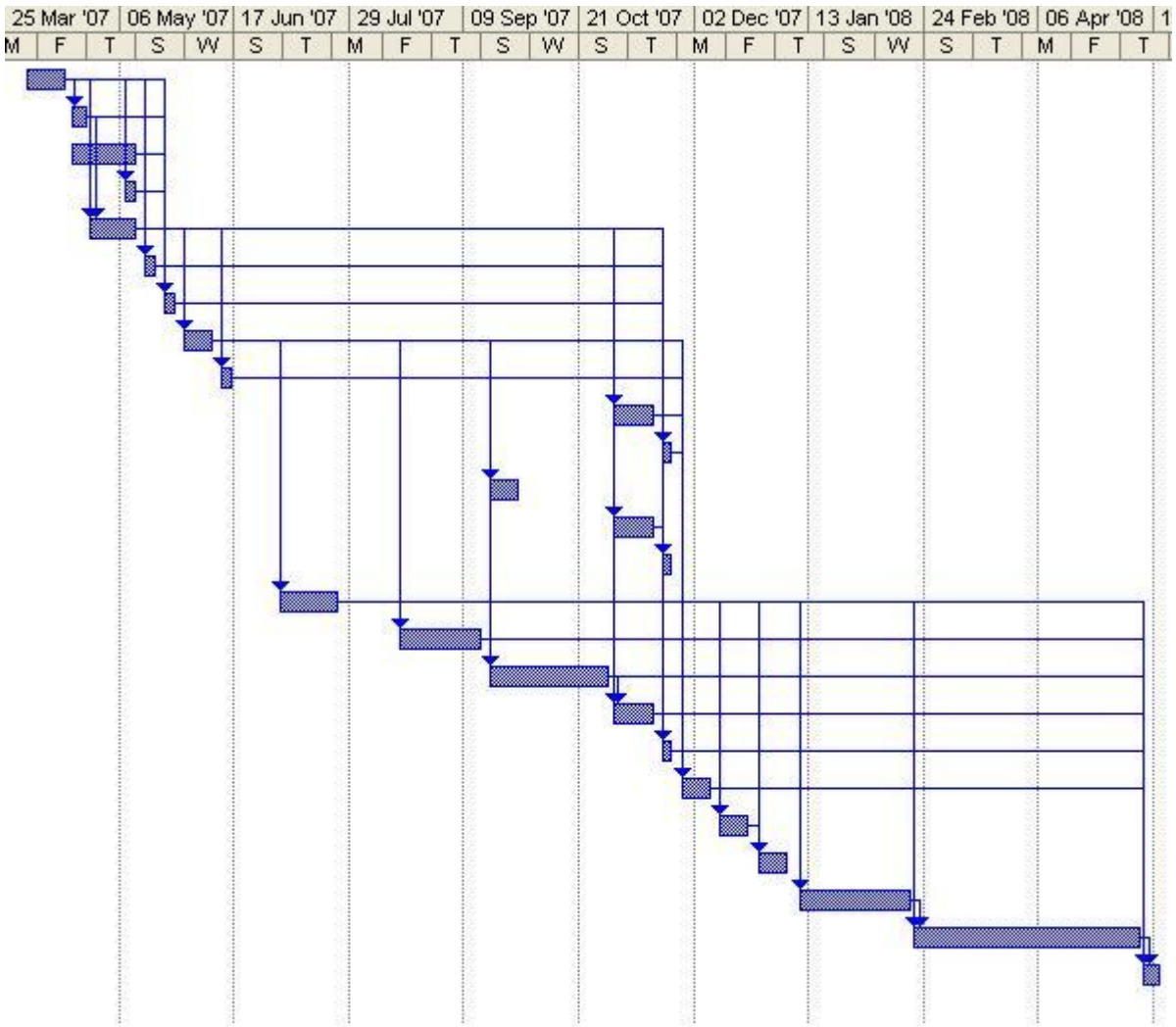


Figura 3. 11 Diagrama de Gantt.

A continuación se procedió a optimizar el cronograma utilizando la propuesta presentada. En ella la optimización se logra por dos aspectos importantes, combinando las tareas que concurren en algún momento en el tiempo y aplicando el algoritmo.

La combinación de las tareas se obtiene aplicando el modelo matemático explicado en el Epígrafe 3.3. De esta combinación resultan las tareas combinadas que se observan en la tabla 3.1

Número de la Tarea combinada	Tareas simples que la componen	Duración
1	1	14
2	2,3	24
3	4,5	17
4	6	4
5	7	4
6	8	11
7	9	4
8	10,13,18	15
9	11,14,19	4
10	12,27	44
11	15	22
12	16	30
13	20	11
14	21	11
15	22	11
16	23	41
17	24	83
18	25	6

Tabla 3. 1 Tareas Combinadas

Una vez combinadas las tareas, se representaron en un grafo. Se buscó el camino mínimo para recorrer todo el grafo utilizando el algoritmo y se obtuvo como resultado el cronograma optimizado,

representado en la Tabla 3.2 con una duración total de 356 días. La optimización de este cronograma representa una reorganización de las tareas.

Número Orden	Número de Tarea	Tarea Incluidas	Duración
1	1	1	14
2	2	2,3	24
3	3	4,5	17
4	4	6	4
5	5	7	4
6	6	8	11
7	7	9	4
8	11	15	22
9	12	16	30
10	10	12,17	44
11	8	10,13,18	15
12	9	11,14,19	4
13	13	20	11
14	14	21	11
15	15	22	11
16	16	23	41
17	17	24	83
18	18	25	6

Tabla 3. 2 Cronograma Optimizado

Conclusiones

La optimización del cronograma de un proyecto representa una reducción de su tiempo de ejecución, esto tiene gran importancia pues muchas veces no se aprovecha el tiempo al máximo. En este capítulo se realizó la implementación del algoritmo Sistema Colonia de Hormigas seleccionado en el capítulo anterior, que permitió cumplir con el objetivo de reducir el tiempo de ejecución del cronograma, ya que se obtuvo uno optimizado, es decir, con menor tiempo de duración. El algoritmo anteriormente mencionado quedó implementado de forma que puede ser reutilizado en otros problemas de búsqueda del camino mínimo.

CONCLUSIONES GENERALES

Los algoritmos de Inteligencia Colectiva se utilizan en el mundo para solucionar un gran número de problemas y son diversas sus aplicaciones. Con este trabajo se implementó un modelo de Inteligencia Colectiva que permitió reducir el tiempo de ejecución de un proyecto mediante una planificación óptima del cronograma.

Se estudiaron varias herramientas que se utilizan para elaborar los cronogramas de proyecto. Se analizaron tres modelos y se obtuvo como resultado que el modelo más adecuado a utilizar para la búsqueda de la planificación óptima según la propuesta que se hace es la Optimización basada en Colonia de Hormigas.

Se desarrolló un modelo matemático que permitió representar el cronograma en un grafo. Se implementó el modelo de hormigas genérico que puede utilizarse en cualquier problema del viajero vendedor en el que se quiera visitar todos los nodos y minimizar el camino recorrido. Se le dio solución al problema de optimización del cronograma aplicando Hormigas.

RECOMENDACIONES

- Valorar otros modelos para convertir el cronograma a grafo, de manera que se logre una topología de grafo más apropiada, que permita optimizar aun más el tiempo de ejecución del cronograma.
- Desarrollar otras variantes de ACO, como por ejemplo, una variante paralela, para minimizar el tiempo de ejecución del algoritmo.
- Desarrollar una herramienta para la planificación de proyectos utilizando este algoritmo.
- Utilizar la implementación de Colonia de Hormigas brindada para dar solución a otros problemas y validar mejor la implementación dada.

BIBLIOGRAFÍA

Addlink Software Científico.Operations Research 3.0. **2007.** 2007.

Aguirre, Roberto. 2007. *Ant Farm Simulator.* 2007.

Alcalá, R., et al. 2001. *Improvement to the cooperative rules methodology by using the Ant Colony System algorithm.* 2001. pp. 321-335.

Alonso, Sergio, et al. *La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques.*

Alonso, Sergio, et al. *La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques.*

Analysis of the Best-Worst Ant System and its variants on the QAP. **Cordón, O., Fernández, I. and Herrera, F. 2002.** 2002, Proceedings of ANTS2002 - From Ant Colonies to Artificial Ants: Third International Workshop on Ant Algorithms, Vol. 2463, pp. 228-234.

Ant can colour graphs. **Costa, D. and Hertz, A. 1997.** 1997, Journal of the Operational Research Society, pp. 295-305.

Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. **Dorigo, M. and Gambardella, L. 1997.** 1997, IEEE Transaction on Evolutionary Computation,, Vol. Vol 1, pp. 53-66.

Arencibia, Joel. 2007. *Implementaciones paralelas del Modelo de Colonia de Hormigas.* 2007.

Atsp. **2007.** 2007.

Barán, B. and M.Almirón. 2002. *Colonia de hormigas en un ambiente paralelo asíncrono.* 2002.

Bioinformática. **2006-2007.** 2006-2007.

Bishop, J M, Torr, P and Lingard, R. 1992. *The Stochastic Search Network.* [ed.] D J Myers and Nightingale. Chapman and Hall, New York : s.n., 1992. C Neural Networks for Images, Speech and Natural Language. .

Blum, Christian. 2007. *hc-mmas-ubqp.* 2007.

Bullnheimer, B., Hartl, R. and Strauss, C. 1999. *An improved ant system algorithm for the vehicle routing problem. Annals of Operations Research.* 1999.

Campos, L. de, Gámez, J. and Puerta, J. 2002. *Learning bayesian networks by ant colony optimisation: searching in two different spaces.* 2002. pp. 251-268.

Capacitated vehicle routing problem. **2007.** 2007.

Cepeda, Lina, Benavides, Grace and Lopez, Maria. Inteligencia Artificial. [Online] http://www.monografias.com/trabajos16/la-inteligencia-artificial/la-inteligencia-artificial.shtml?monosearch.

COMPUTACIÓN EVOLUTIVA Y ALGORITMOS BIOINSPIRADOS. **LOZANO, M. and HERRERA, F.** SOFT COMPUTING Y SISTEMAS INTELIGENTES.

Cordón, Oscar. 2007. Imitar a las hormigas para resolver problemas empresariales. *Diario independiente de Asturia.* 2007.

De Meyer, K, Bishop, J M and J, Nasuto S. 2000. *Attention through SelfSynchronisation in the Spiking Neuron Stochastic Di®usion Network. Consciousness and Cognition 9(2).* 2000.

Delgado, Yeneit. 2004. *Clasificación automática mediante Colonia de Hormigas. Configuración automática de parámetros.* Facultad de Matemática y Computación, Universidad de La Habana. La Habana : s.n., 2004.

Dorigo, M. and Caro, G. Di. 1999. *The Ant Colony Optimization meta-heuristic.* McGraw Hill,London, UK : s.n., 1999. pp. 11-32.

Dorigo, M., Maniezzo, V. and Colorni., A. 1996. *The Ant System: Optimization by a colony ofcooperating agents.* 1996.

Escenarios para el aprendizaje de tendencias bioinspiradas. **Jiménez, Daniel and Peñuela, David. 2007.** 2007, REDIS.

Esquivel, Susana, Aragón, Victoria and Cagnina, Leticia. *Optimización Mono y Multiobjetivo a través de Métodos de Aproximación de Soluciones en Ambientes Estacionarios y No Estacionarios.*

Farooq, Muddassar. 2007. *Antnet-1.1,.* 2007.

Gambardella, L. and Dorigo, M. 2000. *Ant Colony System hybridized with a new local search for the sequential ordering problem.* 2000. pp. 237-255.

García, Jose. 2007. Algoritmos Basados en Inteligencia Colectiva para la Resolución de Problemas de Bioinformática y Telecomunicaciones. [Online] 2007. http://neo.lcc.uma.es/staff/jmgn/doc/Proyecto_master.pdf.

García, Luis. 2004. Usos y aplicaciones de la inteligencia artificial. [Online] Septiembre - Diciembre 2004. http://www.uv.mx/cienciahombre/revistae/vol17num3/articulos/inteligencia/index.htm.

GONZÁLEZ, J. G. I. *NGELIGENCIA COLECTIVA.* Seminario LISI.

Grech-Cini, E. 1995. *Locating Facial Features.* University of Read-ing. 1995.

Hamiltonian cycle problem. **2007.** 2007.

Inteligencia Colectiva y sus Aplicaciones. **Arencibia, Joel. 2007.** 2007.

Kennedy, J and Eberhart, R. Optimización Basada en Nubes de Partículas (Particle Swarm). [Online] http://sci2s.ugr.es/docencia/bioinformatica/Tema_4_PSO_06-07.pdf.

Kohout, Peter. 2006. Genetic and Ant Colony Optimization Algorithms. [Online] 2006. http://www.codeproject.com/KB/recipes/GeneticandAntAlgorithms.aspx.

Latorre, Jesús. *Resolución Distribuida de Problemas de Optimización Estocástica. Aplicación al problema de coordinación hidrotérmica.* Instituto de Investigación Tecnológica, UNIVERSIDAD PONTIFICIA COMILLAS. ESCUELA TECNICA SUPERIOR DE INGENIERIA (ICAI). Madrid : s.n.

Locating the Mouth Region in Images of Human Faces. **Grech-Cini, McKee, H.J. & and G.T. 1993.** [ed.] P.S.Schenker. 1993. Proceedings of SPIE - The International Society for Optical Engineering. Vol. Sensor Fusion VI 2059.

Marczyk, Adam. 2004. Algoritmos genéticos y computación evolutiva. [Online] 2004. http://sci2s.ugr.es/docencia/sf1/Sesion-3b-F-Herrera-Swarm-Intelligence-Alg-Bio-Comp-Evo-07-08.pdf.

Meyer, Fernando and Parpinelli, Rafael. 2007. *GUIAnt-Miner.* 2007.

Meyer, Kris De, Slawomir, J. Nasuto and Bishop, Mark. Stochastic Diffusion Search: partial function evaluation in swarm intelligence dynamic optimisation. [Online] http://www.doc.gold.ac.uk/~mas02mb/Selected%20Papers/2006%20Swarm%20Intelligence.pdf.

Nasuto, S J and Bishop, J M. 1999. *Convergence Analysis of Stochastic Diffusion Search. Parallel Algorithms and Applications.* 1999.

1998. *Neural Stochastic Diffusion Search Network -a Theoretical Solution to the Binding Problem. Proc. ASSC2, Bremen.* 1998.

Nasuto, S J. 1999. *Resource Allocation Analysis of the Stochastic Diffusion Search.* 1999.

Nasuto, S J, Bishop, J M and Lauria, S. 1998. *Time Complexity of Stochastic Diffusion Search.* 1998.

Nasuto, S J, Dautenhahn, K and Bishop, J M. 1999. *Communication as an Emergent Methaphor for Neuronal Operation.* 1999.

Optimización (matemática).

Optimización Basada en Nubes de Partículas (Particle Swarm). **Kennedy, J and Eberhart, R. 2006-2007.** 2006-2007.

Parallel global optimization with the particle swarm algorithm. **Schutte, J., et al.** International Journal for Numerical Methods in Engineering, Vol. 61, pp. 2296-2315.

Parodi, C. 2001. *«El lenguaje de los proyectos», en Gerencia social. Diseño, monitoreo y evaluación de proyectos sociales.* Lima-Perú : s.n., 2001.

Parpinelli, R., Lopes, H. and Freitas, A. 2002. *Data mining with an Ant Colony Optimization algorithm.* 2002. pp. 321-332.

Pérez, Jesús. 2005. Contribución a los métodos de optimización basados en procesos naturales y su aplicación a la medida de antenas en campo próximo. [Online] 2005. http://www.tdx.cbuc.es/TESIS_UC/AVAILABLE/TDR-0305107-180847//0de8.JRPL_previo.pdf.

Rennard, Jean-Philippe. 2007. *Ants Viewer 1.0.* 2007.

samdani, saurabh. [Online] http://www.geocities.com/saurabhsamdani/sourcecodes.html.

Saravia, Roger. 2007. *Agentes Inteligentes.* Facultad de Ciencias Puras y Naturales, UNIVERSIDAD MAYOR DE SAN ANDRES. La Paz, Bolivia : s.n., 2007.

Self-localisation in the senario autonomous wheel-chair. **Beattie, P and Bishop, J. 1998.** 1998, Journal of Intelligent and Robotic Systems 22, pp. 255-267.

Sequential ordering problem. **2007.** 2007.

SimAnt, Maxis Software Inc. **2007.** 2007.

2007. StarLogo. [Online] 2007. http://educationmit.edu/starlogo/..

Stochastic Searching Networks. **Bishop, J.M. 1989.** London : s.n., 1989. Proc. 1st IEE Conf. on Artificial Neural Networks.

Bishop, J.M. Proc. 1st IEE Conf. on Artificial Neural Networks. pp. 329-331.

Stützle, T. 1998. An ant approach to the flow shop problem. *En Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98).* 1998, Vol. 3, pp. 1560-1564.

Stützle, Thomas. 2007. *ACOTSP.V1.0.* 2007.

Tandem calling: a new kind of signal in ant communication. **M, Moglich, U, Maschwitz and B, Holldobler. 1974.** 1974, Science .

Thomas, Kirenia Nariño. 2007. *Propuesta de un procedimiento para la planificación del Proyecto: Gestión Hospitalaria.* Facultad 7, Universidad de las Ciencias Informáticas. Ciudad de la Habana. : s.n., 2007. Trabajo de Diploma .

Toro, Eliana, Restrepo, Yov and Granada, Mauricio. 2006. ADAPTACIÓN DE LA TÉCNICA DE PARTICLE SWARM AL PROBLEMA DE SECUENCIAMIENTO DE TAREAS. [Online] 2006. http://www.utp.edu.co/php/revistas/ScientiaEtTechnica/docsFTP/161217revista1111111.pdf.

Torres, Enrique La. 2008. Revista Internacional de Ciencias de la Tierra. [Online] mayo 2008. http://www.mappinginteractivo.com/plantilla.asp?id_articulo=1482.

Whitaker, R.M. and Hurley, S. *An agent based approach to site selection for wireless networks.* Madrid : s.n. Proc ACM Symposium on Applied Computing.

ANEXOS

Anexo1.Cronograma de Proyecto.

```
<?xml version="1.0" encoding="UTF-8"?>
<Cronograma>
 <modelo_negocio>
    <tarea>
        <numero>1</numero>
            <nombre>Elaborar catalogo de requisitos</nombre>
        <duracion>14 dias</duracion>
        <inicio>lunes 02/04/07</inicio>
            <fin>lunes 16/04/07</fin>
            <predecesores>0</predecesores>
          </tarea>
          <tarea>
        <numero>2</numero>
            <nombre>Control de calidad modelado</nombre>
        <duracion>6 dias</duracion>
        <inicio>martes 17/04/07</inicio>
            <fin>lunes 23/04/07</fin>
            <predecesores>1</predecesores>
          </tarea>
          <tarea>
        <numero>3</numero>
            <nombre>Preparacion equipo de programacion</nombre>
        <duracion>24 dias</duracion>
        <inicio>lunes 17/04/07</inicio>
            <fin>viernes 11/05/07</fin>
            <predecesores>0</predecesores>
          </tarea>
```

```
        <tarea>
      <numero>4</numero>
          <nombre>Elaborar estructura base de datos</nombre>
      <duracion>4 dias</duracion>
      <inicio>lunes 07/05/07</inicio>
          <fin>viernes 11/05/07</fin>
          <predecesores>1</predecesores>
        </tarea>
</modelo_negocio>
<analisis>
  <tarea>
      <numero>5</numero>
          <nombre>Elaborar descripciones de casos de uso</nombre>
      <duracion>17 dias</duracion>
      <inicio>lunes 24/04/07</inicio>
          <fin>viernes 11/05/07</fin>
          <predecesores>1,2</predecesores>
        </tarea>
        <tarea>
      <numero>6</numero>
          <nombre>Elaborar diagrama de clases de analisis</nombre>
      <duracion>4 dias</duracion>
      <inicio>martes 14/05/07</inicio>
          <fin>viernes 18/05/07</fin>
          <predecesores>1,2,3,4,5</predecesores>
        </tarea>
        <tarea>
      <numero>7</numero>
          <nombre>Control de calidad analisis</nombre>
      <duracion>4 dias</duracion>
```

```
        <inicio>lunes 21/05/07</inicio>
            <fin>viernes 25/05/07</fin>
            <predecesores>1,2,3,4,6</predecesores>
          </tarea>
</analisis>
<disenno>
   <tarea>
        <numero>8</numero>
            <nombre>Elaborar diagramas de secuencias</nombre>
        <duracion>11 dias</duracion>
        <inicio>lunes 28/05/07</inicio>
            <fin>viernes 08/06/07</fin>
            <predecesores>5,6,7</predecesores>
          </tarea>
          <tarea>
        <numero>9</numero>
            <nombre>Elaborar diagrama de clases de disenno</nombre>
        <duracion>4 dias</duracion>
        <inicio>lunes 11/06/07</inicio>
            <fin>viernes 15/06/07</fin>
            <predecesores>5,6,7,8</predecesores>
          </tarea>
          <tarea>
        <numero>10</numero>
            <nombre>Elaborar diagrama de secuencias nuevos requisitos</nombre>
        <duracion>15 dias</duracion>
        <inicio>jueves 01/11/07</inicio>
            <fin>viernes 16/11/07</fin>
            <predecesores>5,6,7,9</predecesores>
          </tarea>
```

```
        <tarea>
      <numero>11</numero>
          <nombre>Control de calidad disenno</nombre>
      <duracion>4 dias</duracion>
      <inicio>lunes 19/11/07</inicio>
          <fin>viernes 23/11/07</fin>
          <predecesores>5,6,7,10</predecesores>
        </tarea>
</disenno>
<manual_usuario>
  <tarea>
      <numero>12</numero>
          <nombre>Elaborar version 1.0l manual de usuario</nombre>
      <duracion>11 dias</duracion>
      <inicio>lunes 17/09/07</inicio>
          <fin>viernes 28/09/07</fin>
          <predecesores>8,9</predecesores>
        </tarea>
        <tarea>
      <numero>13</numero>
          <nombre>Elaborar version definitiva manual de usuario</nombre>
      <duracion>15 dias</duracion>
      <inicio>jueves 01/11/07</inicio>
          <fin>viernes 16/11/07</fin>
          <predecesores>8,9</predecesores>
        </tarea>
        <tarea>
      <numero>14</numero>
          <nombre>Control de calidad del manual de usuario</nombre>
      <duracion>4 dias</duracion>
```

```
        <inicio>lunes 19/11/07</inicio>
            <fin>viernes 23/11/07</fin>
            <predecesores>8,9,10,13</predecesores>
          </tarea>
</manual_usuario>
<programacion>
  <tarea>
        <numero>15</numero>
            <nombre>Programacion modulo de administracion</nombre>
        <duracion>22 dias</duracion>
        <inicio>lunes 02/07/07</inicio>
            <fin>martes 24/07/07</fin>
            <predecesores>8,9</predecesores>
          </tarea>
          <tarea>
        <numero>16</numero>
            <nombre>Programacion modulos de otros usuarios</nombre>
        <duracion>30 dias</duracion>
        <inicio>miercoles 15/08/07</inicio>
            <fin>viernes 14/09/07</fin>
            <predecesores>8,9,15</predecesores>
          </tarea>
          <tarea>
        <numero>17</numero>
            <nombre>Programacion del standalone</nombre>
        <duracion>44 dias</duracion>
        <inicio>lunes 17/09/07</inicio>
            <fin>miercoles 31/10/07</fin>
            <predecesores>8,9,16</predecesores>
          </tarea>
```

```
    <tarea>
        <numero>18</numero>
            <nombre>Programacion nuevos requisitos</nombre>
        <duracion>15 dias</duracion>
        <inicio>jueves 01/11/07</inicio>
            <fin>viernes 16/11/07</fin>
            <predecesores>8,9,17</predecesores>
          </tarea>
          <tarea>
        <numero>19</numero>
            <nombre>Control de calidad programacion</nombre>
        <duracion>4 dias</duracion>
        <inicio>lunes 19/11/07</inicio>
            <fin>viernes 23/11/07</fin>
            <predecesores>8,9,10,18</predecesores>
          </tarea>
          <tarea>
        <numero>20</numero>
            <nombre>Pruebas funcionales</nombre>
        <duracion>11 dias</duracion>
        <inicio>lunes 26/11/07</inicio>
            <fin>viernes 07/12/07</fin>
            <predecesores>8,9,10,11,19</predecesores>
          </tarea>
</programacion>
<aprobacion_cliente>
    <tarea>
        <numero>21</numero>
            <nombre>Presentacion a expertos</nombre>
        <duracion>11 dias</duracion>
```

```
        <inicio>lunes 10/12/07</inicio>
            <fin>viernes 21/12/07</fin>
            <predecesores>15,16,17,18,19,20</predecesores>
          </tarea>
          <tarea>
        <numero>22</numero>
            <nombre>Adecuaciones</nombre>
        <duracion>11 dias</duracion>
        <inicio>lunes 24/12/07</inicio>
            <fin>viernes 04/01/08</fin>
            <predecesores>15,16,17,18,19,20,21</predecesores>
          </tarea>
          <tarea>
        <numero>23</numero>
            <nombre>Desarrollo de la interfaz de impresion</nombre>
        <duracion>41 dias</duracion>
        <inicio>martes 08/01/08</inicio>
            <fin>lunes 18/02/08</fin>
            <predecesores>15,16,17,18,19,20</predecesores>
          </tarea>
          <tarea>
        <numero>24</numero>
            <nombre>Prueba piloto para el desarrollo de la interfaz de comunicacion con el hasrdware de impresion en una planta</nombre>
        <duracion>83 dias</duracion>
        <inicio>martes 19/02/08</inicio>
            <fin>lunes 12/05/08</fin>
            <predecesores>15,16,17,18,19,20,23</predecesores>
          </tarea>
          <tarea>
```

```
        <numero>25</numero>
            <nombre>Pruebas de aceptacion por el cliente</nombre>
        <duracion>6 dias</duracion>
        <inicio>martes 13/05/08</inicio>
            <fin>lunes 19/05/08</fin>
            <predecesores>15,16,17,18,19,20,24</predecesores>
          </tarea>
  </aprobacion_cliente>

</Cronograma>
```

GLOSARIO DE TÉRMINOS

ACO: Ant Colony System o Optimización basada en Colonia de Hormigas.

AS: Ant System o Sistema de Hormigas.

ASC: Ant Colony System o Sistema de Colonia de Hormigas.

BWACS: Best-Worst Ant Colony System o Sistema de Colonias de la Mejor-Peor Hormiga.

BWAS: Best-Worst Ant System o Sistema de la Mejor-Peor Hormiga.

Ciclo Hamiltoniano: Es un ciclo simple que visita todos los vértices.

HPSO: Versión hibridizada de la Optimización basada en Nubes de Partículas.

HPSO-kn: Revela la mejor acción de la Optimización basada en Nubes de Partículas.

HPSOvecin: Versión de la Optimización basada en Nubes de Partículas que mejoró la performance a través de la utilización de vecindarios lógicos.

JSP: Job-Shop Scheduling o Problema de Secuenciación de Tareas

Metaheurística: Algoritmo aproximados de propósito general consistente en procedimientos iterativos que guían una heurística subordinada combinando de forma inteligente distintos conceptos para explorar y explotar adecuadamente el espacio de búsqueda.

MMAS: Max-Min Ant System o Sistema de Hormigas Max-Min.

PSO: Particle Swarm Optimization o Optimización basada en Nubes de Partículas.

QAP: Quadratic Assignation Problem o Problema de Asignación Cuadrática

SDS: Stochastic Diffusion Search ó Búsqueda por Difusión Estocástica.

SI: Swarm Intelligence o Inteligencia Colectiva o Inteligencia de Enjambre.

TSP: Traveling Salesman Problem o como se conoce en español, Problema del Viajante de Comercio o Viajante Vendedor.