Pontificia Universidad Católica de Valparaíso

Facultad de Ingeniería

Escuela de Ingeniería Informática

RESOLUCIÓN DE PROBLEMAS DE COBERTURA DE CONJUNTOS UTILIZANDO MÚLTIPLES COLONIAS DE HORMIGAS EN PARALELO

Autor:

Luis Alberto Castillo Flores

Informe final del Proyecto para optar al Título profesional de

Ingeniero Civil en Informática

Profesor guía:

Broderick Crawford Labrín

Profesor Co-referente:

Nibaldo Rodríguez Agurto

Julio 2007

A todos aquellos que han participado
en el presente proyecto profesional y personal,
en especial a mi esposa y mis padres
por su amor y apoyo constante.

Glosario de Términos

Ant Colony Optimization : Ant colony optimization (ACO) es una Metaheurística que se basa en comportamiento de colonias de hormigas, puede ser usado para encontrar soluciones aproximadas a problemas de optimización complejos.

Ant Colony System : Es un algoritmo que pertenece a la familia de algoritmos ACO, el cual presenta la primera mejora principal sobre el algoritmo base de hormigas, se diferencia principalmente en la elección pseudo randómica del siguiente elemento a ser agregado a la solución.

Ant Q : Es un algoritmo que pertenece a la familia de algoritmos ACO, el cual presenta similitudes con procesos de Q-learning.

Ant System : Es un algoritmo que pertenece a la familia de algoritmos ACO, define la primera estructura de funcionamiento de colonias de hormigas.

Asíncrona : Tipo de transmisión en la cual no hay ninguna relación temporal entre el envío de información desde el emisor a el receptor.

Benchmark : Es la disponibilidad de resultados de la ejecución de un programa informático o un conjunto de programas para la resolución de un problema, con el objetivo de estimar el

rendimiento de un nuevo desarrollo sobre un problema ya resuelto y así poder comparar los resultados.

Best Worst Ant System : Es un algoritmo que pertenece a la familia de algoritmos ACO, tiene como características la penalización de las peores soluciones generadas y reinicialización de valores al momento de detectar estancamiento.

Colonias : Referente a la simulación del comportamiento en poblaciones de hormigas, están determinadas por la precencia de software que simula el comportamiento de múltiples hormigas.

Esclava : Se refiere a colonias de hormigas que no tienen la característica de administrar procesos globales, sino que dependen de un administrador.

Feromona : Se refiere a la simulación de sustancias químicas secretadas por las hormigas con el fin de provocar un comportamiento de atracción a un objetivo. Son un medio de señales cuyas principales ventajas son el gran alcance y la evitación de obstáculos, puesto que sufren de proceso de evaporación.

Files : Archivo en inglés, se refiere a carpetas y archivos dentro de un PC, el cual es de importancia configurar su acceso, ya que en la carpeta de dicho nombre se dejan accesibles archivos necesarios para el proceso.

Firewall : O cortafuegos, es un elemento de hardware o software utilizado en una red de computadores para controlar las comunicaciones, permitiéndolas o prohibiéndolas según las políticas de red que se haya definido.

Heurística : Se trata de métodos exploratorios durante la resolución de problemas en los cuales las soluciones se descubren por la evaluación del progreso logrado en la búsqueda de un resultado final.

Hibridación : Es el proceso de mezclar diferentes especies o variedades de programas o software para crear uno nuevo con mejores características.

Iteraciones : Repeticiones consecutivas de un proceso.

Knapsack Problem : O problema de la mochila, es un problema de optimización combinatorial. La idea es maximizar el uso de un contenedor de elementos, considerando un conjunto de elementos, cada uno con un costo y un valor, tratando de determinar que elementos incluir que tenga el mínimo costo y el mayor valor posible.

Localhost : Es un nombre reservado que tienen todos los dispositivos que dispongan de una tarjeta de red ethernet para referirse a sí mismo. El nombre localhost es traducido como la dirección IP de loopback 127.0.0.1

Max-Min Ant System : Es un algoritmo que pertenece a la familia de algoritmos ACO, tiene como características la restricción entre valores mínimos y máximos de carga de feromona.

Metaheurística : Es un método heurístico para resolver un tipo de problema de optimización combinatorial, usando los parámetros dados por el usuario sobre unos procedimientos genéricos y abstractos de una manera que se espera eficiente. Generalmente se aplican a problemas que no tienen un algoritmo o heurística específica que dé una solución satisfactoria o no es posible de implementar.

Migración : Se denomina migración a todo desplazamiento de población que se produce desde un lugar de origen a otro destino, en este caso se refiere a las hormigas que llevan su solución de una colonia a otra.

Mozart-Oz : Es una plataforma de desarrollo avanzada para usos inteligentes y distribuidos. Tiene un ambiente de desarrollo interactivo incremental. Es ideal tanto para usos distribuidos de uso general así como para problemas difíciles que requieren la optimización sofisticada como para capacidades de inferencia.

Paralelización : Es la ejecución de un algoritmo por partes en un mismo instante de tiempo por varias unidades de procesamiento, para finalmente unir todas las partes y obtener el resultado final.

Particionado : Es el proceso de dividir un espacio o conjunto en dos o más conjuntos disjuntos.

Rank based Ant System : Es un algoritmo que pertenece a la familia de algoritmos ACO, tiene como características la existencia de hormigas elitistas.

Set Covering Problem : Problema de Cobertura de Conjuntos, problema de optimización combinatorial, que a partir de un conjunto de posibles soluciones dadas, se debe seleccionar un mínimo de conjuntos que contengan todos los elementos del conjunto total de elementos inicial.

Set Partitioning Problem : Problema de Particionamiento de Conjuntos, problema de optimización combinatorial, considerando una colección de los subconjuntos base con costos asociados, se divide el problema para encontrar una partición de mínimo costo.

Síncrona : Tipo de transmisión que consiste en el envío de datos en un momento dado del proceso, el cual esta coordinado mediante etapas o tiempos de reloj.

Socket : Un socket es un método para la comunicación entre un programa del cliente y un programa servidor en una red. Un socket se define como el punto final en una conexión. Es una dirección de Internet, combinando una dirección IP y un número de puerto.

Subset Selection Problem : Problema de Selección de Subconjuntos, problema combinatorial que se basa en dividir el total de posibles combinaciones de un problema en subconjuntos que cumplan la característica de cumplir con cierto objetivo.

Thread : O hilo, se considera como la agrupación de un trozo de programa que tiene su propia ejecución en un procesador.

Ticket : Elemento perteneciente al lenguaje Mozart-Oz, el cual permite la comunicación entre distintas aplicaciones de manera remota, en el cual se especifican entre otras cosas la dirección IP y puerto dedicado para la comunicación.

Vecindad : Se refiere a la cercanía lógica utilizada para la comunicación directa entre colonias de hormigas, en el cual se especifica que colonias se encuentran cercanas de manera lógica; de manera física pueden estar a miles de kilómetros.

Lista de Abreviaturas o Siglas

ACO : Ant Colony Optimization

ACS : Ant Colony System

Ant-Q : Ant Q-learning

AS : Ant System

ASrank : Rank based Ant System

BWAS : Best Worst Ant System

GA : Genetic Algorithm

IP : Internet Protocol

KP : Knapsack Problem

LR : Reinforcement Learning

MMAS : Max-Min Ant System

PC : Personal Computer

PSO : Particle Swarm Optimization

Q-l : Q-learning

SA : Simulated Anealing

SCP : Set Covering Problem

SPP : Set Partitioning Problem

SSP : Subset Selection Problem

TCP : Transmission Control Protocol

TSP : Traveling Salesman Problem

Simbología

$f(x)$: Función.

x : Solución a problema combinatorial.

X: Conjunto de posibles soluciones.

Ω : Conjunto de todas las posibles combinaciones.

S : Conjunto de objetos.

$S_{posible}$: Conjunto que contiene a todos los subconjuntos posibles de S.

a_{ij} : Matriz donde se representa el problema.

$\varnothing$: Conjunto vacío.

τ : Carga de feromona.

η : Información Heurística.

P: Probabilidad.

ρ : Parámetro Ro.

Resumen

Se presenta un estudio y desarrollo de un software para problemas de optimización combinatorial, específicamente se trata de resolver problemas de cobertura de conjuntos.

Se indica el uso, modificación y desarrollo de estrategias y herramientas efectivas para encontrar soluciones lo suficientemente buenas y rápidas. La estrategia de desarrollo elegida se basa en la utilización del paralelismo para mejorar las prestaciones colaborativas de la Metaheurística ACO. El paralelismo es la elección para el potenciamiento de las características colaborativas que se encuentran presentes en los algoritmos ACO, los procesos son llevados a cabo de manera incremental y colaborativa. Los procesos de comunicación son desarrollados de manera intensiva tanto de modo externo donde se comunican soluciones

parciales entre las colonias de hormigas y de manera interna donde cada hormiga se ve influenciada por las búsquedas desarrolladas por las otras hormigas de su propia colonia.

El objetivo de este proceso de búsqueda colaborativa, incremental y paralelo es poder obtener los mejores resultados asegurando un buen nivel de calidad, en el menor número de experimentos y en el menor tiempo posible. Se presentan desarrollos prácticos de aplicaciones y sus respectivos experimentos. En los cuales se deja constancia de la utilidad de este tipo de sistemas, los cuales además de permitir el aseguramiento de obtener un buen resultado, permite que en un tiempo mucho menor se realicen experimentos con todas las combinaciones de valores posibles en los parámetros involucrados en los algoritmos. Se logra un menor tiempo de búsqueda, se asegura la obtención del mejor resultado por parte del algoritmo y se minimiza la cantidad de experimentos necesarios para los logros anteriores.

Palabras Claves: ACO, Optimización Combinatorial, Paralelismo.

Abstract

This document presents the study and development of software for solving combinatorial optimization problems, specifically the Set Covering Problem.

It is indicated the use, modification and development of strategies and effective tools to find solutions sufficiently good and quick. The strategy of development chosen is based on the utilization of the parallelism to improve the collaborative characteristics of ACO metaheuristic. The parallelism is the election to foster the collaborative characteristics present in the ACO algorithms, the processes are carried out in an incremental and collaborative way. The communication processes are developed in an intensive way, in an external perspective, where they communicate partial solutions between the ants colonies and of an internal perspective where every ant is influenced by the searches developed (paths) by other ants in her colony.

The goal of this collaborative, incremental and parallel search process is to be able to obtain the best results assuring a good level of quality, with the minor number of experiments and time. The applications and its experiments show the utility of this type of solver, assuring the obtention of good results, in a very much minor time for all the combinations of possible

values in the parameters involved in the algorithms. It is achieved a minor time of search, assuming to reach the best result by the algorithm and minimizing the quantity of experiments necessary for the previous achievements.

Keywords: ACO, Combinatorial Optimization, Parallelism.

Capítulo 1
Introducción

1.1 Introducción

El siguiente documento presenta el desarrollo completo de un proyecto destinado al estudio y aplicación de un sistema que sea capaz de resolver Problemas de Cobertura de Conjuntos, en inglés Set Covering Problem (SCP), mediante algoritmos ACO (Ant Colony Optimization) funcionando en Paralelo.

El tipo de algoritmo utilizado llamado ACO (Ant Colony Optimization), pertenece a una familia de algoritmos llamados bio-inspirados cuya característica principal es la imitación de procesos y comportamientos de la naturaleza, los cuales se tratan de aplicar para poder resolver problemas humanos complejos. Específicamente la inspiración de ACO radica en la imitación del comportamiento de una colonia de hormigas para la resolución del problema de conseguir alimentos, adaptándolo para que pueda ser aplicado a problemas combinatoriales complejos, otorgando respuestas de óptimos locales, es decir, entregar “buenas soluciones”.

En relación al tipo de problema combinatorial, específicamente se trata de desarrollar un proceso y respuesta al problema de Cobertura de Conjuntos (Set Covering Problem), que tiene como principal característica la resolución de problemas a partir de un subconjunto de elementos que deben cumplir ciertas restricciones y que tienen un costo asociado, de manera que al ser seleccionados los elementos sean cubiertas las restricciones y a partir de una matriz, donde se representan los recursos y necesidades, se deben elegir tales combinaciones que sean cubiertas por al menos una columna con el objetivo de reducir al mínimo el costo de las columnas usadas.

Un problema SCP al resolverlo mediante un algoritmo ACO requiere de algunas consideraciones para el desarrollo de su proceso y aplicación, específicamente durante las iteraciones en la selección sucesiva de columnas, cuyos elementos se deben agregar iterativamente a la solución inicialmente vacía, considerando que la elección del siguiente

elemento (columna) se realizará mediante una heurística que considere la conveniencia de la elección mediante la carga de feromona. Una vez construida o encontrada una solución posible (subconjunto) se debe realizar la evaluación de la misma para determinar su nivel de actualización de feromonas, tanto de manera local, global o distribuida. Cuyo proceso y características individuales se detallarán en el transcurso del presente documento.

Dada la naturaleza del proyecto al no tener una investigación ya desarrollada por algún otro autor, los procesos y el trabajo fueron realizados inicialmente mediante un proceso "*Exploratorio*", es decir, buscar en distintos conocimientos teoricos y prácticos, las características y potencialidades de utilización de distintos elementos.

Los principales elementos participantes, que resultan de interés investigar y desarrollar son los siguientes:

- Algoritmos ACO.
- Problemas de Optimización Combinatorial.
- Paralelismo.
- Estrategias de Comunicación en Paralelo.
- Estructuras de Posicionamiento y Comunicación entre procesos Paralelos.
- Lenguajes con prestaciones de paralelismo.

Debiéndose analizar y plantear las soluciones, en un diseño que abarque las siguientes posibles configuraciones:

- Búsqueda en Paralelo o Independiente: Analizar el resultado obtenido por colonias individuales o mediante colonias que actúan de manera paralela y colaborativa durante el desarrollo iterativo de la búsqueda de las soluciones.
- Problema a desarrollar: El problema se limitó al del tipo de cobertura de conjuntos SCP, pero permitiendo la elección de distintos archivos Benchmark a resolver.
- Comunicación Síncrona o Asíncrona: determinando su aplicabilidad y características de prestaciones de cada una.
- Número de Colonias de hormigas: Permitir que la búsqueda paralela sea configurable en relación a la cantidad de equipos que participarán en el proceso.

Tipo de algoritmo ACO: Existiendo distintos tipos y características de los mismos, en el diseño se plantea la posibilidad de utilizar principalmente ACS (Ant Colony System) como algoritmo base para el desarrollo, pero su configuración permite un comportamiento de variantes de algortimos.

Heurística: Dentro de los algoritmos ACO existe una característica que afecta de manera considerable el desarrollo y la cual también tiene algunas posibilidades de utilización de diferentes maneras, afectando el proceso de búsqueda y como se tomará en consideración la elección de una nueva columna.

Número de Hormigas: Cada colonia tendrá la posibilidad de trabajar con un número distinto de agentes (hormigas) en la búsqueda de las soluciones.

Estructura de comunicación: permitir que la forma como estarán distribuidas las colonias, les permita comunicarse de manera de tener un solo receptor de las mejores soluciones, es decir que todas colaboran en un solo núcleo de conocimiento (Colonias administrativas y buscadoras) o mediante un trabajo colaborativo trabajando con algunas vecindades (Colonia en vecindades).

Intervalo de migración de las soluciones: Permitir determinar cada cierto número de iteraciones se comunicarán las colonias.

Elección del comunicado: Poder determinar cual es la mejor forma de conseguir que la información comunicada provoque el mejoramiento del proceso de búsqueda de la solución en cada colonia.

Este análisis determinará la posibilidad de construcción o codificación de la plataforma de búsqueda, la elección de una herramienta que se adecúe y además permita el desarrollo práctico. Dicho producto software deberá presentar características de configuración del proceso y además de poder utilizar los recursos de una red de manera lo más simple posible, solo requiriendo tareas básicas de configuración (pre-proceso).

El paso final se refiere al proceso de pruebas sobre el producto, tratando de determinar los mejores resultados y poder inferir cuales son los mejores valores de parámetros para cada problema combinatorial.

1.2 Motivación Personal

La principal característica que radica en la elección del tema de trabajo de título, es apoyar el desarrollo de sistemas que generen soluciones concretas a problemas altamente complejos.

Existiendo el especial interés del desarrollo de aplicaciones de algoritmos ACO aplicados a resolver problemas de selección de Subconjuntos, en ingles Subset Selection Problem (SSP), como (Set Packing, Set Partitioning, Set Covering, entre otros) específicamente la bibliografía reciente se ha abocado principalmente a utilizar algoritmos ACO en problemas de ordenamiento, tales como el problema del vendedor viajero y el problema de asignación cuadrática.

Un aspecto que le otorga mayor singularidad al proyecto, es la búsqueda de mejorar las prestaciones de la resolución de problemas SCP mediante algoritmos ACO utilizando procesos distribuidos y paralelos, que consisten en utilizar procesos capaces de trabajar cooperativamente para resolver un mismo problema. El paralelismo se aplicará mediante procesos colaborativos distribuidos, que pretende abaratar costos y hacer mucho más aplicable a diversos problemas.

La justificación para el uso del paralelismo en ACO, se debe a su característica de comportamiento al momento de resolver un problema, el cual se desarrolla por un agente (hormiga) quien individualmente sería incapaz de resolver un problema, pero la unión (colonias) entre los agentes les permite llevar a cabo procesos mucho mas complejos.

Utilizando la conversión de un algoritmo secuencial a un algoritmo paralelo se espera representar mejoras en el aumento de la capacidad de poder explorar espacios de búsquedas mayores y más complejos, reducción en los tiempos de ejecución y poder encontrar soluciones de mayor calidad.

1.3 Descripción del Trabajo

Dentro de este punto se dará a conocer la forma en que estará estructurada la información del documento.

Como todo proceso de desarrollo la primera etapa consiste en llevar a cabo una exhaustiva investigación de los distintos elementos participantes en el proyecto y principalmente de sus posibilidades y restricciones de utilización en conjunto.

En primer lugar se expondrá el análisis correspondiente de cada uno de los elementos participantes en el proceso, cuyo resultado es la propuesta de variadas alternativas de combinaciones entre los mismos. El problema a enfrentar, donde las alternativas serán: problema se cobertura de conjuntos (Set Covering Problem, SCP), problema de particionamiento de conjuntos (Set Partitioning Problem, SPP) y problema de la mochila (Knapsack Problem, KP). El segundo elemento encontrado corresponde a el/los algoritmos pertenecientes a la Metaheurística ACO. El tercer elemento es la arquitectura de paralelización, donde las alternativas son: Administración Central y Vecindades. Un cuarto elemento que corresponde a la estrategia de comunicación, donde las alternativas son: Síncrona y Asíncrona.

Sobre las consideraciones realizadas en la etapa de análisis, se llevará a cabo la construcción de distintos diseños. Donde se integrarán todas las variables antes estudiadas y prever algún tipo de aplicación funcional.

La etapa siguiente corresponde a la codificación y desarrollo de software sobre las funcionalidades generadas en los diseños propuestos.

Una vez terminado algún producto se realizará una explicación sobre los desarrollos funcionales, comentando sus virtudes y defectos, principalmente mediante etapas correspondientes a las pruebas.

Para terminar el proceso de desarrollo, se mostrarán los resultados obtenidos, indicando las mejoras realizadas al proceso y consideraciones finales.

Capítulo 2

Objetivos

2.1 Objetivo General

Desarrollar un software que permita implementar en paralelo un algoritmo ACO con el objetivo de resolver problemas combinatoriales del tipo SCP.

2.2 Objetivos Específicos

Los objetivos específicos que se desglosan desde el objetivo general son:

Estudiar los tipos de algoritmos ACO.
Estudiar los problemas combinatoriales del tipo de Selección de Sub-Conjuntos.
Seleccionar el Algoritmo ACO y Problema Combinatorial que se presenten con mayores potencialidades.
Estudiar el desempeño y aplicabilidad de los algoritmos ACO en la resolución de problemas combinatoriales.
Estudiar la aplicación de características paralelas existentes en los algoritmos ACO a nivel general entre colonias.
Estudiar las estrategias de comunicación para compartir soluciones entre los algoritmos ACO.
Estudiar la aplicabilidad de los algoritmos implementados en paralelo para resolver instancias benchmark de problemas SCP.
Desarrollar un software configurable que genere procesos de algoritmos ACO funcionando en paralelo y de manera distribuida.
Probar y analizar los resultados del software sobre instancias benchmark.

Capítulo 3
Estado del Arte

Para el desarrollo de cualquier solución siempre es necesario entender el ámbito en el cual se encuentra, tratando de poder cuantificar y calificar cada uno de los elementos que forman parte del sistema, además de considerar aquellos aspectos que influyen desde el ambiente y que restringen o posibilitan cada uno de los pasos del desarrollo.

El sistema lo podemos incluir en el ámbito de la resolución de problemas combinatoriales, los cuales requieren de herramientas que sean capaces de obtener los mejores resultados y en un tiempo razonable.

A continuación se presentan cada uno de los elementos participantes en el sistema, explicando características apropiadas para el desarrollo y otras que requieran de alguna adecuación o simplemente no sean prometedoras para continuar desarrollandolas.

3.1 Problemas de Optimización

El primer análisis se desarrolla sobre el objetivo final, es decir el problema que queremos solucionar, específicamente un problema de optimización combinatorial.

Un problema de optimización, se puede interpretar como la situación cuando nos enfrentamos a un problema cuantificable o medible, y lo que se busca es encontrar una configuración óptima entre un número finito de elementos., los cuales pueden ser configurados, agrupados, seleccionados o distribuidos de manera de satisfacer una serie de requerimientos y/o restricciones; con el objetivo de encontrar, si existen, la mayoría o totalidad de las posibles soluciones y poder determinar cual de ellas es la mejor, de acuerdo a cierto criterio.[35]

El proceso de búsqueda para encontrar la(s) solución(es) que minimice(n) o maximice(n) una función objetivo, dependerá del tipo de problema combinatorial, debido a que no existe un único tipo de problema o una única forma de determinar que es lo que se busca, cada posible

explicación encontrada en las publicaciones hasta el momento ha servido para poder evaluar que tipo de problema es de interés y prometedor realizar.

Formalmente una resolución de un problema combinatorial, se puede representar como: [46]

- *Optimizar f(x)*
- Sujeto a: $x \in X \subset \Omega$

Donde *x* es una solución al problema combinatorial que esta siendo resuelto. *f(x)* es la función objetivo, X es el conjunto de posibles soluciones y Ω es el conjunto de todas las posibles combinaciones de todos los componentes involucrados en el problema. Que en otras palabras nos indica que debemos encontrar una solución a la optimización mínima o máxima de una función en base a un conjunto de soluciones posibles dentro de una cantidad inmensa de posibles combinaciones.

La limitante del tamaño del problema y sus respectivos flujos de datos, produce que cuando se trata de solucionar o planificar un proceso, la cantidad de posibles soluciones sea enorme, lo que naturalmente requerirá de bastante consumo de tiempo y recursos. Lo cual lo ha convertido a los problemas combinatoriales en un problema casi intratable computacionalmente en problemas de gran tamaño.

Existen dos formas generales de resolución de los problemas combinatoriales, las cuales corresponden a técnicas completas y a técnicas incompletas.

La primera lo que busca son los óptimos globales, es decir la mejor solución que tenga el problema; la segunda técnica corresponde a encontrar óptimos locales, es decir encontrar una solución suficientemente buena, que permita sin la necesidad de encontrar la mejor, poder encontrar una solución que tenga la calidad suficiente para ser útil y que demás se pueda encontrar en un tiempo razonable.

En el área de las técnicas incompletas, es que se desenvuelve el presente desarrollo, ya que la factibilidad de encontrar soluciones “suficientemente buenas” en un tiempo de computación razonable es un proceso interesante y posible de aplicar. Las características que se encuentran dentro de estas técnicas están principalmente ligadas al grado de especificación del problema y como se enfrentará la búsqueda de la solución, la cual estará desarrollada mediante la

Metaheurística ACO (Ant Colony Optimization), lo cual será explicado en detalle en capítulos posteriores del documento.

En relación a que tipo de problema de optimización se resolverá, es preciso señalar que existen varias opciones y características documentadas, pero la elección se refiere a problemas de selección de subconjuntos en un universo de combinaciones complejas.

3.1.1 Problema de Selección de Subconjuntos, Subset Selection Problem (SSP)

Problema de optimización combinatorial, el cual a modo general consiste en encontrar, dentro de un conjunto de posibles soluciones y algunas restricciones, un óptimo factible en un subconjunto de las posibles soluciones generales para una función objetivo.

Formalmente un problema de selección de subconjuntos [35], se podría definir como (S, $S_{posible}$, f):

- S un conjunto de objetos.
- $S_{posible} \subseteq P(S)$ es un conjunto que contiene todos los subconjuntos posibles de S.
- $f : S_{posible} \rightarrow \mathrm{R}$ es una función objetivo que asocia el valor real del costo $f(S`)$ con cada posible subconjunto de objetos $S`$ perteneciente a $S_{posible}$.
- El objetivo es encontrar $S^* \subseteq S$ tal que $S^* \in S_{posible}$ y $f(S^*)$ sea el máximo o mínimo.

Donde S se refiere al conjunto que contiene el total de elementos pertenecientes al universo del problema, lo que provoca encontrar soluciones posibles $S_{posible}$, dentro de todos los subconjuntos posibles de encontrar dentro de S $P(S)$. La manera de determinar el resultado de la búsqueda es mediante una función f que esta determinada por el costo de cada subconjunto posible dentro de $S_{posible}$.

Muchos problemas combinatoriales bien conocidos son miembros de esta clase, los cuales han sido siempre motivo de interés desarrollar por diversos investigadores, por ejemplo problemas de cobertura de conjuntos (SCP: Set Covering Problem), problemas de particionamiento (SPP: Set Partitioning Problem) y el problema de la mochila (KP: Knapsack Problem).

Debido a la existencia de los tipos de problemas antes mencionados, es que a continuación se realizará un análisis detallado de cada uno de ellos, mostrando sus características como potencialidades.

3.1.1.1 Problema de Cobertura de Conjuntos, Set Covering Problem (SCP)

Dentro de los problemas de subconjuntos, se encuentra el problema de Cobertura de Conjuntos (SCP), el cual tiene una gran cantidad de aplicaciones prácticas en la vida real, principalmente en lo que se refiere al uso correcto de los recursos de una organización, tanto a recursos materiales como de personas.

Este tipo de resolución de problemas [4] funciona a partir de un subconjunto de elementos que deben cumplir ciertas restricciones y tienen un costo asociado. De manera que al ser seleccionados los elementos sean cubiertas las restricciones y a partir de una matriz donde se representan los recursos y necesidades a resolver por el problema, se deben elegir tales combinaciones que sean cubiertas por al menos una columna con el objetivo de reducir al mínimo el costo de las columnas usadas [14,20].

La matriz *A* es representada por *N={1…n}* que corresponde al índice del conjunto de columnas y *M={1...m}* que corresponde al índice del conjunto de filas. Uno se encuentra además con un vector de costos *c* , donde cada componente c_j del vector corresponde al costo que tiene seleccionar la columna *j* de la matriz. Se dice que la fila *i* es cubierta por la columna *j*, cuando es igual a 1.

Matemáticamente, el SCP se puede expresar como sigue:

$$\text{Minimizar:} \quad z = \sum_{j=1}^{n} c_j x_j \qquad (3.1)$$

$$\text{Sujeto a:} \quad \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad \text{para} \quad i = 1,\ldots,m \qquad (3.2)$$

$$x_j = 0 \text{ o } 1 \quad \text{para} \quad j = 1,\ldots,n \qquad (3.3)$$

$$\text{Donde} \quad x_j = \begin{cases} 1 & \text{Si j está en la cobertura.} \\ 0 & \text{En caso Contrario.} \end{cases}$$

$$a_{ij} = \begin{cases} 1 & \text{Si i} \in P_j \\ 0 & \text{En caso Contrario.} \end{cases}$$

3.1.1.2 Problema de Particionamiento de Conjuntos, Set Partitioning Problem (SPP)

Dentro de los problemas de subconjuntos, el problema de Particionamiento de Conjuntos (SPP) ha sido uno de los menos desarrollados, este tipo de resolución de problema funciona a partir de un subconjunto de elementos que deben cumplir ciertas restricciones y tienen un costo asociado, de manera que al ser seleccionados los elementos particionan las restricciones [32] y a partir de una matriz, donde se representan los recursos y necesidades a resolver por el problema, en la cual se deben elegir tales combinaciones que son cubiertas exactamente por una columna con el objetivo de reducir al mínimo el costo de las columnas usadas.[30]

Matemáticamente, el SPP se puede expresar como sigue [30]:

Minimizar: $$z = \sum_{j=1}^{n} c_j x_j \qquad (3.4)$$

Sujeto a: $$\sum_{j=1}^{n} a_{ij} x_j = 1 \quad \text{for } i = 1,\ldots,m \qquad (3.5)$$

$x_j = 0$ o 1 for $j = 1,\ldots,n$ (3.6)

Donde
$$x_j = \begin{cases} 1 & \text{Si j está en la cobertura.} \\ 0 & \text{En caso Contrario.} \end{cases}$$
$$a_{ij} = \begin{cases} 1 & \text{Si } i \in P_j \\ 0 & \text{En caso Contrario.} \end{cases}$$

Existe además otra notación [32] que le da formalmente el nombre de particionamiento:

Dado
$I = \{1,\ldots,m\}$ conjunto de filas.
$J = \{1,\ldots,n\}1,\ldots,n\}$ conjunto de columnas.
$P = [P_1,\ldots,Pn]$

Donde: $P_j = \{i \in I \mid a_{ij} = 1\}$

$j \in J$
P_j es el conjunto de índices de filas que tienen una en la j-esima columna.
$| P_j |$ es la cardinalidad de P_j

Un conjunto $J^* \subseteq J$ es llamada una partición si:

$$\bigcup_{j \in J^*} P_j = I \qquad (3.7)$$

$$j,k \in J^*, j \neq k \Rightarrow P_j \cap P_k = \phi \qquad (3.8)$$

Asociado con cada partición J^* es un costo dado por: $\sum_{j \in J} c_j$

El objetivo mediante esta formulación es encontrar la partición con el mínimo costo.

3.1.1.3 Problema de la Mochila, Knapsack Problem (KP)

El problema de la mochila (KP: Knapsack Problem) es el tercero de los problemas de selección de subconjuntos estudiados, en general su funcionamiento es bastante parecido, pero lo que se busca es maximizar el uso de "la mochila", por lo tanto un elemento a considerar es la capacidad de la mochila denominado *K*. Otras diferencias se presentan en la fórmula al cambiar el objetivo por maximizar y la condición para expresar que pueden existir o no alguna columna que cubra la/s filas. [9, 23]

Matemáticamente, el KP se puede expresar como sigue, observando la pequeña diferencia del cambio entre *igualdad* (SPP) o *mayor igual* (SCP) por *menor igual* a la capacidad de la "mochila":

Maximizar: $$z = \sum_{j=1}^{n} c_j x_j \qquad (3.9)$$

Sujeto a: $$\sum_{j=1}^{n} a_{ij} x_j \leq K \qquad \text{for } i = 1,\ldots,m \qquad (3.10)$$

$$x_j = 0 \text{ o } 1 \qquad \text{for } j = 1,\ldots,n \qquad (3.11)$$

Cabe destacar que este problema puede ser aplicado a diversos problemas donde tenemos que maximizar la utilización del espacio disponible, como por ejemplo en problemas cargas de ciertos medios de transporte de mercancías.

3.2 Metaheurística Ant Colony Optimization (ACO)

Tal como se mencionó anteriormente, existen técnicas que buscan o mejor dicho son utilizadas para resolver problemas combinatoriales difíciles. La metaheurística ACO corresponde a una técnica incompleta que busca encontrar óptimos locales o como se mencionó "soluciones suficientemente buenas".

La metaheurística ACO, que se compone de una familia de algoritmos llamados ACO (Ant Colony Optimization) [19], se encuentra dentro de una serie de meta heurísticas inspiradas en

criaturas naturales, que como su sigla en inglés lo indica, es la imitación del comportamiento de una colonia de hormigas para la resolución de problemas. Su funcionamiento es de manera constructiva al ir generando las soluciones a partir de iteraciones consecutivas y comunicación de sus procesos y etapas, lo que les ha permitido ser utilizadas en la resolución de varios tipos de problemas [37,38].

Las características de este tipo de Metaheurísticas, es la capacidad de poder imitar un comportamiento colaborativo, donde la resolución de algún problema por un agente (hormiga) individualmente sería casi imposible, pero la unión (colonias) entre los agentes les permite llevar a cabo procesos mucho mas complejos.

Entre las características adoptadas de la vida real es la utilización de un elemento “feromona” que le permita a cada individuo de la colonia ir aprendiendo de la experiencia de procesos anteriores, específicamente, al igual que las hormigas reales los agentes “depositan” feromona en la tierra, influyendo en las decisiones de los otros agentes al haber mayor cantidad de feromona en un camino, mayor probabilidad de elegirlo [18,20]. Un aspecto relacionado a este elemento adoptado de la vida real es el proceso natural de evaporación que tiene la feromona, el cual consiste en que gradualmente la feromona depositada en la tierra ira disminuyendo en el tiempo. En el caso de los problemas combinatoriales tanto la carga como la evaporación se realiza sobre los elementos que constituyen el conjunto de posibles soluciones del problema.

3.2.1 Algoritmos Ant Colony Optimization (ACO)

Un algoritmo ACO es básicamente un sistema multiagente, donde el nivel de interacción entre los agentes (hormigas artificiales) deriva en un comportamiento sistémico complejo.

Un algoritmo ACO funciona de manera constructiva y colaborativa; a través de la utilización de hormigas, que mediante una serie de iteraciones en las cuales van entregando información mediante rastros (feromonas) permitiendo la resolución de problemas, que en la vida real, se aplica en la búsqueda de comida y encontrar el camino más corto para acceder a ella.

La metaheurística ACO proporciona estrategias efectivas para encontrar soluciones (óptimos locales) a los problemas combinatoriales de optimización [1], pero la relación tiempo-calidad de la solución encontrada, no siempre es lo bastante satisfactoria, además no siempre es capaz

de encontrar efectivamente la solución al tener problemas de estancamiento o convergencia a valores no deseados, debido a este último problema es que se han propuesto diversas mejoras, como por ejemplo la hibridación con otras técnicas[12,13,14], específicamente en este desarrollo las mejoras se buscarán realizar a través de la paralelización, concepto que será expuesto posteriormente en el presente documento.

Entre los elementos característicos que es interesante mencionar, es que los algoritmos ACO utilizan el concepto de heurística para determinar cual es el siguiente paso a realizar durante el proceso, es decir cual es el elemento que se considerará para la construcción de la misma, como por ejemplo la mejor solución hasta el momento o el menor valor del siguiente elemento a sumar a la solución, entre otros; las heurísticas dependerán del tipo de problema y como se quiera resolver.

Existiendo distintas versiones del algoritmo ACO, es conveniente tener una visión general sobre el funcionamiento de cada uno de ellos, para lograr obtener una visión y evaluación sobre cual/es alternativa/s de desarrollo permitieran su un mejor funcionamiento orientado hacia la aplicación en paralelo y para resolver problemas combinatoriales de selección de subconjuntos.

A continuación se resumirán los principales algoritmos ACO, donde cronológicamente los primeros algoritmos los podemos encontrar descritos en [19] con Ant System (AS), al cual luego se le realizaron variaciones y nuevos desarrollos., que fueron tomando otros nombres, por ejemplo: Ant-Q[17,22], Ant Colony System (ACS)[16], Max-Min Ant System (MMAS)[36,39,40,41], Rank Based Ant System (ASrank)[7] y Best-Worst Ant System (BWAS)[11].

3.2.1.1 Ant System (AS)

El algoritmo Ant System (AS), publicado por Dorigo en 1991 [19], define una primera estructura de funcionamiento de un algoritmo que imita el comportamiento de las hormigas aplicado al problema del vendedor viajero (TSP), que consiste posicionar las hormigas en una ciudad inicial, a continuación se seguirían una serie de pasos para encontrar la ruta mas corta entre ciudades y dada la característica del problema poder además visitarlas todas, la distancia entre ciudades se denotó por d(i,j), la cual era conocida. El objetivo era reducir la distancia recorrida para visitar todas las ciudades.

La información heurística esta determinada por el inverso de las distancias entre ciudades, formulada como:

$$\eta(i,j) = \frac{1}{d(i,j)} \qquad (3.12)$$

La carga de feromona realizada sobre los caminos entre las ciudades *i* y *j*, se denota como: $\tau(i,j)$

Para que todas las ciudades fueran visitadas, el funcionamiento consistía en que cada iteración se iniciaba agregando una nueva ciudad de manera randómica para cada ruta de la hormiga, entonces en cada iteración la hormiga elegía la próxima ciudad de acuerdo a la siguiente probabilidad:

$$P_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{he\,posible_k(t)}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}} & \text{si } j \in \text{posibles}_{k}(t) \\ 0 & \text{en caso contrario} \end{cases} \qquad (3.13)$$

La actualización de feromona se realiza una vez que todas las hormigas hayan completado sus soluciones; el nuevo depósito de feromona se realiza en función de la calidad de la solución encontrada y se realiza un proceso de evaporación mediante un factor igual para cada camino.

En esta primera versión de AS, no contaba con varias características que luego se le fueron incorporando como son las acciones del demonio y cargas elitistas, donde las acciones del demonio depositan adicionalmente feromona en las aristas que pertenecen a la mejor solución encontrada hasta el momento (mejor global). Y donde la cantidad de feromona depositada, que depende de la calidad de la mejor solución global, se incrementa en un factor *e*, que se relaciona con el número de hormigas elitistas que se considerarán.

3.2.1.2 Ant Q

Tal como se mencionó en el punto anterior, a los algoritmos se le fueron realizando una serie de actualizaciones y nuevas versiones, es así que la versión de AS propuesta por Dorigo y Gambardella en 1995 [17,22], se le realizaron una serie de cambios que dieron forma al algoritmo llamado Ant-Q, dichos cambios tenían la característica de combinar el algoritmo

desarrollado anteriormente con el aprendizaje reforzado (LR: Reinforcement Learning) relacionado con el Q-Learning.

Este algoritmo defiere respecto al AS en dos cosas principalmente:

En primer lugar, utiliza una regla de selección diferente pseudo randómica proporcional, provocando que la elección de nueva ciudad *l* se base en:

$$l = \begin{cases} \arg\max_{l \in posible} \{\tau(S[t-1], l')^{\alpha} \cdot \eta(S[t-1], l')^{\beta}\} & \text{si } q \leq q_0 \\ j & \text{si no} \end{cases} \qquad (3.14)$$

La segunda diferencia corresponde a la forma en que es actualizada la feromona, donde el algoritmo Ant-Q incluye una actualización local del nivel de feromona. Donde mientras todas las hormigas forman parte del proceso de actualización local, el algoritmo permite que solo una solución encontrada, la mejor, sea utilizada para actualizar la carga de feromona global.

3.2.1.3 Ant Colony System (ACS)

Una nueva versión de un algoritmo ACO propuesta por Gambardella y Dorigo en 1996, el cual es una extensión de Ant-Q y que presenta algunas mejoras o cambios del algoritmo. [16,18]

La primera mejora se refiere a la regla de transición de estados, donde se ofrece un balance entre la exploración de nuevos caminos y la explotación del conocimiento acumulado a través del desarrollo de la solución hasta el momento. El cambio se produce cuando se compara “q ≤ q0”, dicha comparación es utilizada para determinar como será considerada la elección del siguiente elemento a ser incorporado a la solución. Con q > q0 se favorece más la exploración y se favorece la explotación en el caso contrario. Cuando q0 se aproxima al valor 1 se seleccionan sólo soluciones óptimas locales, sin embargo el óptimo local puede no corresponder al óptimo global. Cuando q0 esta cercano a 0 se debieran examinar todas las posibles soluciones locales. Ecuación 3.14.

La segunda mejora se encuentra en la regla de actualización global del feromonas, que permitirá actualizar la matriz de feromonas solo con la mejor solución encontrada hasta el

momento, provocando que en la próxima iteración, las hormigas sean motivadas a buscar combinaciones en la vecindad de la mejor solución encontrada anteriormente.

La tercera modificación se refiere a los cambios locales de trazados de feromona para favorecer la exploración, donde se permitirá la actualización local a todas las hormigas al terminar su construcción de la solución. Una consecuencia de esta estrategia es que las hormigas tienden a no converger a una misma solución en etapas tempranas de iteraciones de la colonia.

Además contiene otros aspectos interesantes mencionados en algunas publicaciones de desarrollo posterior, como por ejemplo la utilización de búsquedas locales.

Una característica que también es interesante mencionar es la utilización de una lista candidata para restringir la elección del próximo elemento a ser incorporado a la solución, la lista contiene los elementos preferidos a ser elegidos; en lugar de examinar todas las posibilidades desde el estado actual, se examinan primero aquellos elementos no seleccionados que se encuentren dentro de la lista de candidatos y luego los restantes. La lista de candidatos contiene los elementos más prometedores de selección. Claramente esta característica es más aplicable para ciertos problemas como por ejemplo para minimizar las distancias en un problema TSP (Traveling Salesman Problem) donde los elementos (ciudades) pudieran están ordenados de menor a mayor distancia y la búsqueda en la lista sería secuencial; para otros problemas combinatoriales tiene mayor dificultad poder determinar quienes pertenecerán a dicha lista candidata, la opción más común es la elección de aquellos elementos que presenten un menor o mayor costo para ser agregado a la solución.

3.2.1.4 Max-Min Ant System (MMAS)

A partir de los distintos desarrollos y pruebas realizadas, se pudo determinar un tipo de problema que tenían los algoritmos ACO en su funcionamiento, el cual consistía en el riesgo relativamente alto de que la carga de feromonas tienda rápidamente hacia una solución, la cual no necesariamente era la mejor, por lo tanto ese valor de convergencia hacia que las siguientes iteraciones tendieran a la misma solución. Este problema fue enfrentado por Stutzle y Hoos con publicaciones en 1996, 1998 y 2000 [36,39,40,41].

En dichos trabajos se publican las definiciones del Max-Min Ant System, el cual contiene ciertas modificaciones, donde la primera es la 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. Las soluciones que ofrecen las hormigas suelen ser mejoradas usando optimizadores locales antes de la actualización de feromona.

La segunda se refiere a que los valores posibles para los rastros de feromona están limitados al rango [τmin, τmax], lo que le provee al algoritmo la posibilidad de disminuir el estancamiento, al darle a cada conexión existente una probabilidad, aunque bastante pequeña, de ser escogida. Para poder incrementar la exploración de nuevas soluciones, el MMAS utiliza en ocasiones re-inicializaciones de los rastros de feromona.

Otra característica es que en vez de inicializar los rastros de feromona en una cantidad pequeña, permite inicializarlos en una estimación del máximo permitido para un rastro; lo que es un componente adicional de diversificación en el algoritmo, ya que al comienzo las diferencias relativas entre los rastros de feromona no serán muy notorios, lo que no ocurre cuando los rastros de feromona se inicializan en un valor muy pequeño, donde las variaciones de la carga de feromona se comienzan a tomar en cuenta rápidamente por los agentes participantes en las iteraciones posteriores.

La carga de feromona esta dada por:

$$\tau_{ij}(t+1) = \rho \cdot \tau_{ij}(t) + \Delta \tau_{ij}^{mejor} \qquad (3.15)$$

3.2.1.5 Rank-Based Ant System (ASrank)

Corresponde al desarrollo de una nueva versión de algoritmo ACO, mediante la extensión del algoritmo AS con hormigas elitistas, realizada por Bullnheimer, Hartl y Strauss publicada en 1999. [7]

Las características de utilizar hormigas elitistas, las cuales producen que la mejor solución de cada iteración sea reforzada por una carga de feromona, provoca que el algoritmo provea una mejor solución, pero sea menos efectiva cuando las diferencias entre las soluciones encontradas son pequeñas. La introducción de ranking ayuda en tal situación.

La incorporación del ordenamiento de las hormigas para realizar la actualización de feromona, conlleva las siguientes características:

Las *m* hormigas se ordenan de mejor a peor según la calidad de sus soluciones.

Las acciones del demonio depositan feromona en las soluciones por las que han sido seleccionadas las σ *-1* mejores hormigas (hormigas elitistas). La cantidad de feromona depositada depende directamente del orden de la hormiga y de la calidad de su solución.

Los elementos pertenecientes a las soluciones realizadas por la mejor hormiga global reciben una cantidad adicional de feromona que depende únicamente de la calidad de dicha solución. Este depósito de feromona se considera el más importante, de hecho, recibe un peso de σ.

Dichos cambios en la actualización, son representados de manera matemática de la siguiente forma:

$$\tau_{ij}(t+1) := \rho\tau_{ij}(t) + \Delta\tau_{ij} + \Delta\tau_{ij}^{*} \qquad (3.16)$$

Donde $$\Delta\tau_{ij} = \sum_{\mu=1}^{\sigma-1} \Delta\tau_{ij}^{\mu} \qquad (3.17)$$

y $$\Delta\tau_{ij}^{\mu} = \begin{cases} (\sigma-\mu)\dfrac{Q}{L_{\mu}} & \text{si la } \mu\text{-ésima mejor hormiga viaja por el camino (i, j)} \\ 0 & \text{si no} \end{cases}$$

y $$\Delta\tau_{ij}^{*} = \begin{cases} \sigma\dfrac{Q}{L_{\mu}} & \text{si el camino (i, j) es parte de la mejor solución} \\ 0 & \text{si no} \end{cases}$$

3.2.1.6 Best-Worst Ant System (BWAS)

Otra variación del algoritmo ACO corresponde a la presentada por Cordon, Fenandez, Viana y Herrera en el 2000 [11,12], cuya propuesta incluye la incorporación de técnicas de computación evolutiva de aprendizaje incremental basado en poblaciones (PBIL: Population-Based Incremental Learning). La característica que presenta esta versión, es la penalización de las peores soluciones generadas en cada iteración, mediante la reducción del nivel de feromona, mutación de los valores de feromonas y re-inicialización de dichos valores cuando un valor de estancamiento es detectado.

La actualización se efectúa de acuerdo a:

$$\tau_{ij} := (1-\rho)\cdot\tau_{ij} + \Delta\tau_{ij} \qquad (3.18)$$

$$\text{Donde} \qquad \Delta\tau_{ij} = \begin{cases} f(C(L_{MejorTouGlobal})) & \text{si camino } (i, j) \in L_{MejorTouGlobal} \\ 0 & \text{si no} \end{cases}$$

Además existe una penalización mediante la evaporación de las iteraciones que no se encuentran en las mejores soluciones globales, de acuerdo a:

$$\tau_{ij} := (1-\rho)\cdot\tau_{ij} \qquad \forall(i,j) \in L_{tourPeor} \text{ y camino } (i, j) \notin L_{MejoresTourGlobal} \quad (3.19)$$

3.3 Paralelización

El tercer elemento participante en el desarrollo y que es necesario analizar, corresponde a la paralelización. Que consiste en llevar cabo un número determinado de procesos de manera simultánea en el tiempo, con el fin de resolver un problema en común [3].

Lo que hace interesante utilizar el concepto de paralelismo, es que se pueden esperar mejoras [45], en lo relacionado a la capacidad de poder explorar espacios de búsquedas mayores y más complejos, posibilidad de reducción en el tiempo total de ejecución y posible mejora en la calidad de las soluciones.

Sobre los espacios de búsqueda y su correspondiente complejidad, un problema combinatorial, como el que se quiere resolver en el presente desarrollo y sus respectivos flujos de datos al momento de la resolución, produce que cuando se trata de solucionar o planificar un proceso, la cantidad de posibles soluciones sea enorme, lo que naturalmente requerirá de bastante consumo de tiempo y recursos.

La característica planteada conlleva que al tratar de resolver problemas combinatoriales complejos hace que la capacidad de resolución que se pueda obtener a partir de un único proceso sea insuficiente, por esto es que existe la necesidad de incrementar dichas prestaciones, donde el procesamiento paralelo surge como la alternativa para satisfacer las características de velocidad y calidad que las aplicaciones requieren [43].

Un desarrollo en paralelo, es un conjunto de procesos capaces de trabajar cooperativamente para resolver un mismo problema computacional [21]. Claramente dentro del paralelismo aparecen distintas configuraciones posibles para generarlo, ya que se puede entender de diversas maneras:

Como un supercomputador con varios procesadores, capaz de administrar y desarrollar cálculos de manera paralela entre los distintos procesadores.

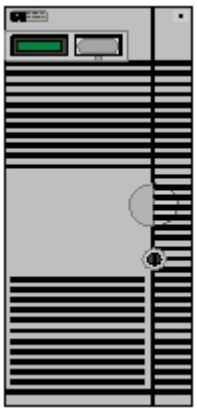


Ilustración 3.1: Imagen Computador Multi-Procesador.

Conjuntos de computadores personales distribuidos en una red limitada y/o conjuntos de computadores personales distribuidos en Internet, etc. Donde se presenta menor capacidad individual de procesamiento, pero se potencia la distribución y paralelización de procesos; permitiendo a un bajo costo aprovechar los recursos disponibles para resolver los problemas.

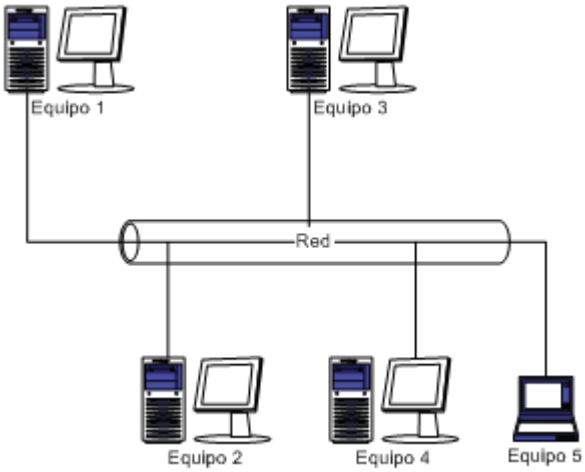


Ilustración 3.2: Red de Computadores.

Como tercera opción estaría en combinar las dos opciones anteriores, generando una red de supercomputadores, capaces de generar grandes cálculos de manera individual pero además potenciándolo mediante un trabajo en equipo y distribuido.

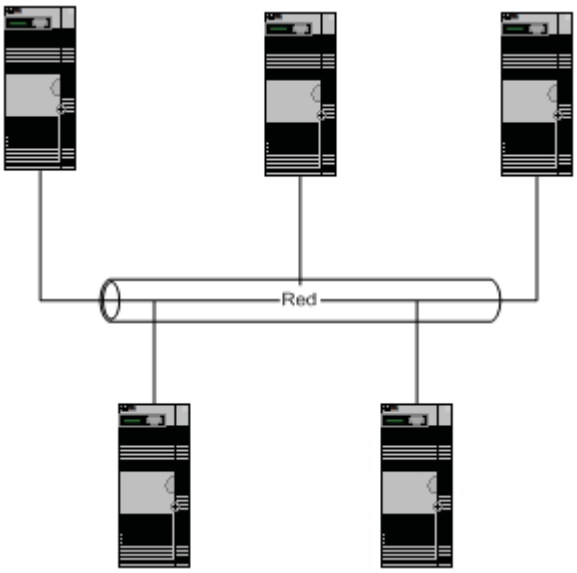


Ilustración 3.3: Red de Computadores Multi-Procesadores.

Cada una de las opciones antes mencionadas tiene características que podrían ser aprovechadas por el presente desarrollo, pero la opción claramente se debe limitar a recursos disponibles y real capacidad de operabilidad.

Debido a lo antes mencionado, es que la opción de utilizar recursos disponibles y de fácil acceso, determina que la utilización de Internet o redes locales de computadores personales sea la opción presente en el desarrollo. Donde los procesos realizados paralelamente en distintos equipos al mismo tiempo, ofrecen la posibilidad de concentrar recursos computacionales en la solución de un mismo problema [21] a bajo costo y con grandes potencialidades de capacidad y rapidez.

Pero existe un aspecto que debe tomarse en consideración y que también se refiere a las características de las opciones de paralelismo mencionadas, que se refiere a que no se puede esperar que aumente la velocidad y calidad de la solución proporcionalmente al número de procesos usados o algoritmos en funcionamiento paralelo, ya que el desempeño se ve influido por los costos y límites de comunicación entre los procesos. Este efecto secundario se conoce como ley de Amdahl.

Para obtener una mejor visión sobre las características del paralelismo, es que se analizarán en los puntos siguientes como llevar a cabo los procesos de comunicación entre los distintos procesos paralelos.

3.3.1 Proceso Particionado

El primer modo de paralelización, se refiere a la clásica técnica de dividir un problema en pequeños trozos para poder resolver cada uno por separado y llegar a resolver el problema general, comúnmente es llamada “divide y vencerás”, esta opción es generalmente usada y con buenos resultados en gran cantidad de problemas, tanto informáticos como de otras disciplinas tanto de la vida real como desde el punto académico.

Pero debido a la naturaleza del presente desarrollo y principalmente al elemento analizado anteriormente que corresponde a los algoritmos ACO, es que resulta casi imposible o ilógico poder llevar a cabo este tipo de proceso.

El funcionamiento de cada una de las clase de algoritmos ACO se basa en el desarrollo de la búsqueda de la solución de manera *constructiva*, mediante constantes *iteraciones* que se basan en procesos anteriores, por lo tanto la idea de poder desarrollar una iteración en un proceso y otra iteración en un proceso diferente, no tiene ningún fin práctico,.ya que el proceso no podría ser llevado en paralelo, sino que se llevaría cabo de manera secuencial pero en

procesos diferentes, lo que a primera vista resulta totalmente inútil y más aún resultaría más costoso por la necesidad de que al final de cada iteración se debería llevar a cabo un proceso de comunicación de la solución parcial encontrada.

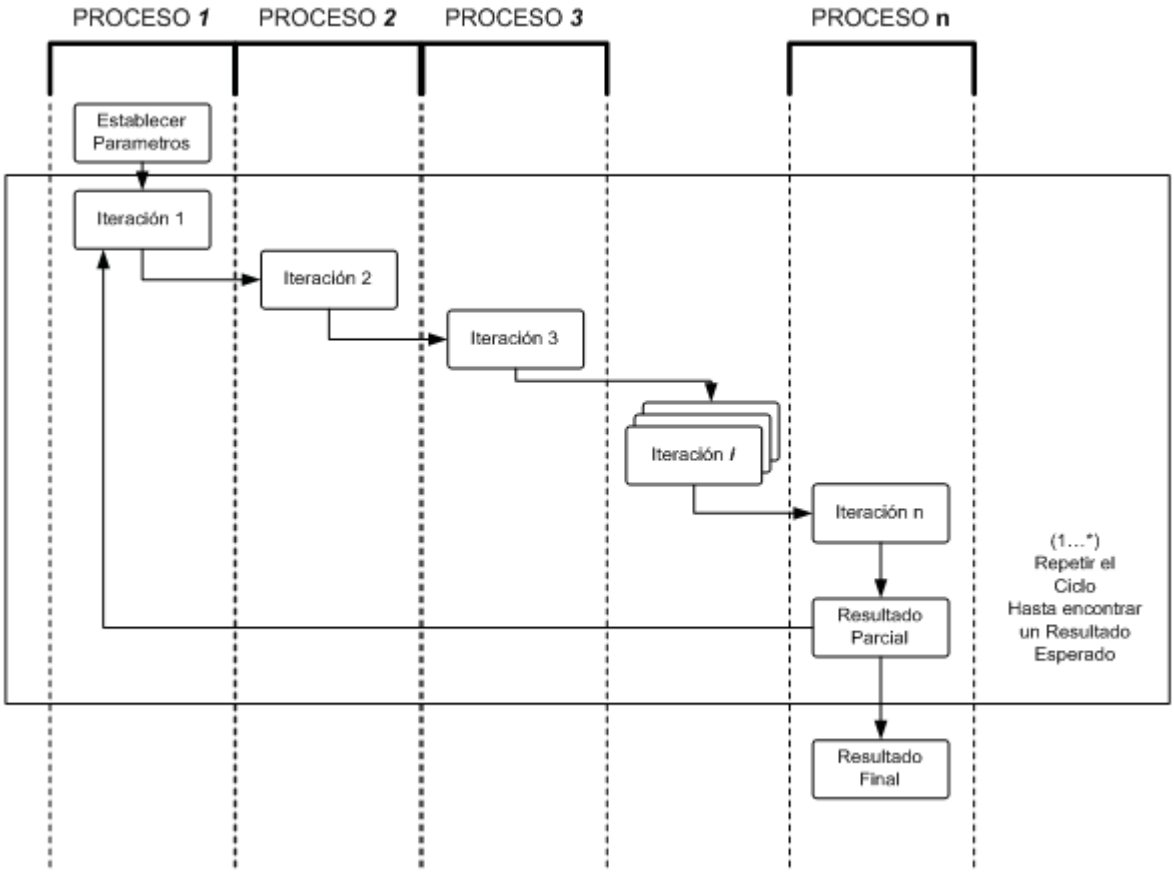


Ilustración 3.4: Esquema de Proceso Particionado.

3.3.2 Procesos Paralelos Independientes

El segundo modo de paralelizar, es mediante procesos paralelos independientes, lo que consiste en llevar a cabo procesos de búsqueda de soluciones de manera aislada, es decir, un proceso desarrollará todas las etapas por si solo, con sus respectivos parámetros y características individuales. La característica que presenta esta configuración, es la nula cooperación entre procesos, ya que el incremento y conocimiento de las soluciones parciales solo es generada de manera local.

Cabe destacar que el tipo de resultado esperado de este tipo de configuración es sobre poder determinar cuales son los parámetros que presentan mejores prestaciones, y solo una vez terminada las búsquedas por cada proceso, existirá una comunicación y evaluación central

para determinar cual o cuales han presentado la mejor solución y determinará la solución final. [10].

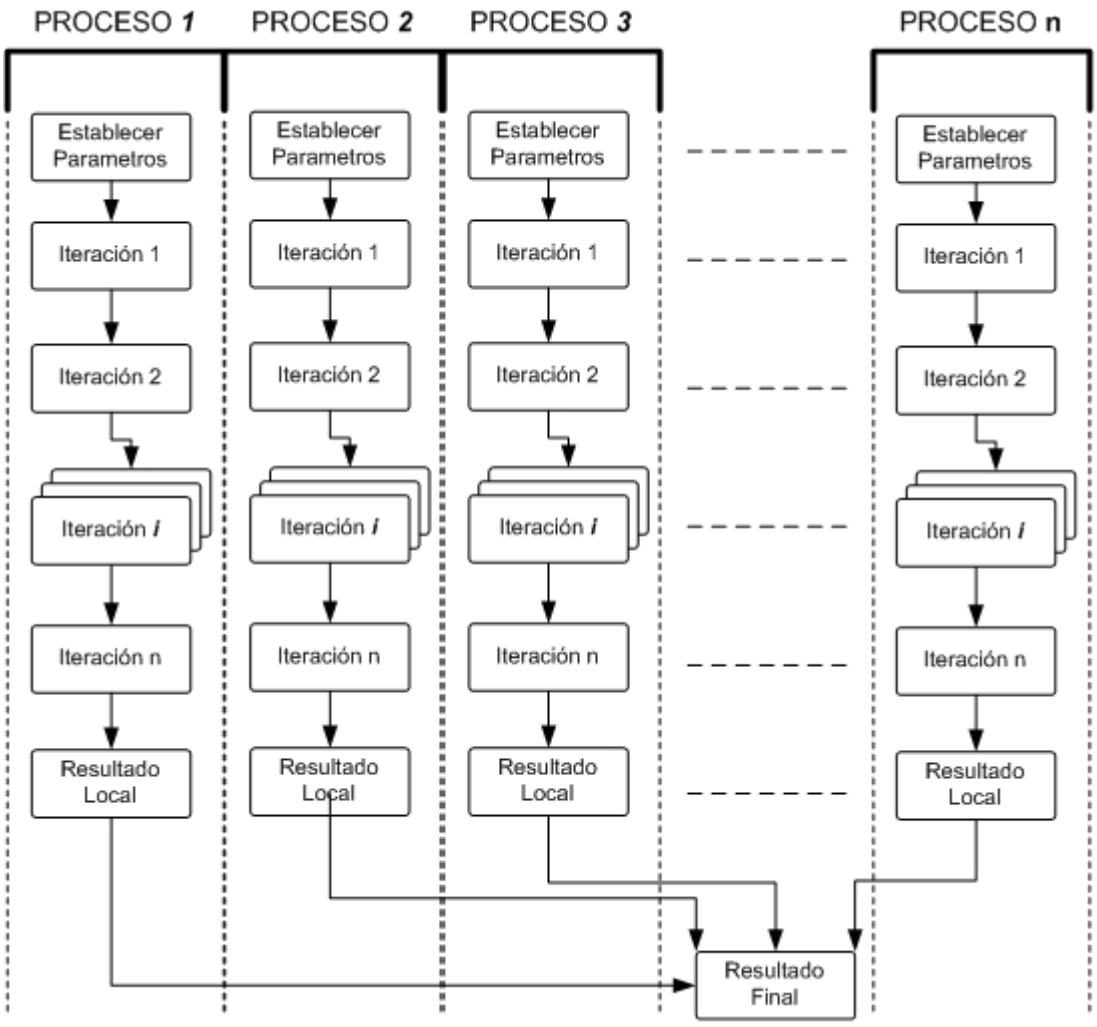


Ilustración 3.5: Esquema de Procesos Paralelos Independientes.

3.3.3 Procesos Paralelos Colaborativos

Este modelo consiste en generar solo un tipo de aplicación, que se replica en varios equipos o procesadores, los cuales podrán ser configurados o seteados de manera independiente, pero con la característica de poder comunicarse durante el desarrollo de las soluciones. Lo característico de este modelo, es que presenta las capacidades de ser colaborativo y constructivo, lo cual tiene gran semejanza con los algoritmos ACO que estarían ejecutándose en cada proceso [10].

Específicamente aplicado a algoritmos ACO, se trata de configurar distintas colonias de hormigas en cada proceso, de manera que según algún tipo de instrucción les permitan comunicarse las soluciones parciales, de manera de que cada colonia no solo aprenda del desarrollo anterior de ella, sino que de las demás colonias, por ejemplo, aprender específicamente de la mejor colonia de las que están funcionando en paralelo.

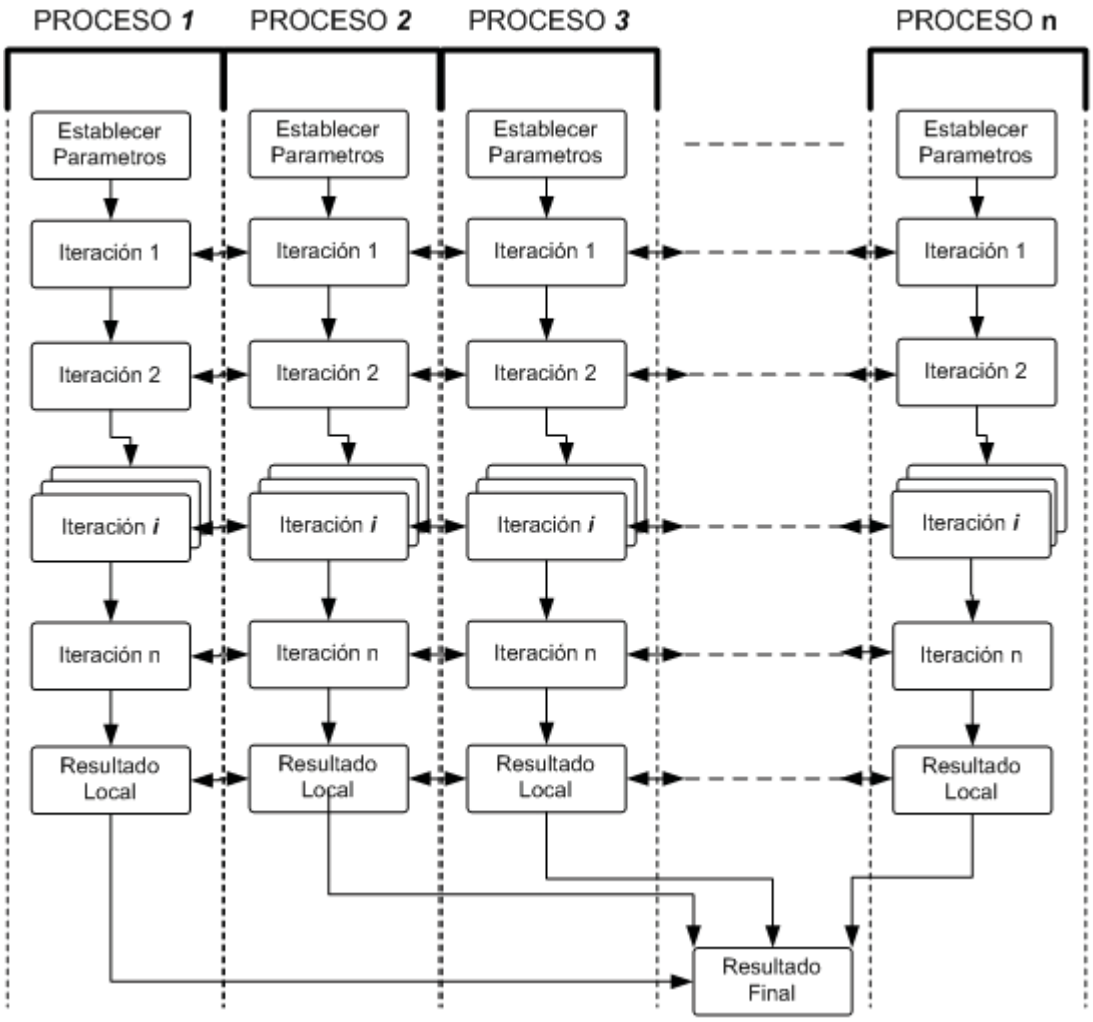


Ilustración 3.6: Esquema de Procesos Paralelos Colaborativos.

3.3.4 Procesos Paralelos Colaborativos Heterogéneos

La teoría indica que sería posible desarrollar procesos de diferente naturaleza y que mediante algún proceso de comunicación en común puedan intercambiar información, por ejemplo, entre procesos que ejecuten algoritmos ACO diferentes, con procesos que desarrollen SA (Simulatead Anealing) y/o GA (Genetic Algorithm) y/o PSO (Particle Swarm Optimization) u sus diversas combinaciones [10].

Este tipo de paralelización conlleva a una homologación sobre el desarrollo de las soluciones, que principalmente se verá reflejado en la forma que estará expresada la misma.

Este tipo de configuración claramente presenta dificultades de implementación ostensibles, ya que tratar de solucionar un mismo problema con distintos algoritmos y distintas formas de expresar sus soluciones, el llegar a un proceso colaborativo es demasiado difícil.

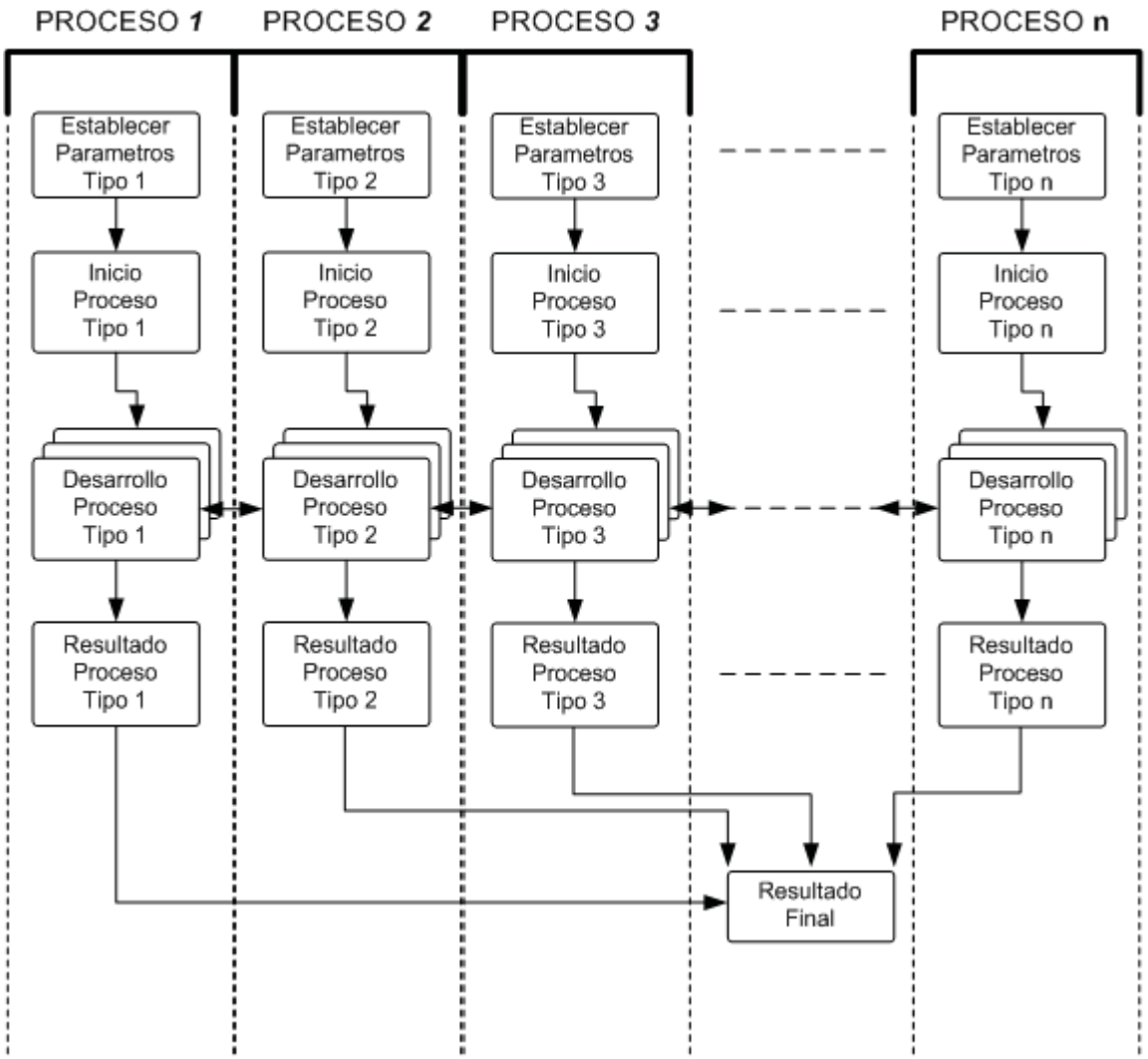


Ilustración 3.7: Esquema de Procesos Paralelos Colaborativos Heterogéneos.

3.4 Heurística

Las heurísticas forman parte importante del presente desarrollo, ya que de ella dependerán las características de la manera de como se llevará a cabo el proceso de selección de los elementos a incluir en la solución final.

Se podrá entender como heurísticas las reglas aproximadas seleccionadas sobre la base de que ayudarán en la solución del problema o contribuirán a reducir el tiempo consumido en la búsqueda de la solución. La evaluación sobre las técnicas de programación heurísticas se realiza principalmente por procedimientos inductivos, más que por procedimientos deductivos.

Si bien existe un número inmenso de heurísticas por elegir, la opción se determina principalmente por el nivel de aspiración en las mejoras en los resultados esperados.

La heurística base aplicada en problemas combinatoriales desarrollados mediante algoritmos ACO, se basa en la división de 1 por el consto del siguiente elemento a agregar a la solución, el concepto de costo cambia de acuerdo a cada problema, ya que mientras en un problema TSP el costo es la distancia entre las ciudades, en un problema SCP el costo se refiere al valor que tiene sumar una columna a la solución. Cabe mencionar que en el desarrollo de la aplicación se utilizarán la mayor cantidad de heurísticas posibles para el problema.

3.5 Integración de Componentes

Debido a la gran cantidad de posibles combinaciones de cada uno de los elementos involucrados en el desarrollo, es que se realiza a continuación un estudio y análisis, sobre la compatibilidad y aplicabilidad de los componentes en conjunto.

En primer lugar se desarrollará el análisis de la posibilidad de utilizar un algoritmo ACO en la resolución de un problema combinatorial, como es el caso del tipo de problema de selección de subconjuntos.

En el análisis de los tipos de problemas de selección de subconjuntos se identificaron tres tipos: SCP, SPP y KP. De estos tipos, todos son igual de prometedores, pero la decisión de cual esta involucrado en el desarrollo, se basa en las potencialidades de aplicación en problemas conocidos y de gran aplicabilidad en problemas reales.

Del tipo Particionamiento (SPP), es el menos prometedor debido a su forma restrictiva en relación a la selección de cada elemento de la solución. Ya que la cobertura de cada fila por alguna columna sea única.

Entre problemas de Cobertura (SCP) y Capacidades (KP), la determinación sobre el primero se debe a su potencial de aplicación y a las preferencias del estamento gestor del ámbito de estudio, que mostró mayor interés en que se desarrollara la resolución de un problema de cobertura de conjuntos SCP.

Por estas razones y las mayores potencialidades de desarrollo, es que se buscará solucionar un problema de cobertura de conjuntos mediante algún algoritmo de colonias de hormigas ACO.

3.5.1 Metaheurística ACO aplicada a Problemas SSP.

En todos los algoritmos ACO, una hormiga comienza su proceso con una solución vacía y construye una solución completa mediante la inclusión iterativa de columnas hasta que todas las columnas sean cubiertas, esto hace que la construcción se realice a través de la elección de la columna por el valor del rastro de feromona y/o por la heurística, para luego de realizar la elección actualizar el rastro de feromona.

A continuación se muestra un algoritmo genérico para el caso de la aplicación de un algoritmo ACO en un problema se selección de subconjuntos [13,14].

```
procedimiento ACO_para_SSP
  inicializarParametros();
  mientras queden iteraciones hacer
    desde k := 1 hasta nhormigas hacer
      mientras solución incompleta hacer
        Seleccionar ítem i a ser incorporado con probabilidad P(k,t,n)
      Fin_mientras
      Calcular Costo_de_la_Solución();
      Grabar la_mejor_solución_hasta_el_momento();
    Fin_desde
    Actualizar_Feromona();
  Fin_mientras
  Retornar mejor_solución_encontrada
Fin_procedimiento
```

Ilustración 3.8: Algoritmo ACO para SSP.

Dentro del marco de trabajo y tomando en cuenta que el objetivo es aplicar un algoritmo ACO a un tipo de problema de selección de Subconjuntos específico que es el de Cobertura de conjuntos, en el siguiente punto y su correspondiente pseudo código se explicarán sus características de aplicación mostrando un algoritmo genérico ACO aplicado a SCP.

3.5.2 Metaheurística ACO aplicada al Problema SCP.

Los algoritmos ACO pueden ser analizados desde el punto de vista la heurística utilizada para la construcción de la solución, donde en el caso de SCP cada columna j es asociada a un rastro de feromona τj que indica si es deseable incluir la columna j en la solución de la hormiga y ηj indica la heurística de escoger la columna j.

La utilización de los algoritmos ACO en problemas SCP requiere de algunos cambios en relación a otras aplicaciones como por ejemplo al del vendedor viajero, las principales diferencias se refieren a:

Las hormigas al construir su solución individual, no necesariamente se demorarán lo mismo o mejor dicho no terminarán después del mismo número de pasos, que dependerá solamente si la solución esta cubierta.

El orden en que las hormigas agregan columnas a la solución no es importante, lo cual difiere bastante de algunas aplicaciones, por ejemplo de TSP donde el orden sí era importante.

Otra característica es que al algoritmo ACO aplicado a SCP puede contener algunos algoritmos de búsqueda local que modifiquen el proceso, por ejemplo al dejar que en la construcción de la solución por una hormiga esta pueda contener componentes redundantes que serán eliminados mediante algún procedimiento de búsqueda local.

```
procedimiento ACO_para_SCP
  inicializarParametros();
  mientras queden iteraciones hacer
    desde k := 1 hasta nhormigas hacer
      mientras solución no completa hacer
        agregarColumna(k);
        agregarAListaTabu(k);
      fin_mientras
    fin_desde
    actualizarOptimo();
    actualizarFeromona();
  fin_mientras
  retornar mejor_solución_encontrada
fin_procedimiento
```

Ilustración 3.9: Algoritmo ACO para SCP.

Como ya se mencionó en puntos anteriores, no existe una única versión de los algoritmos ACO, la versión que se optó fue ACS principalmente por sus procesos de actualización

constantes y que se prevé de gran utilidad cuando se desarrollen búsquedas en paralelo y mejoramiento de las soluciones a partir de conocimiento y soluciones realizadas por otras colonias.

3.5.2.1 Ant Colony System aplicado a Problema SCP.

ACS explota la elección pseudo randómica en la construcción de la solución, donde la hormiga *k* elegirá la siguiente columna según: [27,28]

$$j = \begin{cases} \arg\max_{l \in S_k} \{\tau_l [\eta_l]^\beta\} & \text{si } q < q_0 \\ P & \text{si no} \end{cases} \qquad (3.20)$$

Donde q es un número randómico uniformemente distribuido entre 0 y 1; q0 es un parámetro que controla como será la decisión de la hormiga en cada paso, si q0 es fijado en un valor mas cercano a 1 se utilizará en mayor medida la experiencia, ya que casi siempre se optará por la columna que tenga la mejor relación de carga de feromona y la información heurística que se este utilizando.

En caso contrario, si q0 es fijado en un valor bajo, la opción sobre que columna se agregará a la solución será en base a una nueva ponderación *P*, que permite la elección de una columna que no necesariamente es la con mejor potencial, *P* esta determinado por la probabilidad P_j^k que matemáticamente se representa como:

$$P_j^k = \begin{cases} \dfrac{\tau_j [\eta_j]^\beta}{\sum_{l \in S_k} \tau_l [\eta_l]^\beta} & \text{si } j \notin S_k \\ 0 & \text{si no} \end{cases} \qquad (3.21)$$

La elección del siguiente elemento a agregar a la solución, se realiza considerando la carga de feromona y la heurística de la potencial columna, la cual esta determinada por la carga de feromona y la heurística de todas las columnas que aún no han sido consideradas en la solución parcial.

Donde S_k es el conjunto de columnas incluidas en la solución parcial de la hormiga *k* y el parámetro β es mayor o igual a cero y determina la influencia relativa de la información heurística respecto a la feromona.

Una vez todas las soluciones sean completadas, los valores de feromona son actualizados en primer lugar por la evaporación de los rastros mediante τ_j, que esta dada por:

$$\tau_j = (1-\rho)\cdot\tau_j \qquad (3.22)$$

Luego es agregada una cantidad de feromona $\Delta\tau = \frac{1}{z}$ a las columnas contenidas en la mejor solución encontrada hasta el momento, donde z es el costo de la solución utilizada en la actualización de feromona.

En ACS las hormigas modifican los rastros de feromona también mientras construyen la solución, inmediatamente después de haber agregado una columna a la solución parcial Sk , las hormigas modifican el rastro de feromona τ_j de acuerdo a:

$$\tau_j = (1-\xi)\tau_j + \xi\tau_0 \qquad (3.23)$$

Donde ξ es un parámetro entre 0 y 1;

τ_0 es el valor inicial del rastro de feromona y esta dado por:

$$\tau_0 = \frac{1}{(n \bullet z_{gr})} \qquad (3.24)$$

Dicho parámetro a pesar de ser un cálculo, en el caso práctico utilizado en el presente proyecto el valor es bastante bajo, por lo tanto τ_0 se fijo en un valor de 10^{-6} .

Una vez que todas las hormigas han construido su solución, se realiza el depósito de feromona de acuerdo a:

$$\tau_j = (1-\rho)\tau_j + \rho\cdot\Delta\tau_j^* \quad \forall j \in S^* \qquad (3.25)$$

Donde S^* es la mejor solución hasta el momento, y $\Delta\tau^*_j$ esta dado por:

$$\Delta\tau_j^* = \frac{1}{z^*} \qquad (3.26)$$

Donde z^* es el costo de S^*, valor que puede cambiar según la heurística.

3.5.3 Metaheurística ACO aplicada en Paralelo.

Aunque existen eficientes estrategias que hacen posible la implementación de metaheurísticas secuencialmente, el tiempo requerido para una adecuada exploración del espacio de búsqueda puede ser muy largo para implementaciones de procesos de gran tamaño y/o complejidad. Por ello es que la utilización del paralelismo como medio de búsqueda de mejoras en tiempo y calidad de las soluciones, apareció como una alternativa tentadora para ser explorada y explotada [5,6,8,31,33].

La utilización de algoritmos ACO al tener características colaborativas en la confección de sus soluciones, los hacen especialmente apropiados para su paralelización y distribución entre diversos procesadores de una red. Los hace interesantes, pero a la vez genera el requerimiento de realizar un estudio sobre las características de cada uno de los algoritmos y como se debe llevar a cabo en paralelo, donde toma gran importancia el momento en el cual se actualizan los valores globales de la búsqueda y cuando esas actualizaciones afectan al proceso completo, ya que al estar funcionando de manera distribuida, un algoritmo que requiera actualizaciones constantes, también requerirá de comunicaciones constantes.

Es necesario considerar una limitante que se encuentra para la utilización de este tipo de algoritmos, que se refiere al aspecto técnico, ya que en su aplicación requieren gran cantidad de recursos tanto de procesador como de memoria, este problema se ha ido superando debido al mejoramiento y desarrollo de nuevas tecnologías, tanto de estaciones de trabajo, como de plataformas de desarrollo de software. Sobre el proceso de paralelización existen 2 hipótesis de trabajo, a utilizar referentes al modo que se podría llevar a cabo.

3.5.3.1 Metaheurística ACO aplicada en Paralelo estructura Independiente.

Esta opción se refiere a que el desarrollo de la solución se realizará de manera paralela, pero no existiendo comunicación durante el proceso, es decir, cada colonia trabajará en búsqueda de su propia solución y una vez desarrollados todos los procesos de búsqueda en cada colonia, se evaluará el resultado para poder determinar cual ha sido la mejor solución encontrada, esta opción presenta algunas características que son interesantes de analizar para poder determinar su real potencial.

Al ser procesos independientes, no existe el riesgo de que algún proceso pueda influir de manera errónea en otro, esto se produce generalmente en algoritmos ACO cuando han transcurrido algunas pocas iteraciones y las cuales han producido alguna solución de rápida convergencia, la cual no necesariamente es la mejor solución, sino que se debe a que las primeras búsquedas optaron por recargar demasiado una potencial buena solución.

Existe la posibilidad de poder realizar procesos tan independientes que hasta pueden tener características diferentes, es decir, se podrían realizar procesos de distinto funcionamiento pero con un objetivo de resolución compartido, por ejemplo, desarrollar procesos con diferentes valores o rangos que afecten al algoritmo ACO.

Además existe la posibilidad mediante la creación de un proceso evaluador de resultados, que lleve a cabo una comparación y administración de la soluciones entregadas, esto se podría explicar, si por ejemplo existen *n* colonias trabajando en pos de encontrar la solución, pero el administrador se da cuenta que ya ha recibido *(n+1)/2* resultados y todos entregan el mismo valor, entonces resultaría razonable abortar los procesos de las colonias restantes y dejar como solución final la ya recibida por la mayoría de las colonias. Lamentablemente esta última idea requeriría de un proceso administrador que este informado constantemente del funcionamiento de cada colonia y además que tenga la capacidad de poder abortar los procesos de cada colonia.

Existe además la opción de poder utilizar varias máquinas (PC) para resolver cada uno de los procesos independientes; y una vez terminado se comunique con un servidor.

Esto más bien se refiere a la posibilidad de aprovechar los recursos distribuidamente y no saturar alguna plataforma con grandes y numerosas aplicaciones.

Luego de analizar dichas características y expectativas, se piensa que es totalmente aplicable una buena aplicación de este modelo, ya que si uno piensa en problemas donde las distancias entre las peores y la mejor solución son muy pequeñas, resultaría bastante útil poder evaluar las distintas soluciones, y poder encontrar la mejor o estar lo bastante cerca de ella, y no correr el riesgo de quedarse con una solución rápida de convergencia que este tan cerca de la mejor como de las más malas. En capítulos posteriores se demostrará la aplicabilidad del modelo y la posibilidad de poder funcionar con otros modelos de paralelización sin necesidad

realizar grandes cambios, ya que se podrá fijar un intervalo de migración igual al número de iteraciones que se realizarán en cada colonia.

Es interesante mencionar que este modelo de paralelización, ha sido aplicado con sorpresivos buenos resultados a problemas de TSP [31], donde los experimentos mostraron los buenos resultados obtenidos al utilizar aplicaciones de algoritmos por separado y no necesariamente colaborativos.

3.5.3.2 Metaheurística ACO aplicada en Paralelo estructura Colaborativa.

El segundo modelo es mediante procesos paralelos colaborativos, donde la colaboración, se refiere a que mientras se van desarrollando las iteraciones del algoritmo ACO, no solo se comuniquen soluciones finales sino que el resultado de la construcción parcial de la solución, esto permitiría comunicar cuales han sido las soluciones más prometedoras en cada colonia.

Este modelo presenta principalmente la ventaja de poder adecuar directamente el algoritmo ACO para que desarrolle la búsqueda de la solución, no solo de manera colaborativa dentro de cada colonia, sino que entre colonias.

La desventaja es que hay que controlar adecuadamente el flujo de comunicación, ye que si se realizan procesos intensivos de comunicación, se corre el riesgo de saturar los recursos necesarios para llevar a cabo la comunicación, esto claramente dependerá luego de la estrategia de comunicación a utilizar.

Dada la potencialidad y característica de este modelo, es que se hace necesario realizar un análisis más acabado de cómo poder llevar a cabo el modelo, lo que conllevará algunos desafíos en cuanto a una estrategia de comunicación que permita planificar el momento en que se desarrollara la comunicación. Esto esta directamente relacionado con el Sincronismo o Asincronismo de la comunicación [8,10]

3.6 Estrategias de Comunicación

En primer lugar, es necesario determinar los tipos principales de estrategias de comunicación, que son mediante procesos de comunicación sincronizados y asíncronos.

Cuando se desarrolla la comunicación de manera *Sincronizada*, el proceso se basa en la función de un proceso administrador o colonia maestra que inicia el proceso comunicando a las sub-colonias los valores iniciales del proceso, cada sub-colonia desarrolla sus procesos de búsqueda localmente y luego de terminada, cada proceso envía sus resultados a la colonia central, la cual espera todos los resultados, los compara y clasifica, para luego comunicárselos a las colonias indicando los nuevos valores de las mejores soluciones encontradas hasta el momento por todas las colonias.

Cuando se trata de *Asincronismo*, hay que especificar que se trata de asincronismo parcial, ya que de igual manera necesita de alguna etapa estipulada de comunicación; pero la diferencia se encuentra en que la comunicación de sus resultados no necesariamente la llevarán a cabo con todas las colonias, es decir, si al comenzar la búsqueda, un grupo de colonias pueden desarrollar más rápido su búsqueda, estas no necesitan que las demás colonias hagan lo mismo, sino que comunican sus soluciones y las colonias que se encuentren más demoradas podrán aprender de dichas soluciones y mejorar sus procesos rápidamente.

En consecuencia una vez realizado este análisis, se identificaron algunos elementos que deben ser estudiados para poder aplicar en el desarrollo: [5,6]

- *El intervalo de migración:* es necesario determinar la cantidad de iteraciones que se desarrollarán de manera local, antes de comunicar su resultado hacia las otras colonias o hacia la aplicación administradora, este parámetro es el que permitirá como se mencionó anteriormente poder realizar procesos altamente colaborativos o procesos independientes entre si.

- *La elección del Comunicado*: la implementación en paralelo permite diversas comunicaciones y diversos elementos a comunicar, esto se refiere a que en la práctica lo que se comunica son las soluciones parciales de las colonias, pero la comunicación podría incluir otros valores dependiendo de la necesidad de investigar nuevas prestaciones; por ejemplo, además de comunicar las soluciones, se podría comunicar las listas tabú y/o matriz de feromonas y/o soluciones parciales; dicha elección claramente requiere de una análisis práctico sobre el desarrollo, donde la opción que se considera con mejores

potencialidades y de mayor lógica de utilización es comunicar la mejor solución encontrada hasta el momento por cada colonia.

- *La tasa de migración*: este valor dependerá en gran medida del punto anterior, ya que la tasa de migración se refiere a la cantidad de valores que serán comunicados, y este valor depende de que se comunicará; ya que si se tratara de comunicar la matriz de feromona claramente la tasa sería unitaria ya que existe solo una matriz de feromona por colonia, pero si la elección del comunicado serían las mejores soluciones, la cantidad de hormigas o soluciones extremas, la tasa cambiaría y estaría limitada.

- *El criterio de selección*: este criterio lo que determina es la selección de que valores incluirá el comunicado, ya que si se opta por comunicar el conjunto de columnas pertenecientes a una solución, debe considerarse si se comunicará la mejor solución, las mejores soluciones o las peores. Tal como se mencionó en el punto anterior ese criterio ha sido establecido en comunicar una solución que contenga la mejor combinación de columnas hasta el momento realizadas por cada colonia.

Dentro de la paralelización y sus procesos de comunicación, hay que tomar en cuenta que no existe una regla específica de cómo llevar acabo dicho proceso, donde la manera de realizar eficientemente dichos procesos dependerá fuertemente de la capacidad de la plataforma, del problema al cual es aplicado y del algoritmo utilizado para paralelizar.

3.7 Estructuras de Comunicación

Existen algunas propuestas que se relacionan principalmente a la forma en que se llevará a cabo la distribución de las colonias. A continuación se detallarán las estructuras generales de paralelización de colonias de hormigas:

3.7.1 Estructura de Comunicación Administrador-Buscador

La propuesta consiste en la utilización de un proceso o colonia central y varias colonias que realizan sus procesos de búsqueda en función de lo que le ordene la colonia central y de que parámetros le envíe esta misma.

El proceso comienza cuando la colonia central envía a cada colonia los datos iniciales o alguna especificación referente al funcionamiento del algoritmo; una vez recibida esta información por cada colonia, estas comienzan el proceso de búsqueda, la cual se desarrolla a partir de una serie de iteraciones, las que van logrando obtener un valor de convergencia, momento en el cual, la colonia termina su proceso temporalmente y comunica la mejor solución encontrada a la colonia central, la cual una vez recibida la mejor solución de cada colonia, se encarga de clasificar y ponderar las soluciones, para luego cambiar si es necesario los nuevos mejores valores encontrados por el funcionamiento en paralelo. Este proceso se repite hasta que se cumpla un número determinado de iteraciones o se encuentre una solución aceptable [10].

A continuación se muestran en pseudos-código los procedimientos de colonia maestra y buscadoras de soluciones.

```
procedimiento Colonia_Administradora
  inicializarParametros();
  Mientras queden_esclavas_sin_conectar
    Recibir_conexión_de colonia ();
  Fin_mientras
  Enviar parametros de inicio();
  mientras queden colonias sin terminar hacer
    desde id := 1 hasta N°deBuscadoras hacer
      Recibir solución de esclavas();
    Fin_desde
    Calcular la_mejor_solución_comunicada_hasta_el_momento();
    Guardar la_mejor_solución_comunicada_hasta_el_momento();
     Enviar mejor solución de esclavas();
  Fin_mientras
  Retornar major_solución_encontrada
Fin_procedimiento
```

Ilustración 3.10: Algoritmo Pseudo-código Colonia Administradora.

```
procedimiento Colonia_Buscadora
  recibirParametros();
  inicializar variables()
  mientras queden iteraciones hacer
    desde k := 1 hasta nhormigas hacer
      mientras solución no completa hacer
        Realiza_proceso de hormigas();
      fin_mientras
    fin_desde
    actualizarOptimo();
    Enviar solución parcial();
    Recibir mejor solución parcial();
    actualizarOptimo()
    actualizarFeromona();
  fin_mientras
  retornar mejor_solución_encontrada
Fin_procedimiento
```

Ilustración 3.11: Algoritmo Pseudo-código Colonia Buscadora.

3.7.2 Estructura de Comunicación en Vecindades

La estructura de vecindad consiste en un proceso que se realiza conceptualmente sin la intervención de una colonia central, el proceso comienza en paralelo, cuando todas las colonias comienzan a desarrollar su proceso de búsqueda a partir de valores iniciales dados por la aplicación central, una vez desarrollado un número de iteraciones o al haber encontrado un valor de convergencia, la colonia comunica su resultado a la colonia siguiente en la estructura de vecindad. Si una colonia recibe información desde la colonia anterior en la estructura, esta decidirá si cambiar y utilizar los nuevos parámetros para intentar mejorar su proceso de búsqueda, este proceso es llevado a cabo por cada colonia.[10]

Existe la posibilidad que la vecindad se represente mediante un anillo bi-direccional, donde el proceso comienza igualmente en paralelo, pero una vez desarrollado un número de iteraciones o al haber encontrado un valor de convergencia o “buena solución“, la colonia comunica su resultado tanto a la colonia siguiente como a la anterior en la estructura lógica. Este tipo de arquitectura presenta mejoras al poder utilizar mayor información proveniente de más colonias, pero a la vez, necesita de mayores recursos de comunicación y de sincronización para poder determina fehacientemente todas las vecindades. Lo que en un principio será tomada como una opción potencial pero no aplicada en el transcurso del desarrollo.

Otro tipo de arquitectura de vecindad, es mediante la representación binaria, donde la estructura se determina a partir de índices para identificar a cada colonia, donde se produce la diferencia en que la comunicación puede ser hacia más de dos colonias, ya que se realiza

hacia varias colonias con las cuales exista solo un bit binario de diferencia. Este tipo de arquitectura presenta mayores prestaciones al poder utilizar y comparar mayor información proveniente de más colonias, pero a la vez, necesita de aún mayores recursos de comunicación y de memoria, ya que puede recibir simultáneamente un gran número de resultados, provenientes de varias colonias, los cuales los puede utilizar de inmediato o almacenarlos hasta que lo estime conveniente.

```
procedimiento Colonia_Administradora
  inicializarParametros();
  Mientras queden_colonias_sin_conectar
    Recibir_conexión_de colonia();
  Fin_mientras
  Enviar parametros de inicio();
  Enviar identificador de colonia();
  mientras queden colonias sin terminar hacer
      Recibir notificación de comunicación de colonia();
      Identificar su posición();
      Mientras existan otras colonias comunicadas a la colonia actual hacer
          Enviar dirección de la siguiente colonia();
      Fin_mientras
  Fin_mientras
  Evaluar los resultados();
  Indicar cual ha sido la mejor colonia();
Fin_procedimiento
```

Ilustración 3.12: Algoritmo Pseudo-código Colonia Administradora en Vecindad.

```
procedimiento Colonia_Buscadora
  recibirParametros();
  inicializar variables()
  recibirIdentificador();
  mientras queden iteraciones hacer
    desde k := 1 hasta nhormigas hacer
      mientras solución no completa hacer
        Realiza_proceso de hormigas();
      fin_mientras
    fin_desde
    actualizarOptimo();
    Enviar petición de comunicación();
    Mientras existan otras colonias comunicadas receptoras de la colonia
actual hacer
      Recibir dirección de la siguiente colonia(),
      Enviar resultado parcial a la siguiente colonia();
    fin_mientras
    Mientras existan otras colonias proveedoras de la colonia actual hacer
      Recibir resultado parcial a la anterior colonia();
      Si resultado recibido mejor al calculado
        actualizarOptimo()
        actualizarFeromona();
      Fin si
    fin_mientras
  fin_mientras
  Enviar solución_encontrada;
Fin_procedimiento
```

Ilustración 3.13: Algoritmo Pseudo-código Colonia Buscadora en Vecindad.

No existiendo límite para poder diseñar alguna estructura de comunicación, las representaciones de vecindades pueden ser combinadas o utilizadas junto con modelos de colonias centrales.

Capítulo 4
Diseño del Sistema

En esta etapa se determina la arquitectura general del sistema y su comportamiento dinámico, adaptando las especificaciones realizadas en capítulos anteriores.

El resultado obtenido de esta etapa de diseño facilita enormemente la implementación posterior, pues proporciona la estructura básica del sistema y como los diferentes componentes actúan y se relacionan.

En lo referente a la forma en que se desarrollará el proceso y cual será el resultado objetivo, la intención es el desarrollo de una gran aplicación, que funcione de manera de un centro de control o cómputo, que permita ir realizando elecciones de cómo se desarrollarán los procesos.

4.1 Elección de Componentes

La creación de una aplicación requiere de conocimientos técnicos, experticia y experiencia en desarrollos de este tipo de problemas; dada las características de ser un proyecto inedito en cuanto a sus elementos y combinaciones entre ellos, provoca que el proceso de diseño se estipule como un proceso dinámico.

4.1.1 Problema

La elección del problema de optimización se basa en la cantidad de aplicaciones posibles de realizar en la vida real, principalmente en lo que se refiere a la utilización y determinación de recursos de una organización, además de ser de interés por parte de los organismos rectores, la elección se cae sobre problemas de cobertura de conjuntos SCP.

4.1.2 Algoritmo ACO

En este aspecto la elección no es única, ya que se desarrolla una aplicación que tiene como estructura un algoritmo ACS (Ant Colony System) pero que al momento de configurar los parámetros, éstos se pueden determinar de manera tal que el algoritmo se comporte como un algoritmo AS (Ant System) u otro.

4.1.3 Paralelización

La manera de cómo se llevará a cabo la paralelización requiere principalmente de una evaluación de factibilidad sobre la aplicación de cada una de las maneras generales de llevar a cabo un proceso en paralelo.

Un proceso particionado, no tiene ninguna lógica ni prevista ninguna ventaja sobre el proceso, ya que dividir el algoritmo lo que presentaría serian bajar en las prestaciones de calidad y tiempo.

Procesos paralelos independientes de distinta naturaleza, si bien presentan características llamativas de llevar a cabo, tiene dificultades en cuanto a su aplicación, ya que el proceso necesario de homologar la forma de expresar los resultados, de llevar a cabo los procesos y realizar efectivamente las comunicaciones; presentan dificultades y posiblemente imposibilidades de llevar a cabo.

Los procesos que si presentan potencialidades claras y por lo tanto son de interés realizar son: procesos en paralelo independientes y colaborativos, donde la principal diferencia radica en el momento de llevar a cabo la comunicación sobre los resultados parciales del desarrollo del proceso. Si se realizan solo al final de cada proceso o de manera intensiva durante la búsqueda.

4.1.4 Heurística

Se trata de métodos exploratorios durante la resolución de problemas en los cuales las soluciones se descubren por la evaluación del progreso logrado en la búsqueda de un resultado final.

Es de interés determinar una buena heurística, ya que de ella depende en gran parte la calidad de la solución y tiempo necesario para obtenerla, ya que si es muy compleja requerirá de demasiados cálculos.

Las heurísticas, si bien presentan gran cantidad de posibilidades, la elección se basa en las heurísticas más documentadas sobre la aplicación de algoritmos ACO a problemas de Cobertura de Conjuntos, donde las principales son:

- Para la elección de la siguiente columna además de la carga de feromona, se considerará aquellas columnas que presenten el menor costo: *1 / Costo de la Columna.*

- La elección de la siguiente columna además de la carga de feromona, se considerará cuantas nuevas filas son agregadas a la solución en conjunto con el costo de la columna que las contenga: *Cantidad de filas que se agregan a la solución / Costo de la Columna.*

4.1.5 Estrategia de Comunicación

Para llevar a cabo los procesos de comunicación entre las aplicaciones en paralelo, se necesita determinar ciertas configuraciones en relación a la forma y objetivos de los desarrollos.

En primer lugar es necesario determinar el sincronismo en la comunicación, es decir si se optará por comunicaciones en solo momento para todos o si serán posibles comunicaciones asíncronas, es decir, sin un momento determinado para aquello. El diseño contempla la implementación de ambas estrategias, siendo el sincronismo la primera opción.

Otros aspectos a tomar en consideración se refieren a:

- *El intervalo de migración:* será configurable por parte del usuario.

- *Elección del Comunicado*: Se comunicarán las combinaciones de columnas pertenecientes a una solución, tanto de manera local como global dependiendo de los valores obtenidos.

- *Tasa de migración*: Ya que el comunicado son las mejores combinaciones, la tasa corresponderá generalmente a una combinación por vez, pero dependerá de la configuración de comunicaciones que se estipule.

- *Criterio de selección*: La selección de las combinaciones a migrar entre procesos (colonias), serán las mejores combinaciones encontradas tanto de manera local o global, esto dependiendo de la etapa del proceso que se realice.

4.1.6 Estructura de Comunicación

La propuesta consiste en la utilización de varias estructuras para llevar a cabo la comunicación, referentes a los componentes o procesos que se comunicarán y como. Las estructuras principales se refieren a una configuración centralizada y otra distribuida.

4.1.6.1 Estructura de Comunicación Administrador-Buscador

Esta configuración estará determinada por la existencia y dependencia sobre un proceso (colonia) central, la cual determinará y controlará los procesos de las demás colonias distribuidas, las cuales llevarán a cabo la búsqueda de la solución.

El proceso o colonia central, fijará parámetros y dará inicio al proceso. Las colonias comenzarán la búsqueda de la solución y de acuerdo al parámetro de intervalo de migración se comunicarán con la colonia administradora de manera asíncrona. Permitiendo que en cualquier momento que una colonia buscadora determine una solución para comunicar, la colonia administradora estará dispuesta a recibirla y poder cambiar la mejor solución encontrada por la totalidad de las colonias pertenecientes a la estructura.

La existencia de la aplicación administradora y la comunicación que se llevará a cabo con ésta, tiene la característica de generar una arquitectura de cliente-servidor y cuyos momentos de comunicación serán Síncronos, es decir, existirán etapas del proceso estipuladas para llevar a cabo las comunicaciones.

El funcionamiento síncrono de la implementación se establece al momento de dar el punto de partida y valores de desarrollo a cada colonia mediante datos que indicarán el ciclo de

iteraciones, que estipula la cantidad de iteraciones que se desarrollarán en cada colonia antes de comunicar su solución hacia la colonia administradora; la colonia administradora esperará la llegada de todas las soluciones provenientes de cada una de las colonias antes de desarrollar una ponderación de las mismas y determinar cual ha sido la mejor, para luego comunicar dicha solución a todas las colonias pertenecientes a la estructura.

Un aspecto práctico sobre el proceso de comunicación y que tiene como objetivo poder mejorar las prestaciones del sistema, es poder utilizar lo mínimo posible los recursos de comunicación, cada colonia comunicará en cada ciclo de iteración hacia la colonia administradora solo el valor de la mejor combinación de columnas, específicamente se refiere al costo de la solución y la cantidad de columnas cubiertas. Donde a partir de dichos datos la colonia central evaluará, dependiendo a que valor le de mayor importancia, cual ha sido la mejor solución encontrada y luego de realizado dicho proceso se le indicará a la colonia que haya realizado esa combinación que la envíe, para una vez recibida por la colonia central comunicarlo hacia las demás colonias y que continúen con su proceso de búsqueda y sigan sus iteraciones utilizando la nueva mejor combinación incorporada a su proceso.

El funcionamiento en detalle de cada colonia participante en el proceso de búsqueda, consistirá en recibir los datos necesarios para el funcionamiento por parte de la colonia administradora, para luego a partir de una señal de partida comenzar el proceso, dicho proceso esta determinado por un límite de iteraciones, el cual indica cuantas iteraciones deben llevarse a cabo antes de comunicar su mejor solución hacia la administración para que ésta la evalúe. Luego de la evaluación pertinente se recibirá si es necesaria una nueva mejor combinación de columnas que será tomada en cuenta para actualizar tanto la matriz de feromona como la mejor combinación encontrada, en la practica es repetir el proceso de actualización global que realiza el algoritmo ACO ACS pero utilizando la mejor combinación de columnas obtenidas hasta el momento por la estructura paralela de colonias.

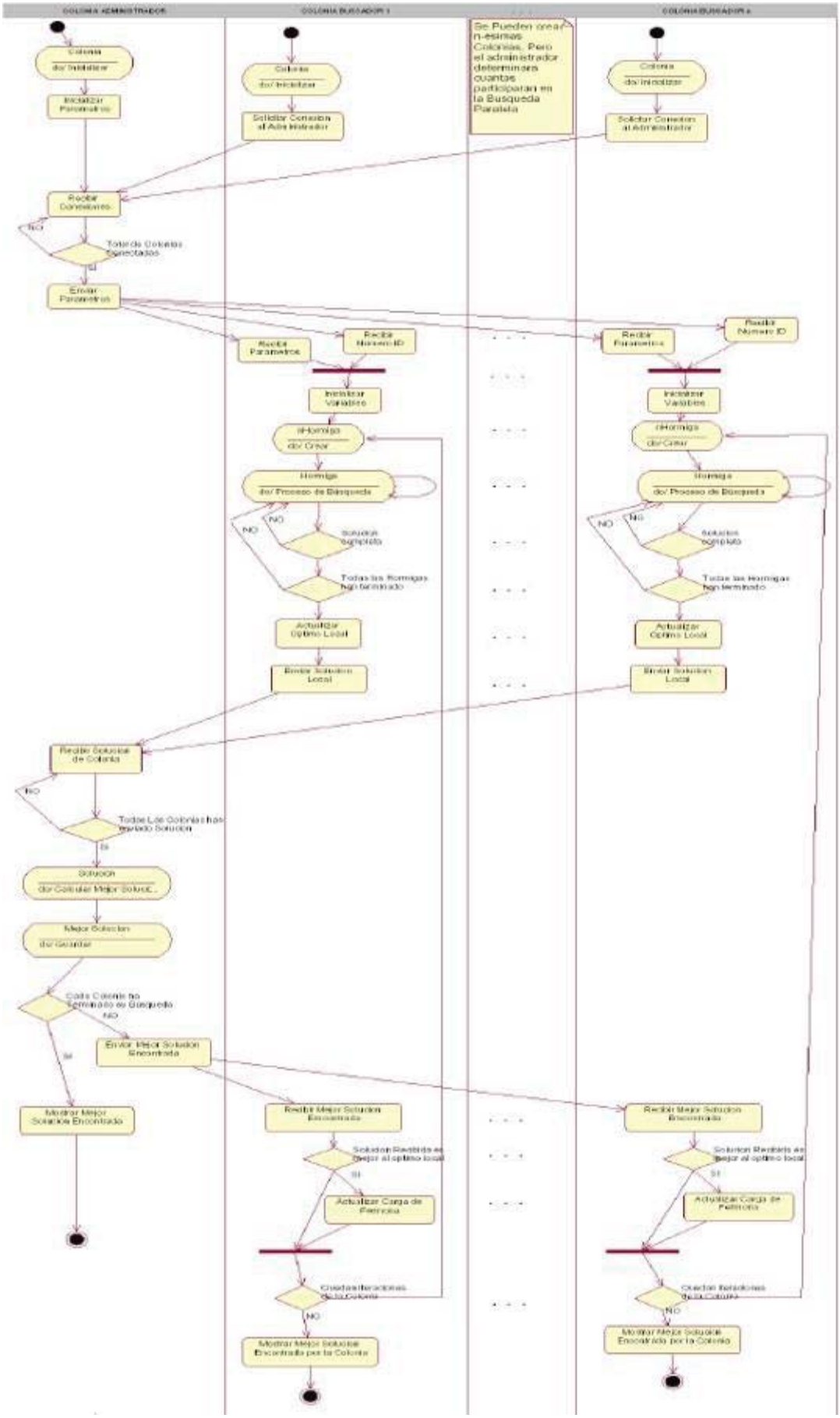

COLONIA ADMINISTRADOR
COLONIA BUSCADOR 1
COLONIA BUSCADOR n
Se Pueden crear n-esimas Colonias. Pero el administrador determinara cuantas participaran en la Busqueda Paralela
Colonia
Inicializar Parametros
Solicitar Conexion al Administrador
Recibir Conexiones
NO
Total de Colonias Conectadas
SI
Enviar Parametros
Recibir Parametros
Recibir Numero ID
Inicializar Variables
nHormiga
Hormiga
Proceso de Busqueda
Solucion Completa
Todas las Hormigas han terminado
Actualizar Optimo Local
Enviar Solucion Local
Recibir Solucion de Colonia
Todas Las Colonias han enviado Solucion
Solucion
Calcular Mejor Solucion
Mejor Solucion
Guardar
Cada Colonia ha Terminado su Busqueda
Enviar Mejor Solucion Encontrada
Recibir Mejor Solucion Encontrada
Mostrar Mejor Solucion Encontrada
Solucion Recibida es Mejor al optimo local
Actualizar Carga de Feromona
Cuantas Iteraciones
Mostrar Mejor Solucion Encontrada por la Colonia

Ilustración 4.1: Diagrama de Actividad Proceso Administrador-Buscador.

Específicamente, los datos que serán necesarios introducir en la aplicación son los siguientes:

Nombre del archivo Benchmark a evaluar.
Método de resolución.
Valor de ro.
Cantidad de Colonias participantes en la búsqueda.
Ciclo de Iteraciones a realizar por cada colonia antes de comunicar su resultado parcial.

Luego de haber realizado el ingreso de los cuatro primeros parámetros, la aplicación administradora se quedará en espera antes de pedir el ciclo de iteraciones, esta espera se debe a que permanecerá en estado de “escucha” hasta que se comuniquen y den señales de creación la totalidad de las colonias establecidas que participarán en la búsqueda; una vez comunicadas e identificadas todas las colonias se ingresará el quinto valor mencionado y se le enviarán los datos y el punto de partida a las colonias para que comiencen sus procesos.

En las aplicaciones de colonias de búsqueda, se deben ingresar y realizar otro tipo de procesos:

Necesidad de identificarse con la colonia central: para ello se debe ingresar la dirección IP del proceso servidor, cabe mencionar, que dicha identificación funcionará indistintamente de manera local ingresando “Localhost” o en un entorno de red ingresando la dirección IP del equipo, por lo tanto el único dato que es necesario ingresar en los procesos de búsqueda es la dirección IP que debe ser conocida por cada operador de las colonias.

El proceso contempla una serie de estados, los que se refieren mayormente a los datos que se encontrarán en movimiento o comunicándose entre aplicaciones, los cuales son los siguientes:

Número identificador de la colonia, que permitirá reconocer mediante un número que colonia es la que realiza la comunicación.
Valores de mejor solución encontrada por la colonia, se comunicarán tanto el número de columnas incluidas en la mejor solución generada hasta el momento, así como el costo de la solución.

Detalle de la mejor solución encontrada en la etapa, una vez ponderados los valores enviados por cada colonia, la aplicación administradora solicita a la mejor colonia que envíe el detalle de dicha solución, principalmente se refiere el conjunto de columnas que forman parte de la solución.

Comunicado global por parte de la administradora: el comunicado del punto anterior ahora se comunica a cada colonia que este participando en la búsqueda y que no presente el mejor resultado.

Los datos de salida o interfaz directa con el usuario se referirán principalmente a resultados del proceso:

Mejor solución global del problema.

Mejor solución encontrada por la colonia.

Archivos de texto con los datos del proceso (iteraciones, tiempos, mejores y peores soluciones, etc.)

4.1.6.2 Estructura de Comunicación en Vecindades

Esta configuración presenta la No dependencia de un proceso (colonia) central, la cual si bien podría existir para determinar y controlar los procesos de las demás colonias, pero no determinará el proceso de búsqueda y comunicación entre las colonias buscadoras de soluciones.

La estructura de vecindad que se diseñará e implementará es del tipo anillo, que consiste en un desarrollo en paralelo de cada colonia buscadora; comunicarán su resultado de acuerdo a una forma lógica de anillo, es decir, la colonia i comunicará su combinación encontrada a la colonia $i+1$; la última colonia del índice se comunicará con la primera y completará la estructura del anillo.

Este desarrollo mantiene la resolución de un problema de cobertura de conjuntos (SCP) mediante la utilización del algoritmo (ACS). El funcionamiento síncrono de la implementación se mantiene al establecer del mismo modo que la estructura anterior, etapas específicas de comunicación. El envío de los datos, a diferencia del modelo administrador-

buscador, consiste tanto en comunicar los valores totales del resultado obtenido así como la combinación de columnas pertenecientes a la solución, en una sola etapa.

Cada colonia perteneciente al anillo, debe al momento de recibir el comunicado ponderarlos y de ser mejores los recibidos que los realizados por la colonia, realizará una nueva actualización global de los valores de búsqueda.

El detalle del proceso, se considera como una adaptación al diseño administrador-buscador, al cual se le efectúan los siguientes cambios:

El proceso servidor no participará en la resolución.

Se entenderá el parámetro cantidad de colonias como la cantidad de colonias pertenecientes al anillo.

Las colonias del anillo necesitarán identificarse con el proceso administrador, pero su número de identificador “ID”, será el considerado para determinar la posición de cada colonia en la estructura de vecindades del anillo.

Una vez comenzados los procesos, los datos de que se encontrarán en movimiento o comunicándose entre las colonias, serán los mismos que la anterior, pero sin ser analizados ni ponderados por el proceso servidor. Los datos de salida representan los resultados del proceso, donde además de obtener resultados globales, cada colonia obtendrá y registrará sus propios resultados.

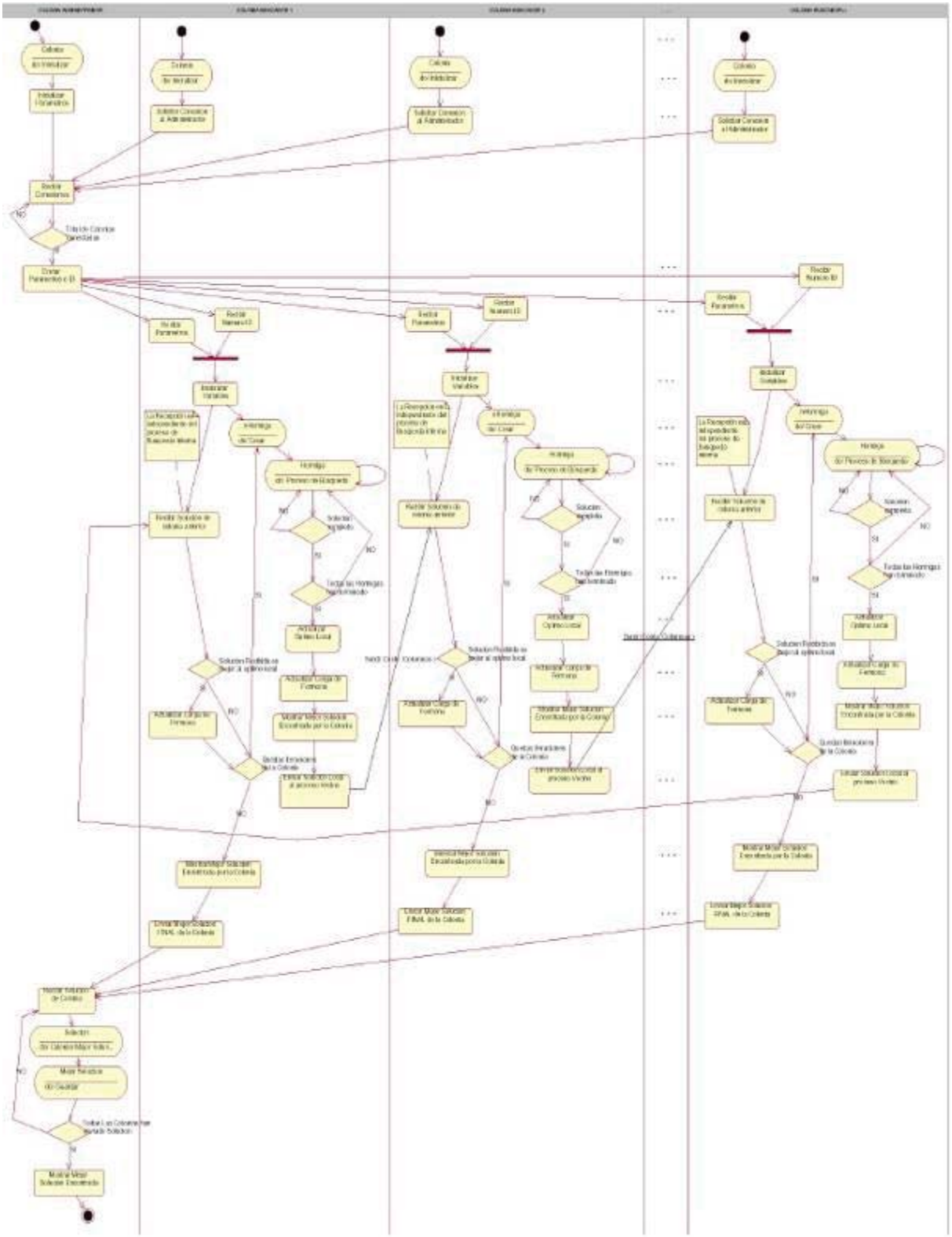


Ilustración 4.2: Diagrama de Actividad Proceso en Vecindad.

Capítulo 5
Metodología.

Para el proceso de investigación y desarrollo se siguió principalmente un proceso de investigación científica simplificada, en el cual se siguieron las siguientes etapas conceptuales:

- *Idea*: Se detectó la existencia de un problema, un fenómeno para el cual sería interesante y llamativa la búsqueda de un razonamiento y mayor entendimiento, específicamente se trata del mejoramiento en los procesos de búsquedas y obtención de mejores resultados en problemas combinatoriales de alta complejidad. Esto se genera principalmente a partir de fuentes ligadas a conversaciones con docentes involucrados en el tema, materiales escritos y búsquedas personales.

- *Planteamiento*: Se realizó un análisis de cuales son los aspectos esenciales del problema para desechar aquéllos que no lo sean. Determinando como se planteó en el capítulos anteriores, un estado del arte ligado principalmente al acotamiento del desarrollo a un problema mas específico mediante técnicas y herramientas determinadas. Además se plantean 3 aspectos claros:
  ◦ Un objetivo determinado en encontrar mejoras al proceso y uso de nuevas técnicas de resolución.
  ◦ Tratando de responder cada una de las preguntas previas y nacidas a raíz de la investigación, como son las mejoras comparativas en cuanto a resultados esperados.
  ◦ Cuya justificación principal radica en la implementación en conjunto de conceptos, técnicas y herramientas, para obtener mejores resultados que los encontrados por algunas investigaciones prévias.

- *Marco Teórico*: Una vez planteado el problema de estudio y establecido el ámbito de la investigación, se realizó un análisis y exposición de aquellas teorías, enfoques, investigaciones relacionadas que se consideraron válidas para el correcto desarrollo de los objetivos específicos y central del desarrollo. Los objetivos del presente punto llevaron a realizar un extenso e intenso trabajo de investigación, donde se trató de obtener los siguientes resultados:
  - Prevenir errores.
  - Orientar el proceso de desarrollo.
  - Centrar el proyecto en ámbitos acotados y concretos, para no sufrir desviaciones o abarcar ámbitos fuera del interés del desarrollo.
  - Generar ideas y áreas de investigación.
  - Desarrollar conceptos que son de interés conocer para un mejor proceso.
  - Tener un marco de referencia donde se encuentra el desarrollo y establecer sus reglas, limitantes y potencialidades.

- *Definición de la investigación y proceso*: Dada la característica del tema del presente proceso de desarrollo, el cual abarca y examina un tema poco estudiado y que no ha sido abordado en su configuración por otros autores, es que se decidió por llevar a cabo un proceso de investigación con características *Exploratorias* y *Descriptivas*, donde se buscarán características llamativas del proceso, para luego describir las propiedades importantes del proceso y sus resultados.

- *Formulación*: Se trata de determinar claramente que se esta buscando, principalmente como objetivo del desarrollo, el cual consiste en determinar si utilizando múltiples colonias de hormigas en paralelo mejoran los tiempos y calidad de las soluciones.

- *Diseño*: El objetivo central de esta etapa es el responder fehacientemente las preguntas de investigación planteadas y someterlas a prueba, mediante una planificación que permita responder las preguntas de investigación de una manera adecuada. Específicamente mediante *Experimentos*.

- *Selección de muestras*: En el presente desarrollo las muestras se refieren a las métricas de tiempo y valor (costo) de los procesos de búsqueda de soluciones a problemas combinatoriales del tipo de cobertura de conjuntos, mediante algoritmos ACO en paralelo.

- *Recolección de datos*: Para el proceso de evaluación y determinación de los resultados del presente desarrollo, es que se generaron una serie de experimentos de los cuales se les obtuvo y almacenaron sus resultados, para luego agruparlos por tipo (problema).

- *Análisis*: Se estudiaron los datos obtenidos de los diversos experimentos y se obtendrá la información necesaria para determinar que y cuales son las características del proceso que permiten asegurar una mejora en los resultados, el análisis siempre contempló no afectar la información por deseos o intuición de obtener mejores resultados. Por lo mismo es que en capítulos posteriores se presentan evoluciones y cambios posteriores al inicio del desarrollo.

- *Resultados*: Por último se presentan los resultados obtenidos en el proceso de desarrollo.

Dada las características del tema y además ligado al objetivo principal del proyecto que consiste en el desarrollo de un software; pero con características muy especiales, principalmente referidas a lo desconocido del tema y la casi mínima experiencia previa en este tipo de desarrollos, es que la metodología utilizada durante el desarrollo ha sido abordada por el marco de investigación antes mencionado, pero agregando particularidades de un proceso de desarrollo de software ágil, iterativo e incremental, donde en primer lugar se desarrolla un proceso de estudio y análisis de la información disponible, tratando de involucrar a todos los conceptos necesarios para el desarrollo.

La siguiente etapa consiste en generar diversos diseños de aplicaciones que tuvieran la característica de ser posibles de realizar y además que se muestren como potenciales generadores de buenos resultados.

Luego disponer la codificación de una aplicación para incluir los diseños generados.

A partir de las aplicaciones funcionales, permitir determinar diseños y productos que se muestren potencialmente como generadores de buenos resultados.

Generar pruebas y conclusiones sobre el desarrollo.

Generar nuevas iteraciones en el proceso, para agregar cambios y mejoras al mismo.

Capítulo 6

Desarrollo.

Una vez desarrollado el proceso de diseño lógico del sistema, la etapa siguiente corresponde al estudio de las características del producto, determinando herramientas de programación y realizando pruebas sobre aplicabilidad de cada una de ellas.

A continuación se presentarán detalles del funcionamiento lógico y físico de los primeros desarrollos prácticos de algunas aplicaciones.

6.1 Desarrollo

Para la realización de los primeros prototipos, se consideraron los siguientes valores de los parámetros incluidos en el proceso:

Elección de un problema de optimización, en primer lugar dada las justificaciones mencionadas en el estudio de las características de los problemas, se considera un problema de Cobertura de Conjuntos. (SCP)

Elección de un algoritmo ACO, tal como el punto anterior y basado en el análisis de las características de los tipos de algoritmos, se utilizará el algoritmo Ant Colony System (ACS).

Elección de una estructura de paralelización, debido a su mayor facilidad de desarrollo y control sobre el proceso la opción es de Colonia Central o Administradora.

Elección de una estrategia de comunicación, la forma en que se desarrollarán las comunicaciones será de manera síncrona, es decir establecer un intervalo de comunicación para todas las colonias y que debe ser cumplido por todas, para continuar con el proceso.

El desarrollo de aplicaciones que requerirán obligatoriamente agregar nuevas prestaciones, lo que conlleva un desafío en cada paso, ya que se debe siempre pensar en como, cada decisión afectará y limitará las nuevas prestaciones, para ello es necesario tener presente un plan de

desarrollo inremental, mediante pruebas y análisis sobre la aplicabilidad en distintas configuraciones, con el fin de orientar y regular su desarrollo para utilizar mejor los recursos, en especial de horas de trabajo. Es por ello que las elecciones antes mencionadas siempre se estipulan como elecciones temporales y que en cualquier momento pueden ser cambiadas, principalmente luego de realizar pruebas que demuestren un mal funcionamiento o que no entreguen los resultados esperados.

6.2 Herramientas para el desarrollo

Dentro de las herramientas necesarias para el desarrollo, existe el elemento primordial del lenguaje de programación que se adecue a las características del problema. Principalmente relacionado a características de comunicación y desarrollo de aplicaciones de resolución de problemas complejos.

En los primeros desarrollos se utilizó el lenguaje de programación C, debido a la experiencia personal y las características de buen funcionamiento en resolución de problemas matemáticos y buenas prestaciones de comunicación.

Un aspecto técnico que se requirió utilizar en el proceso de administración de la comunicación, es el elemento del “socket”, que es la herramienta que contiene la combinación de una dirección IP (la dirección numérica única de cuatro partes que identifica a un computador en particular) y un número de puerto (el número que identifica una aplicación en particular que tiene acceso al puerto), que es utilizado para poder comunicarse entre diversas aplicaciones, la justificación de su uso, se debe a la potencialidad de ser utilizado en una red de procesos paralelos, donde provee la capacidad de intercambiar flujos de datos, generalmente de manera confiable, ordenada y administrada, además tiene las características de poder ser configurado para que tenga distintas características y formas de comunicación (protocolos).

El modo de utilización que se hace de los socket es de manera iterativa, cabe mencionar que dicha forma de trabajo no fue la única opción evaluada, ya que también se hicieron pruebas utilizando thread o socket concurrentes.

Para que se lleve a cabo la comunicación mediante socket es necesario que el proceso cumpla con la característica de que una aplicación sea capaz de localizar a otra de manera remota y que puedan intercambiar datos.

La creación de socket necesita de ciertas características o recursos: Un protocolo de comunicaciones, que permita el intercambio (las aplicaciones funcionan en mediante el protocolo TCP); una dirección IP, que identifique un computador (en el caso del presente proyecto se desarrolla un servidor, de manera local, que puede ser identificado mediante su dirección IP, dicha dirección puede ser de red local, Localhost o dirección de Internet).; y un puerto, que debe estar disponible, que en el diseño lógico se fijo en el puerto 8888. Que cumple con las características de no estar reservado para ningún tipo de servicio.

6.3 Requerimientos

El funcionamiento de las aplicaciones, requirieron de ciertas capacidades de hardware especialmente de Memoria Ram, donde la carga de datos y especialmente el almacenamiento de información en los buffers de cada socket aumenta en gran medida la carga de trabajo de la memoria y el consecuente aumento o retraso en el tiempo de proceso.

En cuanto al uso del procesador, este elemento esta en estrecha relación con las características del lenguaje de programación ya que se crean aplicaciones que no exigen en demasía la capacidad de los procesadores actuales. El tipo de procesador que se utiliza en las pruebas debe presentar una capacidad de procesamiento sobre 1.4Ghz, aunque con menores capacidades se obtienen desarrollos igualmente aceptables.

Disponer de una conexión de red que tenga la capacidad de poder aceptar procesos que administren o accedan a equipos ubicados dentro de la red y que sean identificados mediante direcciones IP. Cuyos quipos deben tener la capacidad de poder liberar el uso de ciertos puertos de comunicación.

Consideraciones de seguridad, se requiere además configurar aspectos de seguridad de equipos y redes, en cuanto a estas últimas, es permitir las conexiones dinámicas entre equipos, ya que cada aplicación puede crear procesos de comunicación cuando se estime conveniente. Sobre los equipos es necesario determinar algunos permisos de funcionamiento de las aplicaciones por parte de antivirus y barreras de seguridad del sistema operativo,

especialmente con sistemas operativos Windows (Firewall de Windows) esto se debe a las características de funcionamiento de las aplicaciones mediante comunicaciones intensas y que el firewall lo reconoce como posibles ataques o virus que están tratando de acceder a datos del equipo.

6.4 Prototipos

Los prototipos desarrollados se basaron en distintas versiones de software con el objetivo de verificar investigaciones e investigar nuevas características.

El primer prototipo fue desarrollado con las siguientes características: Resolución de un problema de optimización de Cobertura de Conjuntos. (SCP) mediante el algoritmo Ant Colony System (ACS), con una estructura de distribución de los procesos mediante Colonia Central o Administradora y forma de comunicación síncrona, que se establece mediante un parámetro de entrada al sistema.

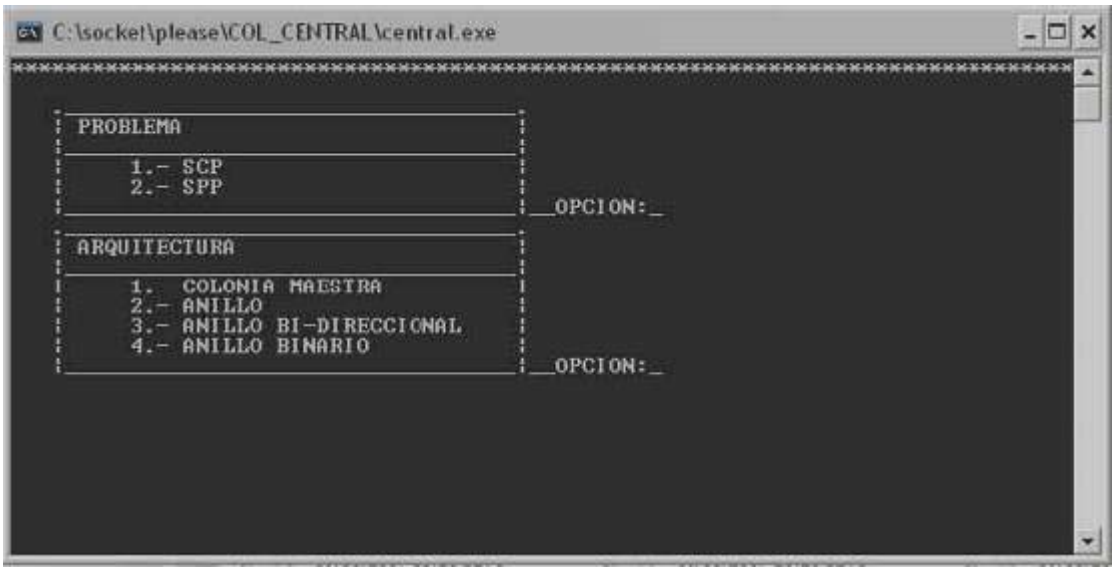


Ilustración 6.1: Pantalla de Inicio en Colonia Administradora

Como se puede observar en la ilustración anterior, el prototipo inicial ya pretendía abarcar muchas mas áreas, las cuales claramente estaban deshabilitadas. Por lo mismo se desarrolló otra estructura de distribución de las colonias, específicamente de anillo. A continuación se muestran imágenes tanto de configuración de administrador como de vecindad de anillo.

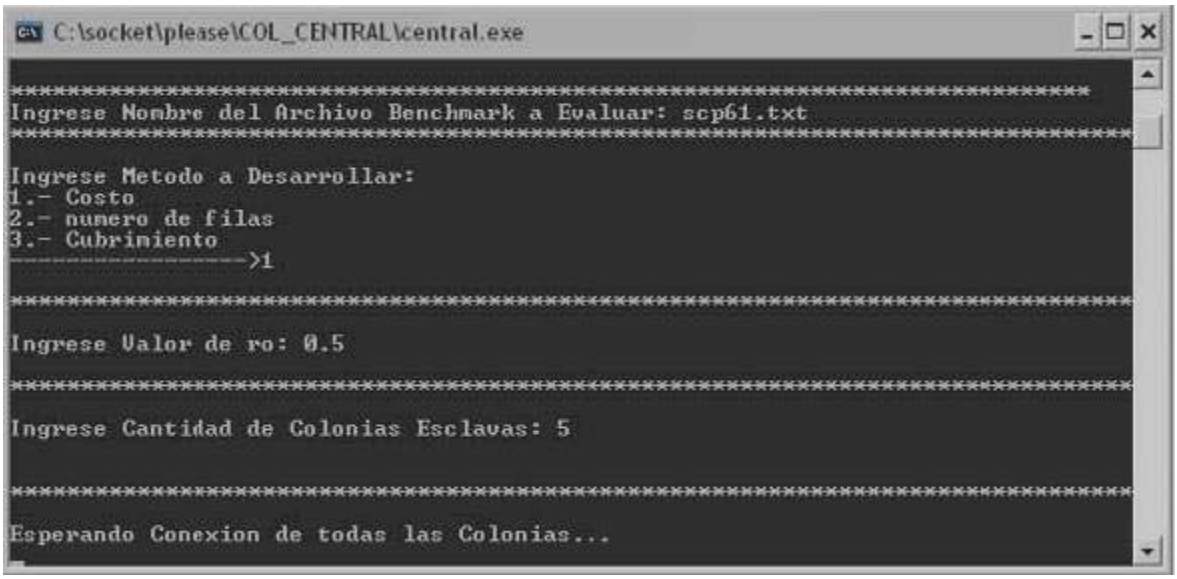


Ilustración 6.2: Pantalla Ingreso de Parámetros Colonia Administradora.

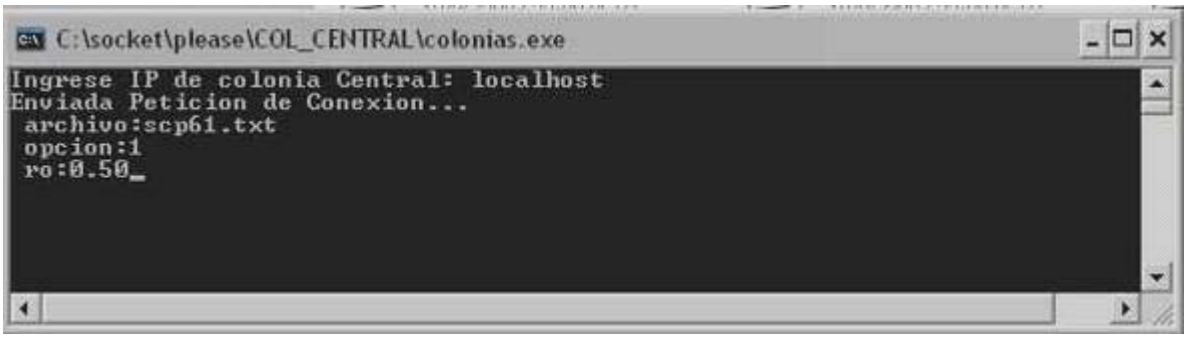


Ilustración 6.3: Pantalla de Inicio Colonia Buscadora.

Una vez creada la colonia administradora y las todas colonias buscadoras (Figuras anteriores) determinadas por la cantidad estipulada en la colonia administradora, aparecerá una ventana con un mensaje en el equipo de la colonia administradora

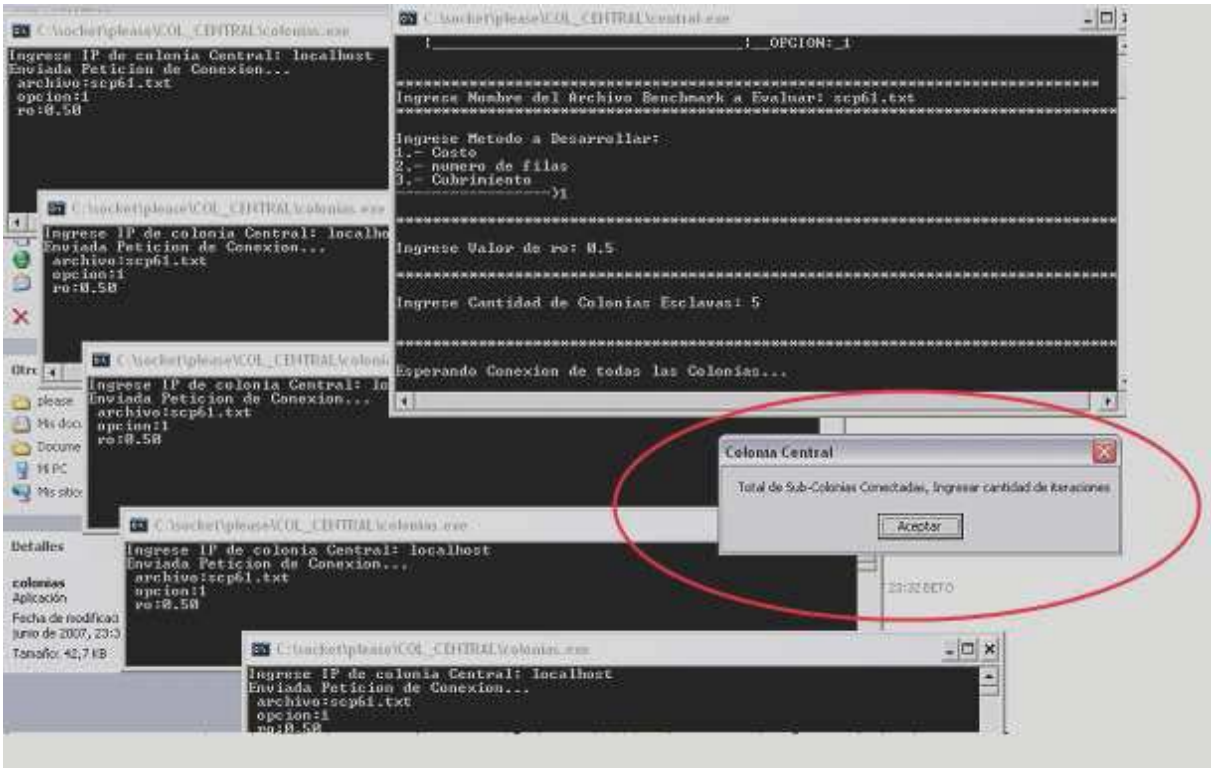


Ilustración 6.4: Pantalla de Aviso de Conexiones Realizadas.

Luego de aceptado el mensaje, la aplicación de colonia administradora, el usuario de la colonia deberá indicar el intervalo de migración o ciclo de iteraciones, es decir, cada cuantas iteraciones se enviarán los resultados de cada colonia buscadora a la colonia administradora.

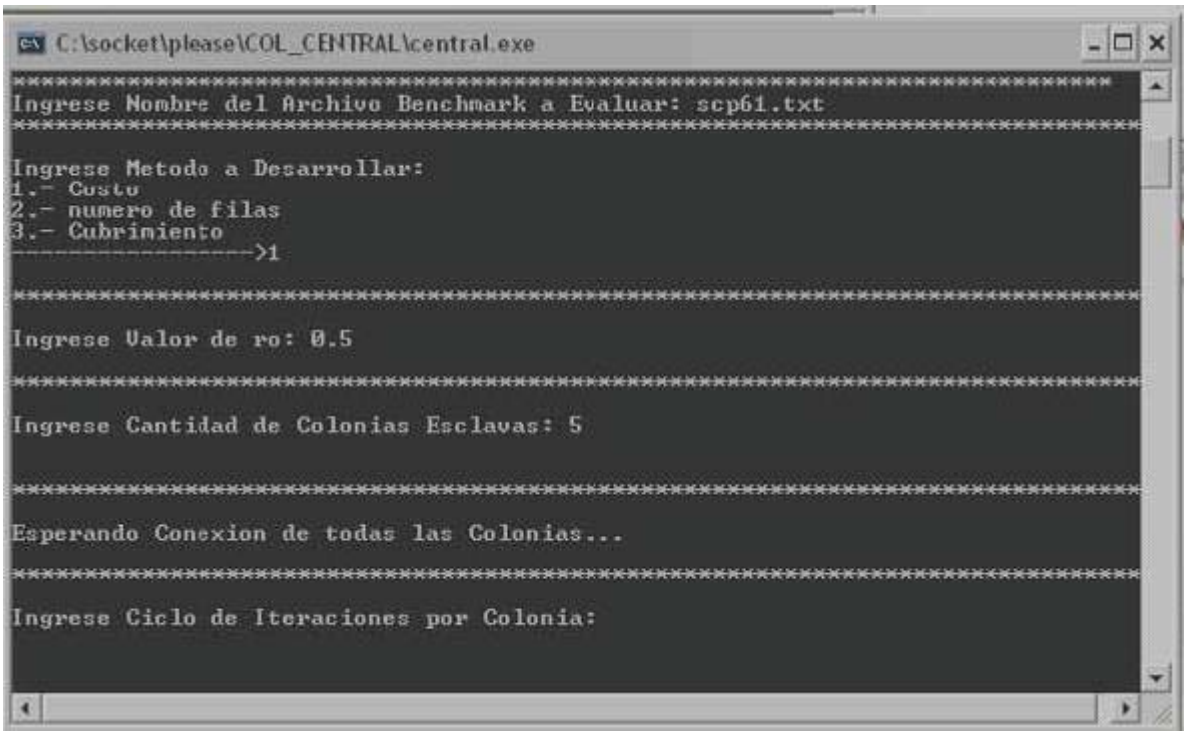


Ilustración 6.5: Pantalla Ingreso de Intervalo de Migración.

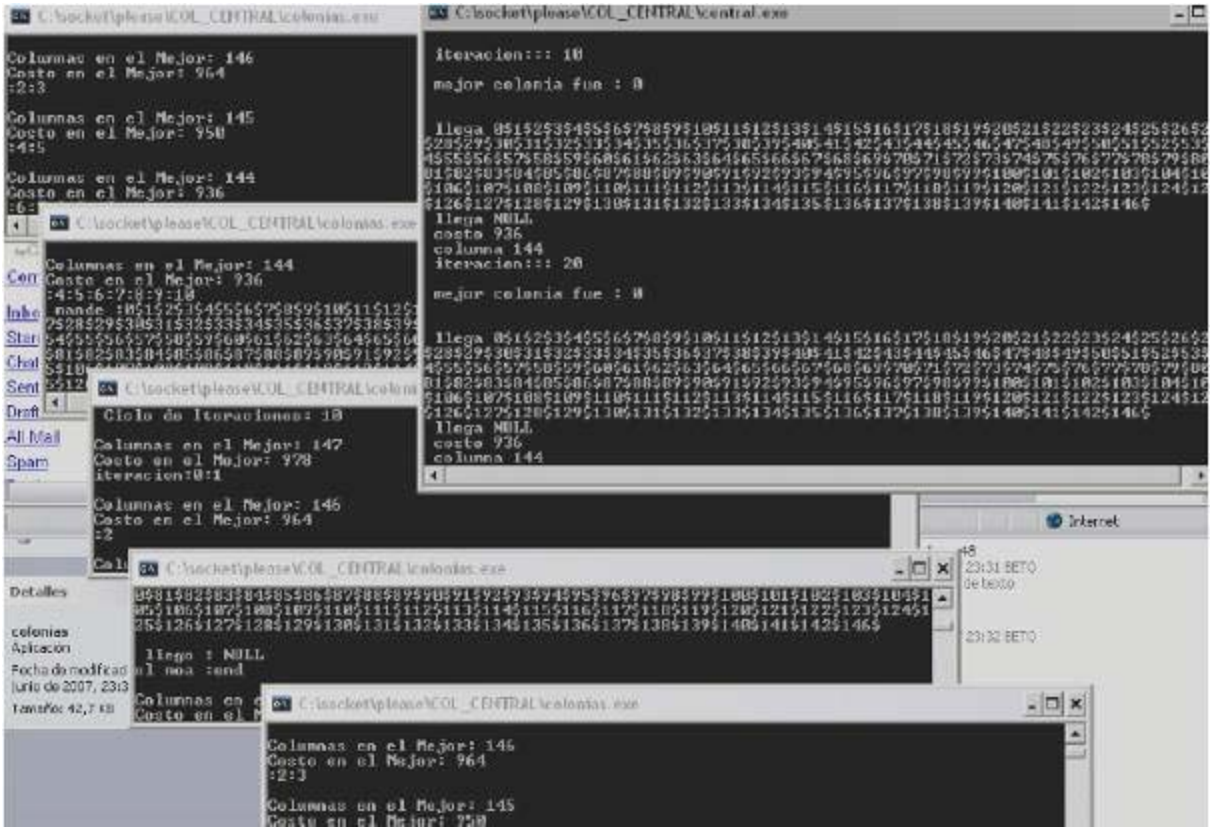


Ilustración 6.6: Pantalla Desarrollo de Búsqueda.

En la última figura se aprecia como se lleva a cabo el proceso en cada aplicación.

Para el segundo prototipo se mantienen las opciones de resolución de un problema de optimización de Cobertura de Conjuntos (SCP) mediante el algoritmo Ant Colony System (ACS), pero con una estructura de distribución de los procesos mediante vecindad de colonias en forma de anillo.

Dadas las características del proceso, cada colonia trabaja sobre los mismos parámetros, y de la misma manera en que se ven las interfaces anteriores, solo que internamente la comunicación se lleva a cabo de manera distinta y a cada colonia se le envía un número identificador.

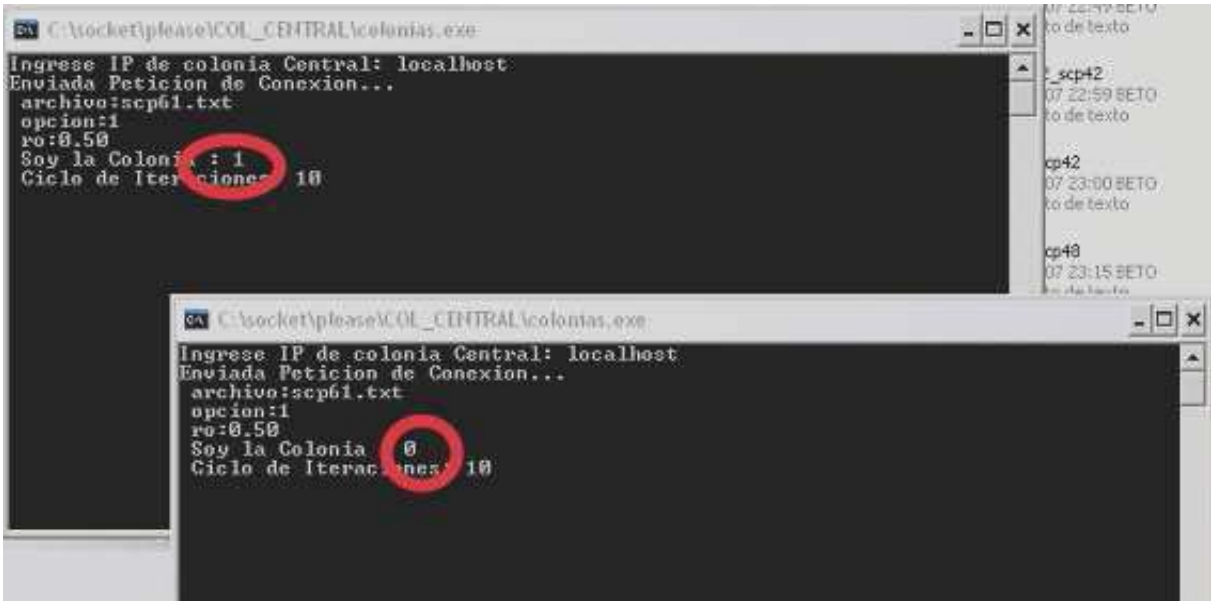


Ilustración 6.7: Pantalla Colonias Buscadoras en Vecindad.

Las evaluaciones sobre el funcionamiento del sistema, se desarrollaron en función de la resolución de problemas benchmark y las estructuras de distribución fijadas en el diseño del desarrollo.

6.5 Pruebas Administrador-Buscador

En primer lugar se analizaron las pruebas realizadas sobre la estructura Aplicación Administradora Central y una serie de aplicaciones de colonias clonadas, el sistema fue probado de 3 maneras principales:

Mediante el uso de una sola aplicación del algoritmo, se utilizó un solo computador con procesador AMD Duron 1,8 Ghz, 376 MB RAM, Sistema operativo Windows XP Service Pack 2, la prueba se realizó con el objetivo de tener un punto de referencia o comparación sobre ejecuciones individuales contra ejecuciones en paralelo.

Mediante el uso de un solo computador, en el cual se ejecutaron la colonia administradora y varias colonias esclavas resolutivas que se ejecutaron 6 veces, las características del computador es un procesador AMD Duron 1,8 Ghz, 376 MB RAM, Sistema operativo Windows XP Service Pack 2, Firewall con manejo de excepciones de aplicación “maestra” y “esclava”.

Mediante el uso de una red, específicamente utilizando Internet, donde se realizaron las mismas pruebas, las cuales se desarrollaron mediante la utilización de cinco computadores, los cuales se ubicaban en lugares distintos geográficamente, que se conectaban a través de Internet. El tipo de computadores que se utilizaron fueron distintos y con capacidades similares, donde el equipo con menores capacidades fue un Intel Celeron de 1,2 Ghz, 128 MB de Ram con Windows XP, mientras que el de mayor capacidades fue un Pentium 4 de 2.0 Ghz, con 512 MB de Ram.

6.5.1 Análisis de las Pruebas

Las pruebas fueron realizadas utilizando problemas de prueba extraídos de ORLIB de Beasley [34]. Las características principales fueron las siguientes:

El resultado de las pruebas en paralelo mostró gran similitud con las pruebas realizadas al algoritmo de forma individual, donde en los problemas desarrollados se observo una pequeña mejoría, pero que al momento de la evaluación sobre el total de pruebas resulta que claramente los resultados muestran la dependencia sobre el funcionamiento del algoritmo.

Existencia de retardo por coordinación, el cual consiste en la espera que debían realizar algunas colonias que desarrollaron su búsqueda mucho más rápido, debiendo quedarse esperando que terminaran las demás colonias para poder continuar su proceso debido a la naturaleza síncrona del proceso.

Uso intensivo de memoria y procesador, estos se dieron a conocer al momento de realizar pruebas de la aplicación con funcionamiento en paralelo pero desarrolladas en un solo equipo, donde claramente se produce un mayor uso de dichos recursos, provocando que algunas colonias esclavas se retrasaran en su desarrollo y retrasaran en un grado importante el proceso, tal como lo justifica el punto anterior. Sobre el desarrollo de manera distribuida el uso de memoria y procesador no fue considerable.

Utilización de redes y puertos de comunicación, estos elementos mostraron un uso intensivo en el desarrollo de la aplicación distribuida del proceso de búsqueda en distintos procesos, que influyo altamente en el desempeño de las pruebas, donde el ejemplo más claro fue en el uso de los buffer de los puertos de comunicación, donde la

intensiva comunicación provocó la saturación produciendo en algunos casos la perdida de datos y la consecuente caída de la prueba.

Tabla 6.1: Resumen de Resultados Administrador-Buscador.

ADMINISTRADOR – BUSCADOR					
Problema	**Filas**	**Columnas**	**Optimo**	**ACS**	**ACS Paralelo**
Scp41	200	1000	429	463	453
Scp42	200	1000	512	590	565
Scp48	200	1000	492	522	514
Scp61	200	1000	138	154	152
Scp62	200	1000	146	163	161
Scp63	200	1000	145	157	155

En resumen, la configuración que se buscaba probar en paralelo y distribuida en varios equipos, no mostró los resultados esperados y optimistas sobre el desarrollo, donde la saturación de los buffers de los puertos de entrada provocó una reducción de la velocidad y caídas inesperadas de las pruebas.

En lo que respecta a la calidad de la solución tampoco mostró grandes resultados, donde podemos ver en la tabla anterior un cuadro resumen de los resultados obtenidos, donde lo más relevante que se observa es la similitud en cuanto a los resultados en paralelo con los resultados de la ejecución de manera independiente, es decir, el comportamiento del sistema en paralelo se puede comparar con el mejor de las ejecuciones individuales del algoritmo, lo cual claramente no era algo esperado ni objetivo del desarrollo.

Además cabe señalar que si bien los resultados fueron satisfactorios en el punto del mejoramiento de la calidad de la solución, lo que se busca es una buena relación calidad versus tiempo de desarrollo, lo que como se comento anteriormente no se logró por el retardo e inestabilidad de los procesos de comunicación.

6.5.2 Detalle de las Pruebas

A continuación se muestran en detalle algunos de los procesos de desarrollo de la solución:

Tabla 6.2: Resultados Prueba Colonias Administrador-Buscador (SCP41; Ro:0,9; Q0:0,2)

Problema	**SCP41**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50
Columnas	1000	**Ro**	0.9
Optimo	426	**Qo**	0.2
ACS	463	**Colonias**	3

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Mejor	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**
1	476	499	476	476	508	476	483	514	483
2	469	500	469	463	499	463	470	499	470
3	460	489	460	473	497	463	467	504	467

4	460	487	460	460	492	460	459	501	459
5	460	493	460	457	506	457	460	502	459
6	459	496	459	460	490	457	456	501	456
7	457	493	457	457	490	457	457	493	456
8	457	497	457	457	497	457	456	489	456
9	457	489	457	457	498	457	457	499	456
10	457	496	457	457	516	457	456	494	456
11	457	492	457	457	497	457	457	499	456
12	456	494	456	457	500	457	456	496	456
13	457	499	456	457	494	457	457	504	456
14	456	496	456	456	494	456	456	496	456
15	456	494	456	456	496	456	456	501	456
16	456	509	456	456	507	456	456	498	456
17	456	493	456	456	493	456	456	487	456
18	456	507	456	456	497	456	456	503	456
19	455	506	455	456	498	456	456	507	456
20	456	495	455	454	500	454	454	492	454
21	455	497	455	455	506	454	455	490	454
22	455	491	455	454	511	454	454	502	454
23	455	500	455	455	498	454	455	494	454

24	455	498	455	454	492	454	454	493	454
25	455	497	455	455	494	454	455	492	454
26	455	488	455	454	495	454	454	494	454
27	455	491	455	455	501	454	455	489	454
28	455	501	455	454	509	454	454	494	454
29	454	499	454	455	507	454	455	505	454
30	455	500	454	454	487	454	453	508	453
31	454	484	454	454	489	454	454	496	453
32	454	495	454	453	496	453	453	492	453
33	454	492	454	454	505	453	454	498	453
34	454	500	454	453	500	453	453	505	453
35	454	481	454	454	488	453	454	498	453
36	454	482	454	453	502	453	453	500	453
37	454	490	454	454	493	453	454	482	453
38	454	513	454	453	502	453	453	483	453
39	453	506	453	454	498	453	454	502	453
40	454	501	453	453	495	453	453	500	453
41	453	490	453	453	489	453	453	502	453
42	453	499	453	453	496	453	453	499	453
43	453	504	453	453	489	453	453	497	453

44	453	494	453	453	491	453	453	496	453
45	453	497	453	453	500	453	453	508	453
46	453	493	453	453	505	453	453	494	453
47	453	495	453	453	503	453	453	497	453
48	453	506	453	453	501	453	453	494	453
49	453	496	453	453	492	453	453	492	453
50	453	490	453	453	503	453	453	495	453
RESULTADO	453								

Tabla 6.3: Resultados Prueba Colonias Administrador-Buscador (SCP41; Ro:0,8; Q0:0,2)

Problema	**SCP41**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50
Columnas	1000	**Ro**	0.8
Optimo	426	**Qo**	0.2
ACS	463	**Colonias**	3

	Colonias		
ITERACIONES	**Col 1**	**Col 2**	**Col 3**

	Mejor	Peor	Acumulado	Mejor	Peor	Acumulado	Mejor	Peor	Acumulado
1	460	478	460	462	484	462	459	480	459
2	460	488	460	461	477	461	463	482	459
3	458	460	458	459	462	459	456	459	456
4	458	460	458	459	462	459	456	459	456
5	456	460	456	459	462	459	456	459	456
6	458	460	456	459	462	459	456	459	456
7	456	458	456	459	459	459	456	456	456
8	456	456	456	459	459	459	456	456	456
9	456	456	456	459	459	459	456	456	456
10	456	456	456	459	459	459	456	456	456
11	456	456	456	459	459	459	456	456	456
12	456	456	456	459	459	459	456	456	456
13	456	456	456	459	459	459	456	456	456
14	456	456	456	459	459	459	456	456	456
15	456	456	456	459	459	459	456	456	456
16	456	456	456	459	459	459	456	456	456
17	456	456	456	459	459	459	456	456	456
18	456	456	456	459	459	459	456	456	456
19	456	456	456	459	459	459	456	456	456

20	456	456	456	459	459	459	456	456	456
21	456	456	456	459	459	459	456	456	456
22	456	456	456	459	459	459	456	456	456
23	456	456	456	459	459	459	456	456	456
24	456	456	456	459	459	459	456	456	456
25	456	456	456	459	459	459	456	456	456
26	456	456	456	459	459	459	456	456	456
27	456	456	456	459	459	459	456	456	456
28	456	456	456	459	459	459	456	456	456
29	456	456	456	459	459	459	456	456	456
30	456	456	456	459	459	459	456	456	456
31	456	456	456	459	459	459	456	456	456
32	456	456	456	459	459	459	456	456	456
33	456	456	456	459	459	459	456	456	456
34	456	456	456	459	459	459	456	456	456
35	456	456	456	459	459	459	456	456	456
36	456	456	456	459	459	459	456	456	456
37	456	456	456	459	459	459	456	456	456
38	456	456	456	459	459	459	456	456	456
39	456	456	456	459	459	459	456	456	456

40	456	456	456	459	459	459	456	456	456
41	456	456	456	459	459	459	456	456	456
42	456	456	456	459	459	459	456	456	456
43	456	456	456	459	459	459	456	456	456
44	456	456	456	459	459	459	456	456	456
45	456	456	456	459	459	459	456	456	456
46	456	456	456	459	459	459	456	456	456
47	456	456	456	459	459	459	456	456	456
48	456	456	456	459	459	459	456	456	456
49	456	456	456	459	459	459	456	456	456
50	456	456	456	459	459	459	456	456	456
RESULTADO	456								

Tabla 6.4: Resultados Prueba Colonias Administrador-Buscador (SCP41; Ro:0,5; Q0:0,5)

Problema	**SCP41**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50
Columnas	1000	**Ro**	0.5
Optimo	426	**Qo**	0.5
ACS	463	**Colonias**	3

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Mejor	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**
1	463	487	463	466	500	466	462	493	462
2	459	487	459	460	492	460	461	490	461
3	460	463	459	460	463	460	458	464	458
4	458	463	458	456	486	456	453	453	453
5	456	456	456	460	460	456	453	459	453
6	456	458	456	456	461	456	453	454	453
7	453	456	453	453	456	453	453	453	453
8	453	456	453	453	458	453	453	465	453
9	453	456	453	453	467	453	453	453	453
10	453	453	453	453	459	453	453	453	453
11	453	463	453	453	456	453	453	458	453
12	453	454	453	453	453	453	453	458	453
13	453	459	453	453	455	453	453	453	453
14	453	456	453	453	453	453	453	453	453
15	453	453	453	453	453	453	453	453	453

16	453	455	453	453	453	453	453	465	453
17	453	453	453	453	453	453	453	453	453
18	453	455	453	453	453	453	453	453	453
19	453	453	453	453	453	453	453	453	453
20	453	453	453	453	456	453	453	453	453
21	453	453	453	453	456	453	453	465	453
22	453	453	453	453	456	453	453	456	453
23	453	465	453	453	459	453	453	453	453
24	453	463	453	453	453	453	453	458	453
25	453	453	453	453	455	453	453	453	453
26	453	453	453	453	453	453	453	453	453
27	453	453	453	453	453	453	453	465	453
28	453	456	453	453	453	453	453	453	453
29	453	453	453	453	455	453	453	465	453
30	453	456	453	453	456	453	453	453	453
31	453	455	453	453	453	453	453	453	453
32	453	453	453	453	453	453	453	453	453
33	453	465	453	453	454	453	453	453	453
34	453	453	453	453	453	453	453	453	453
35	453	453	453	453	455	453	453	455	453

36	453	453	453	453	463	453	453	453	453
37	453	453	453	453	455	453	453	453	453
38	453	465	453	453	453	453	453	453	453
39	453	453	453	453	456	453	453	453	453
40	453	465	453	453	453	453	453	465	453
41	453	453	453	453	453	453	453	453	453
42	453	455	453	453	455	453	453	453	453
43	453	454	453	453	453	453	453	453	453
44	453	453	453	453	456	453	453	453	453
45	453	453	453	453	454	453	453	453	453
46	453	453	453	453	455	453	453	453	453
47	453	455	453	453	453	453	453	453	453
48	453	455	453	453	453	453	453	453	453
49	453	453	453	453	453	453	453	453	453
50	453	465	453	453	455	453	453	453	453
RESULTADO	453								

Tabla 6.5: Resultados Prueba Colonias Administrador-Buscador (SCP42; Ro:0,9; Q0:0,2)

Problema	**SCP42**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50

Columnas	1000	**Ro**	0.9
Optimo	426	**Qo**	0.2
ACS	463	**Colonias**	3

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Mejor	Peor	Acumulado	Mejor	Peor	Acumulado	Mejor	Peor	Acumulado
1	637	681	637	604	676	604	600	684	600
2	618	678	618	604	690	604	627	689	600
3	603	697	603	593	656	593	591	669	591
4	601	676	601	586	684	586	591	687	591
5	586	687	586	586	646	586	587	675	587
6	586	688	586	582	673	582	585	669	585
7	581	674	581	579	656	579	592	668	585
8	581	638	581	579	680	579	582	674	582
9	580	666	580	574	651	574	588	660	582
10	580	680	580	571	690	571	583	676	582
11	580	683	580	574	673	571	585	662	582
12	577	689	577	571	661	571	583	660	582

13	571	692	571	571	657	571	577	678	577
14	571	668	571	571	685	571	577	668	577
15	571	679	571	571	685	571	575	681	575
16	571	668	571	571	656	571	574	680	574
17	571	668	571	571	688	571	577	677	574
18	570	673	570	571	680	571	577	666	574
19	571	672	570	571	696	571	577	679	574
20	570	671	570	571	685	571	574	642	574
21	570	683	570	571	661	571	574	658	574
22	570	663	570	570	640	570	576	674	574
23	570	703	570	570	675	570	576	680	574
24	570	657	570	570	650	570	576	664	574
25	570	675	570	570	656	570	571	706	571
26	569	647	569	570	660	570	576	659	571
27	570	672	569	569	670	569	573	663	571
28	569	673	569	569	668	569	573	677	571
29	569	698	569	569	644	569	575	668	571
30	569	684	569	569	681	569	575	665	571
31	569	661	569	569	665	569	572	693	571
32	569	674	569	569	668	569	573	663	571

33	569	665	569	569	671	569	572	688	571
34	569	651	569	569	667	569	575	666	571
35	569	678	569	569	663	569	572	671	571
36	569	670	569	569	637	569	575	664	571
37	569	649	569	569	667	569	570	685	570
38	569	671	569	569	678	569	575	671	570
39	569	673	569	569	643	569	573	688	570
40	569	673	569	569	687	569	573	654	570
41	569	645	569	569	689	569	572	673	570
42	569	658	569	569	663	569	575	671	570
43	569	676	569	569	665	569	572	673	570
44	569	673	569	569	665	569	575	656	570
45	569	646	569	569	666	569	575	658	570
46	569	687	569	569	665	569	572	681	570
47	569	656	569	569	676	569	572	682	570
48	569	697	569	569	664	569	573	668	570
49	569	673	569	569	670	569	575	657	570
50	569	662	569	569	649	569	573	668	570
RESULTADO	569								

Tabla 6.6: Resultados Prueba Colonias Administrador-Buscador (SCP42; Ro:0,8; Q0:0,2)

Problema	**SCP42**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50
Columnas	1000	**Ro**	0.8
Optimo	426	**Qo**	0.2
ACS	463	**Colonias**	3

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Mejor	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**
1	580	618	580	580	595	580	580	619	580
2	582	608	580	582	605	580	581	601	580
3	577	580	577	577	580	577	577	580	577
4	577	580	577	577	580	577	577	580	577
5	577	580	577	577	577	577	577	577	577
6	577	577	577	577	577	577	577	580	577
7	577	577	577	577	577	577	577	577	577
8	577	577	577	577	577	577	577	577	577
9	577	577	577	577	577	577	577	577	577
10	577	577	577	577	577	577	577	577	577

11	577	577	577	577	577	577	577	577	577
12	577	580	577	577	577	577	577	577	577
13	577	577	577	577	577	577	577	577	577
14	577	577	577	577	577	577	577	577	577
15	577	577	577	577	577	577	577	577	577
16	577	577	577	577	577	577	577	577	577
17	577	577	577	577	577	577	577	577	577
18	577	577	577	577	577	577	577	577	577
19	577	577	577	577	577	577	577	577	577
20	577	577	577	577	577	577	577	577	577
21	577	577	577	577	577	577	577	577	577
22	577	577	577	577	577	577	577	577	577
23	577	577	577	577	577	577	577	577	577
24	577	577	577	577	577	577	577	577	577
25	577	577	577	577	577	577	577	577	577
26	577	577	577	577	577	577	577	577	577
27	577	577	577	577	577	577	577	577	577
28	577	577	577	577	577	577	577	577	577

29	577	577	577	577	577	577	577	577	577
30	577	577	577	577	577	577	577	577	577
31	577	577	577	577	577	577	577	577	577
32	577	577	577	577	577	577	577	577	577
33	577	577	577	577	577	577	577	577	577
34	577	577	577	577	577	577	577	577	577
35	577	577	577	577	577	577	577	577	577
36	577	577	577	577	577	577	577	577	577
37	577	577	577	577	577	577	577	577	577
38	577	577	577	577	577	577	577	577	577
39	577	577	577	577	577	577	577	577	577
40	577	577	577	577	577	577	577	577	577
41	577	577	577	577	577	577	577	577	577
42	577	577	577	577	577	577	577	577	577
43	577	577	577	577	577	577	577	577	577
44	577	577	577	577	577	577	577	577	577
45	577	577	577	577	577	577	577	577	577
46	577	577	577	577	577	577	577	577	577
47	577	577	577	577	577	577	577	577	577
48	577	577	577	577	577	577	577	577	577

49	577	577	577	577	577	577	577	577	577
50	577	577	577	577	577	577	577	577	577
RESULTADO	577								

Tabla 6.7: Resultados Prueba Colonias Administrador-Buscador (SCP42; Ro:0,5; Q0:0,5)

Problema	**SCP42**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50
Columnas	1000	**Ro**	0.5
Optimo	426	**Qo**	0.5
ACS	463	**Colonias**	3

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Mejor	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**
1	603	648	603	590	667	590	594	651	594
2	584	629	584	586	667	586	585	649	585
3	579	594	579	588	592	586	582	615	582
4	575	590	575	583	597	583	578	588	578

5	574	583	574	585	587	583	578	587	578
6	574	582	574	580	584	580	578	585	578
7	574	581	574	579	580	579	578	583	578
8	574	579	574	575	592	575	574	581	574
9	574	579	574	572	576	572	574	579	574
10	574	574	574	572	580	572	574	614	574
11	574	579	574	572	597	572	574	574	574
12	574	574	574	572	595	572	574	579	574
13	574	578	574	572	575	572	574	582	574
14	574	574	574	572	581	572	574	577	574
15	574	574	574	572	576	572	567	578	567
16	574	574	574	572	572	572	567	583	567
17	567	578	567	565	572	565	567	581	567
18	567	577	567	565	572	565	567	580	567
19	567	574	567	565	569	565	567	574	567
20	567	567	567	565	573	565	567	571	567
21	567	567	567	565	565	565	567	580	567
22	567	576	567	565	565	565	567	567	567
23	567	574	567	565	596	565	567	567	567
24	567	567	567	565	565	565	567	567	567

25	567	567	567	565	565	565	567	574	567
26	567	576	567	565	565	565	567	571	567
27	567	570	567	565	565	565	567	580	567
28	567	571	567	565	592	565	567	567	567
29	567	574	567	565	595	565	567	574	567
30	567	571	567	565	572	565	567	567	567
31	567	575	567	565	565	565	567	574	567
32	567	567	567	565	565	565	567	567	567
33	567	567	567	565	592	565	567	567	567
34	567	570	567	565	565	565	567	567	567
35	567	567	567	565	565	565	567	567	567
36	567	567	567	565	569	565	567	574	567
37	567	567	567	565	565	565	567	567	567
38	567	567	567	565	570	565	567	571	567
39	567	576	567	565	569	565	567	567	567
40	567	567	567	565	566	565	567	569	567
41	567	567	567	565	565	565	567	567	567
42	567	567	567	565	565	565	567	567	567
43	567	574	567	565	572	565	567	574	567
44	567	571	567	565	572	565	567	567	567

45	567	567	567	565	565	565	567	567	567
46	567	567	567	565	565	565	567	567	567
47	567	571	567	565	565	565	567	570	567
48	567	571	567	565	565	565	567	574	567
49	567	567	567	565	595	565	567	580	567
50	567	567	567	565	565	565	567	583	567
RESULTADO	565								

Tabla 6.8: Resultados Prueba Colonias Administrador-Buscador (SCP48; Ro:0,9; Q0:0,1)

Problema	**SCP48**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50
Columnas	1000	**Ro**	0.9
Optimo	426	**Qo**	0.1
ACS	463	**Colonias**	3

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Mejor	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**
1	567	600	567	550	589	550	565	607	565

2	555	602	555	552	606	550	567	597	565
3	548	598	548	538	601	538	552	601	552
4	534	603	534	543	585	538	547	581	547
5	540	603	534	537	576	537	552	607	547
6	534	583	534	537	581	537	536	598	536
7	534	577	534	539	593	537	540	611	536
8	528	608	528	528	583	528	528	589	528
9	532	583	528	528	590	528	534	587	528
10	528	603	528	528	585	528	528	597	528
11	528	603	528	526	581	526	526	613	526
12	526	586	526	528	574	526	528	611	526
13	528	592	526	528	571	526	528	596	526
14	526	587	526	528	589	526	528	588	526
15	526	580	526	528	587	526	528	594	526
16	526	602	526	528	588	526	528	583	526
17	526	590	526	526	593	526	526	583	526
18	526	586	526	526	595	526	526	586	526
19	526	578	526	526	600	526	526	595	526
20	526	579	526	526	587	526	526	610	526
21	526	586	526	526	591	526	526	578	526

22	526	613	526	526	600	526	526	602	526
23	526	600	526	526	600	526	523	602	523
24	526	591	526	526	601	526	526	591	523
25	526	597	526	526	600	526	523	598	523
26	526	594	526	526	612	526	523	594	523
27	526	610	526	526	608	526	523	616	523
28	526	577	526	526	599	526	526	573	523
29	526	589	526	526	588	526	523	587	523
30	526	583	526	526	589	526	523	589	523
31	526	601	526	526	587	526	523	595	523
32	526	591	526	526	587	526	523	589	523
33	526	573	526	526	598	526	526	577	523
34	526	590	526	526	593	526	523	582	523
35	526	579	526	526	590	526	523	591	523
36	526	587	526	526	578	526	523	579	523
37	526	584	526	526	576	526	523	595	523
38	526	585	526	526	600	526	526	575	523
39	526	588	526	526	585	526	523	587	523
40	526	602	526	526	596	526	523	588	523
41	526	592	526	526	575	526	523	597	523

42	523	578	523	523	602	523	523	581	523
43	523	587	523	523	599	523	523	588	523
44	523	572	523	523	602	523	523	582	523
45	523	579	523	523	572	523	523	586	523
46	523	593	523	523	596	523	521	616	521
47	523	584	523	523	578	523	523	584	521
48	523	585	523	523	586	523	523	595	521
49	523	575	523	523	618	523	523	602	521
50	523	581	523	523	589	523	523	586	521
RESULTADO	521								

Tabla 6.9: Resultados Prueba Colonias Administrador-Buscador (SCP48; Ro:0,1; Q0:0,9)

Problema	**SCP48**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50
Columnas	1000	**Ro**	0.1
Optimo	426	**Qo**	0.9
ACS	463	**Colonias**	3

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Mejor	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**
1	533	543	533	530	550	530	530	553	530
2	533	547	533	533	546	530	533	548	530
3	533	533	533	530	530	530	525	525	525
4	533	533	533	526	530	526	525	525	525
5	533	533	533	530	530	526	521	525	521
6	533	533	533	530	530	526	521	525	521
7	533	533	533	526	526	526	525	525	521
8	533	533	533	526	530	526	521	525	521
9	533	533	533	526	530	526	521	521	521
10	533	533	533	526	526	526	521	521	521
11	533	533	533	521	526	521	521	521	521
12	533	533	533	521	521	521	521	521	521
13	523	533	523	521	521	521	521	521	521
14	523	523	523	521	521	521	521	521	521
15	523	523	523	521	521	521	521	521	521
16	521	523	521	521	521	521	521	521	521
17	521	521	521	521	521	521	521	521	521

18	521	521	521	521	521	521	521	521	521
19	521	521	521	521	521	521	521	521	521
20	521	521	521	521	521	521	521	521	521
21	521	521	521	521	521	521	521	521	521
22	521	521	521	521	521	521	521	521	521
23	521	521	521	521	521	521	521	521	521
24	521	521	521	521	521	521	521	521	521
25	521	521	521	521	521	521	521	521	521
26	521	521	521	521	521	521	521	521	521
27	521	521	521	521	521	521	521	521	521
28	521	521	521	521	521	521	521	521	521
29	521	521	521	521	521	521	521	521	521
30	521	521	521	521	521	521	521	521	521
31	521	521	521	521	521	521	521	521	521
32	521	521	521	521	521	521	521	521	521
33	521	521	521	521	521	521	521	521	521
34	521	521	521	521	521	521	521	521	521
35	521	521	521	521	521	521	521	521	521
36	521	521	521	521	521	521	521	521	521
37	521	521	521	521	521	521	521	521	521

38	521	521	521	521	521	521	521	521	521
39	521	521	521	521	521	521	521	521	521
40	521	521	521	521	521	521	521	521	521
41	521	521	521	521	521	521	521	521	521
42	521	521	521	521	521	521	521	521	521
43	521	521	521	521	521	521	521	521	521
44	521	521	521	521	521	521	521	521	521
45	521	521	521	521	521	521	521	521	521
46	521	521	521	521	521	521	521	521	521
47	521	521	521	521	521	521	521	521	521
48	521	521	521	521	521	521	521	521	521
49	521	521	521	521	521	521	521	521	521
50	521	521	521	521	521	521	521	521	521
RESULTADO	521								

Tabla 6.10: Resultados Prueba Colonias Administrador-Buscador (SCP48; Ro:0,5; Q0:0,5)

Problema	**SCP48**	**Nº Hormigas**	10
Filas	200	**Nº Iteraciones**	50
Columnas	1000	**Ro**	0.5
Optimo	426	**Qo**	0.5

ACS	463	**Colonias**	3

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Mejor	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**	**Mejor**	**Peor**	**Acumulado**
1	544	586	544	552	591	552	546	580	546
2	539	572	539	541	588	541	530	569	530
3	541	547	539	538	552	538	538	546	530
4	531	548	531	532	557	532	526	547	526
5	533	541	531	530	541	530	515	526	515
6	527	542	527	535	538	530	523	526	515
7	526	533	526	527	530	527	514	523	514
8	523	533	523	527	538	527	514	526	514
9	523	537	523	527	535	527	514	525	514
10	523	541	523	527	534	527	514	517	514
11	523	537	523	527	533	527	514	514	514
12	523	524	523	526	527	526	514	523	514
13	523	532	523	526	535	526	514	514	514
14	523	526	523	526	526	526	514	520	514

15	523	523	523	526	526	526	514	514	514
16	523	531	523	526	532	526	514	514	514
17	523	530	523	526	526	526	514	514	514
18	523	523	523	526	526	526	514	522	514
19	523	530	523	526	526	526	514	514	514
20	523	523	523	526	526	526	514	514	514
21	523	539	523	526	526	526	514	514	514
22	523	540	523	526	543	526	514	522	514
23	523	531	523	526	526	526	514	518	514
24	523	523	523	526	540	526	514	514	514
25	523	523	523	526	531	526	514	514	514
26	523	523	523	526	535	526	514	522	514
27	523	523	523	526	526	526	514	514	514
28	523	530	523	526	526	526	514	514	514
29	523	523	523	526	526	526	514	514	514
30	523	523	523	526	530	526	514	514	514
31	523	525	523	526	526	526	514	514	514
32	523	535	523	526	526	526	514	522	514
33	523	533	523	526	526	526	514	514	514
34	523	523	523	526	526	526	514	514	514

35	523	529	523	526	526	526	514	514	514
36	523	523	523	526	534	526	514	514	514
37	523	531	523	526	526	526	514	522	514
38	523	523	523	526	547	526	514	514	514
39	523	530	523	526	526	526	514	514	514
40	523	540	523	526	530	526	514	519	514
41	523	523	523	526	534	526	514	520	514
42	523	523	523	526	526	526	514	514	514
43	523	530	523	526	526	526	514	514	514
44	523	529	523	526	530	526	514	514	514
45	523	529	523	526	534	526	514	514	514
46	523	531	523	526	526	526	514	514	514
47	523	525	523	526	526	526	514	519	514
48	523	538	523	526	531	526	514	514	514
49	523	531	523	526	534	526	514	514	514
50	523	523	523	526	526	526	514	514	514
RESULTADO	514								

6.6 Pruebas Vecindad en Anillo

El segundo proceso de pruebas se realizó sobre una arquitectura en anillo, donde se realizaron las pruebas utilizando los mismos problemas benchmark y configuraciones.

Se realizaron pruebas en las mismas condiciones:

Anillo unitario, es decir una sola aplicación del algoritmo que se comunicaba consigo mismo, se utilizó un solo computador con procesador AMD Duron 1,8 Ghz, 376 MB RAM, Sistema operativo Windows XP Service Pack 2.

Anillo con múltiples eslabones, mediante el uso de un solo computador, en el cual se ejecutaron todos los procesos o colonias pertenecientes al anillo; la cantidad de eslabones fue de 6, las características del computador es un procesador AMD Duron 1,8 Ghz, 376 MB RAM, Sistema operativo Windows XP Service Pack 2, Firewall con manejo de excepciones de aplicación “esclava”.

Implementando el anillo en una red de computadores, utilizando Internet, donde se realizaron las mismas pruebas, las cuales se desarrollaron mediante la utilización de cinco computadores, los cuales se ubicaban en lugares distintos geográficamente, que se conectaban a través de Internet. Los equipos pertenecientes al anillo fueron los mismos que las pruebas en colonias administrador-buscador.

6.6.1 Análisis de las Pruebas

Las características principales del proceso demostraron que a pesar del cambio de arquitectura, se continuó claramente con la dependencia del algoritmo por sobre la arquitectura, repitiéndose el fenómeno de que los resultados obtenidos tienen gran similitud con la ejecución independiente del algoritmo. Inclusive al momento de comprar las dos estructuras, en lo referente al resultado, quedo demostrado que son altamente similares e inclusive podría decirse que iguales en cuanto a la calidad y un poco mejores en velocidad.

Se observo una mayor saturación de buffers en esta arquitectura, esto se debió al mayor flujo de comunicación que se debió realizar durante el proceso, debido a que la condición para el intercambio o mejor dicho la decisión de recibir los datos proveniente de otra colonia solo dependía de una sola comparación, ya que solo importa si la colonia anterior ha realizado una mejor búsqueda, provocando que la mayoría de las colonias se encontrara en el envío o recepción de datos. Provocando además una tasa de caídas de un 20% aproximadamente.

A continuación se muestra una tabla resumen de los resultados obtenidos en dichas pruebas, donde queda clara la igualdad de resultados en cuanto a la calidad.

Tabla 6.11: Resumen de Resultados Colonias en Anillo

COLONIAS EN ANILLO					
Problema	**Filas**	**Columnas**	**Optimo**	**ACS**	**ACS Paralelo**
Scp41	200	1000	429	463	453
Scp42	200	1000	512	590	567
Scp48	200	1000	492	522	515
Scp61	200	1000	138	154	152
Scp62	200	1000	146	163	161
Scp63	200	1000	145	157	155

6.7 Conclusiones de las Pruebas

En resumen fue posible observar tras el período de pruebas, las siguientes características generales:

Los resultados de las pruebas muestran que el uso en paralelo del algoritmo es similar a las pruebas realizadas independientemente.

El uso intensivo de recursos memoria y procesador de los equipos se observo al realizar pruebas de manera individual, ya que al realizarlas de manera distribuidas el uso de dichos recursos fue menor.

La característica síncrona del proceso limita en gran manera el proceso, al impedir que se aproveche la posible existencia de equipos o procesos que trabajen de manera mucho mejor que el resto de los participantes del proceso.

La utilización de redes de comunicación claramente afectan al proceso, ya que existen variables externas como seguridad y confiabilidad de la red, que provoca retardos o conflictos con el sistema.

Una característica que se encontró presente en los desarrollos fue la influencia de los protocolos de comunicación, donde la opción de TCP afectó el funcionamiento del sistema y su diseño, ya que al momento de mandar información por el socket, existió una limitante en cuanto al tamaño de dicha información, que provocó la saturación de la comunicación cuando se realizaron pruebas en un solo equipo, lo que provocó que se tuvo que desarrollar y establecer manualmente en el código límites de flujos de datos, que consistió en modificar el envió de la mejor combinación de columnas entre colonias, ya que su largo era mas grande que la capacidad del buffer, por tal razón fue necesario desarrollar un proceso de empaquetado de datos y llevar a cabo un proceso de confirmación de llegada de los mismos. Donde a cada mensaje se le agregaron códigos que indicaran si eran inicio, contenido o fin de mensaje.

Un aspecto que resulta interesante mencionar al momento de evaluar los resultados, son los resultados obtenidos por otros autores, donde si bien no se pueden comparar estrictamente los resultados por la naturaleza distinta de los problemas. Es necesario mencionar algunas tendencias, como por ejemplo los resultados obtenidos Max Manfrin

[31] donde si bien el paralelismo se utiliza para resolver otro tipo de problema, resultan llamativas algunas características que se refieren a que la arquitectura que mostró mejores resultados fue la implementación paralela independiente; es decir, en dicho estudio el funcionamiento independiente sobrepasaba el castigo o desmedro importante en el proceso debido a los procesos intensivos de comunicación, esto se refiere a que si bien uno podría encontrar algunas mejoras de manera temprana en los procesos en paralelos, esto se ve disminuido por el castigo que conllevan los procesos continuos de comunicación.

Al terminar de evaluar dichas aplicaciones y analizar las pruebas, se demostraron grandes desafíos y disyuntivas a resolver, claramente ligadas a obtener mejoras en la calidad de los resultados, que su performance tenga un nivel de rapidez aceptable, además de poder mejorar el uso de las herramientas de programación. Donde este último tema fue un desafío que debió considerarse para poder pensar recién en resolver los otros problemas.

En consecuencia, el paso siguiente consiste en hacer un cambio drástico al desarrollo, las opciones son variadas, pero casi todas ligadas al cambio de la plataforma de comunicación para mejorar las características del actual. Por tal motivo se comenzará con el cambio de lenguaje de programación y luego agregarle las mejoras principalmente en la comunicación.

Capítulo 7

Mejoras al Proceso.

Tomando en cuenta las evaluaciones realizadas a las pruebas sobre las implementaciones desarrolladas y detalladas anteriormente, se debió realizar una corrección al proceso de desarrollo, esto se debe principalmente a la necesidad de explorar nuevas formas de realizar el proceso y la búsqueda de herramientas que posibiliten un mejor desempeño.

Por todo lo anterior se decidió realizar los siguientes cambios:

Cambio en la plataforma de desarrollo (lenguaje de programación).
Posibilidad de probar con otras heurísticas.
Mejorar el proceso de comunicación, realizando comunicaciones asíncronas.
Mejorar el proceso de estudio de las pruebas, mediante la posibilidad de configuración de cada colonia individualmente.

7.1 Cambios en Herramientas de desarrollo

El motivo del cambio de plataforma se debe a las restricciones y algunas características de bajas prestaciones que se encontraron en el lenguaje de programación utilizado hasta el momento, en especial en lo referente a la administración de la comunicación especialmente de socket y lo correspondiente al envió y recepción de datos.

Las alternativas evaluadas para el cambio fueron Java y Mozart-Oz, donde la elección de este último se debió a una serie de características que permiten solucionar dichos problemas y además proveen nuevas prestaciones aún más adecuadas a la naturaleza del proyecto.

Mozart en realidad es la implementación del lenguaje Oz, pero su nombre se utiliza de manera conjunta ya que no existen otros desarrollos del lenguaje, por ello es que el presente informe los mencionaré de manera conjunta, Mozart-Oz es un lenguaje multiparadigma que arranca de un kernel procedural (que puede usarse como funcional) con variables lógicas que

sólo se asignan una vez, esto cambia bastante la manera de desarrollar los algoritmos, pero tiene sus ventajas, sobre todo para trabajar con distintos threads, donde las variables funcionan para sincronizar los procesos. Provocando que ya no sea necesario el trabajo y codificación de semáforos y/o bloqueos.

Las características de manejo de concurrencias y traspaso de mensajes (Message passing), programación de flujos de datos asíncrona y la capacidad de poder desarrollar aplicaciones en múltiples paradigmas convirtió a Mozart-Oz en la opción ideal para desarrollar las mejoras a los sistemas antes desarrollados.

El concepto de concurrencia del lenguaje permite el cálculo o proceso independiente, lo que en lenguajes orientados a objeto (Java) claramente trae problemas de codificación y de conceptos. Donde Oz muestra interesantes funcionalidades relacionadas directamente con la naturaleza de los algoritmos desarrollados en los sistemas.

El concepto de sistemas multiagentes ya se encuentra presente en las características de OZ, donde se consideran como agentes a aquellas entidades independientes que se comunican a través de traspaso de mensajes y que trabajan juntos con el objetivo de lograr una meta común. Donde solo es necesario agregar el concepto de canal de comunicación para poder obtener un sistema claramente distribuido y colaborativo. Dejando de lado las limitantes de los modelos concurrentes en la administración de los datos pertenecientes a una comunicación durante la ejecución de la aplicación. El puerto es un elemento ya perteneciente al lenguaje (NewPort) y que corresponde a un tipo de dato stream para poder rescatar los datos.

Al momento de poder combinar las prestaciones del lenguaje y las características de algunas de las arquitecturas de paralelización, es posible determinar que existe un elemento llamado bloque que permite en el caso de la colonia central poder administrar las comunicaciones y los procesos de las colonias buscadoras.

El funcionamiento de los bloques en la arquitectura de colonias administrador-buscador, se basa en poder combinar la programación funcional con el traspaso de mensajes concurrente, donde si bien en principio esta orientado a programas secuenciales, es posible configurar los flujos de datos para su manejo concurrente y asíncrono. En las figuras 7.1 y 7.2 es posible observar de manera simplificada algunas líneas el código que permiten la comunicación entre la colonia administradora y las colonias participantes en la búsqueda. Donde se utilizan

funcionalidades tickets para ofrecer entradas y la toma de los mismos para poder comunicarse con alguna aplicación. Los tickets quedan funcionando mientras este activa la aplicación que los cree, luego de ello el archivo sigue existiendo pero no es funcional.

El proceso de comunicación consiste principalmente en la creación de un ticket, que esta constituido por un archivo que puede ofrecerse de manera global a cualquier aplicación que acceda a ellos, específicamente un ticket tiene la siguiente forma interna:

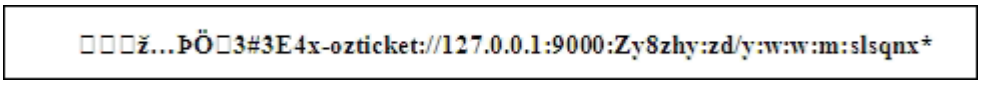


Ilustración 7.1: Ticket en Mozart-Oz

En esta línea se pueden leer partes generadas por el lenguaje y partes más legibles que corresponden a la dirección IP del equipo y el puerto que se deja abierto.

Es necesario señalar que la nueva aplicación tiene otra gran diferencia a las realizadas anteriormente, esto se refiere a la naturaleza asíncrona del proceso, cuyo objetivo principal es eliminar los problemas de retardo que se produjeron con el uso de una estrategia de comunicación síncrona, lo cual tiene como objetivo mejorar el concepto de rapidez en la obtención de resultados.

La elección de presentar mejoras en cuanto a la estrategia de comunicación se fundamenta en solucionar el problema de saturación de buffers y los excesivos retardos de comunicación principalmente. La comunicación asíncrona se desarrollará de manera de utilizar thread en la colonia central para administrar cada colonia, donde la recepción de datos desde cada una de ellas se realiza de manera independiente, permitiendo que varias soluciones lleguen de manera concurrente y sin necesidad de realizar procesos de espera.

La combinación de comunicación asíncrona con la sincronización de flujos de datos, permite que los mensajes entre las colonias no dependan unas de otras en su funcionamiento y solo la aplicación administradora se encarga de verificar y actualizar los valores generales de la solución, obteniendo mejores prestaciones en relación a las esperas y principalmente en lo referente a la calidad de las soluciones y tiempo de desarrollo de las mismas.

```
        Import                          % Librerías que proveen los elementos de
comunicación.
          Connection
          Pickle
          ....
        proc{Offer X FN}          % Procedimiento para ofrecer el ticket.
           {Pickle.save{Connection.offerUnlimited X}FN}
        end

          fun{Take FNURL}         % Procedimiento para tomar el ticket.
           {Connection.take {Pickle.load FNURL}}
          end
        ....
        {NewPort S Ticket}                % Se crea el puerto.
        {Offer Ticket "C:/files/ticket_m"}  %Se graba el ticket en un archivo

        Comunicacion={Take "//"#@IPServer#"/files/ticket_m"}   %Se lee el
archivo del ticket y se logra la conexión.

        {Send Comunicacion  «hola »}        %Con solo indica la variable del ticket
se le envía la información directamente.
```

Ilustración 7.2: Instrucciones de Comunicación en Mozart-Oz.

Cada aplicación ofrece ticket y toma tickets para comunicarse, las colonias lo primero que deben enviar es un mensaje con su IP para que la colonia administradora la pueda localizar y tomar su ticket.

7.2 Implementación

Para la realización de los cambios, se consideraron las siguientes características:

- *Elección de un problema de optimización:* se continuó con el problema de Cobertura de Conjuntos. (SCP)

- *Elección de un algoritmo ACO:* tal como el desarrollo anterior y basado en el análisis de buenas características del primer desarrollo, se continuó con la utilización del algoritmo Ant Colony System (ACS).

- *La estructura de paralelización:* es configurable por el controlador de la colonia administradora.

- *Estrategia de comunicación:* la forma en que se desarrollarán las comunicaciones será de manera asíncrona, es decir cada colonia tendrá libertad en cuanto a su ritmo de desarrollo de la solución.

- *Configuración*: existe la posibilidad de configurar cada parámetro de entrada a cada colonia, es decir Ro, Q0, Cantidad de colonias que participaras, cantidad de hormigas en la colonia, cantidad de iteraciones, problema benchmark, intervalo de migración y heurística

- *Heurística*: existe la posibilidad de configurar que tipo de heurística ocupar, teniendo 2 opciones:
  - *1 / Costo de la Columna:* para la elección de la siguiente columna a agregar a la solución además de la carga de feromona, se considerará aquellas columnas que presenten el menor costo.
  - *Cantidad de filas que se agregan a la solución / Costo de la Columna:* para la elección de la siguiente columna a agregar, además de la carga de feromona, se considerará cuantas nuevas filas son agregadas a la solución en conjunto con el costo de la columna que las contenga.

Las nuevas elecciones antes mencionadas agregaron las características necesarias para poder transformar una serie de aplicaciones en una “plataforma de búsquedas”.

La elección sobre la estructura a desarrollar también debió ser llevada a cabo, ya que si bien se podría desarrollar cada una de ellas, es necesario evaluar cual representa de mejor manera los cambios y su efecto en el proceso y resultados; es por ello que la elección de una estructura administradora central – colonias buscadoras tiene la correspondencia necesaria para evaluar el mejoramiento de realizar una comunicación asíncrona.

El funcionamiento en si consiste en una colonia administradora que recibirá las mejores soluciones generadas por cada colonia y las colonias serán tan independientes que hasta tienen la facultad de determinar cuando comunicarse, al tener la capacidad de evaluar sus resultados en comparación a los globales. Cada colonia además pude tener configuraciones distintas, las cuales se fijan en la colonia Administradora. A continuación se muestran imagenes sobre la versión mejorada:

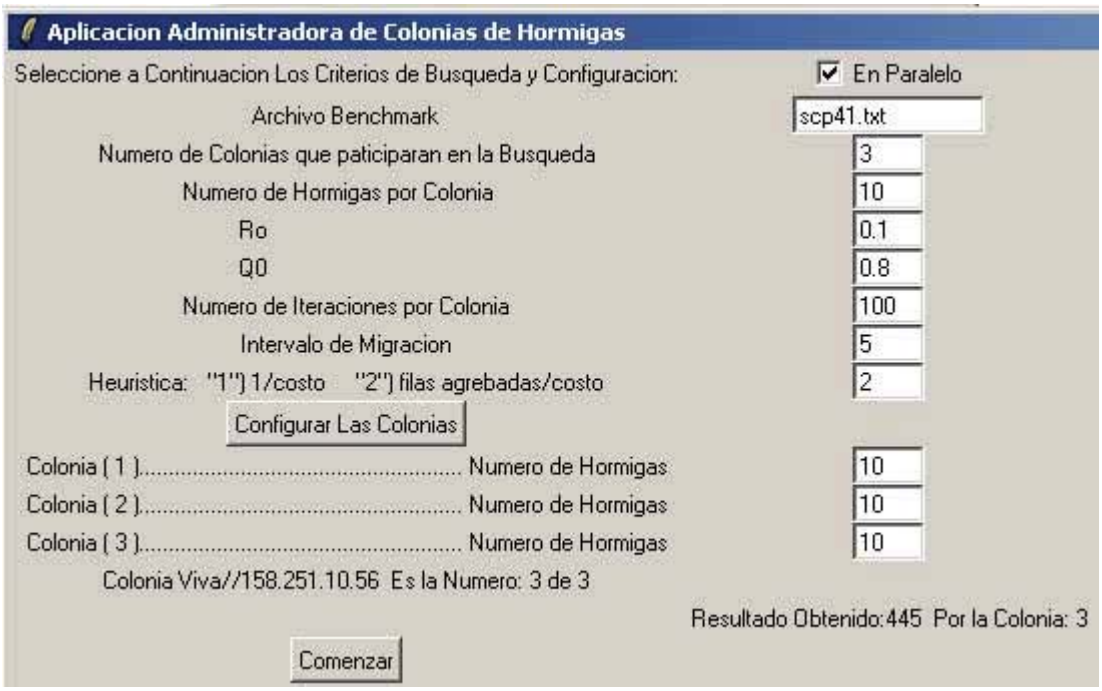


Ilustración 7.3: Parte Izquierda Pantalla Funcionamiento de Colonia Administradora.

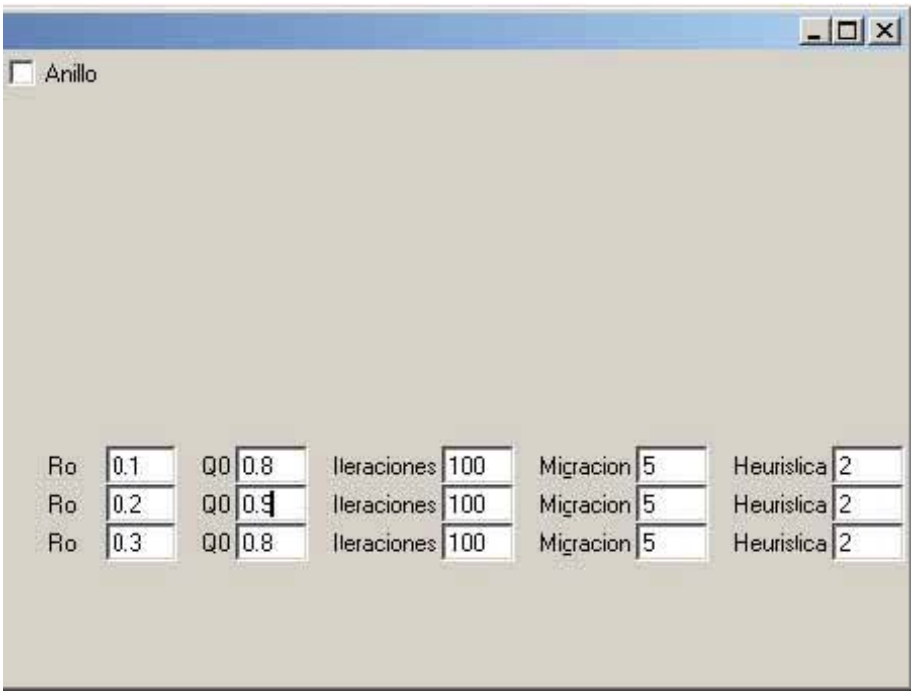


Ilustración 7.4: Parte Derecha Pantalla Funcionamiento de Colonia Administradora.

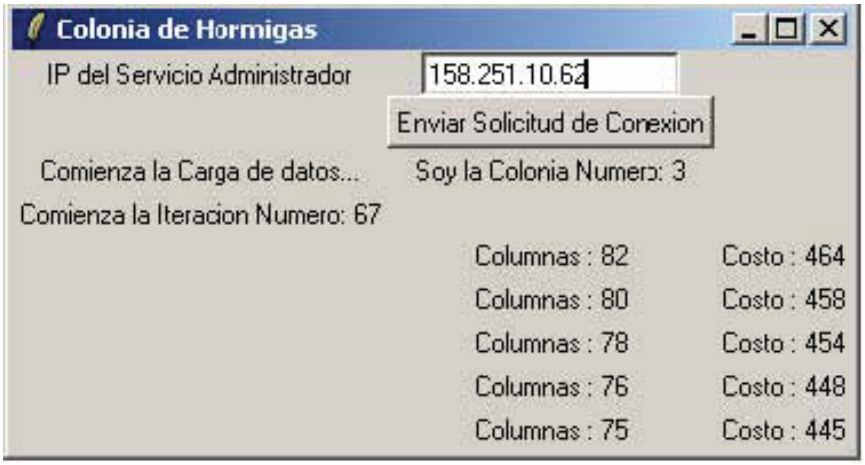


Ilustración 7.5: Pantalla Funcionamiento Colonia Buscadora.

7.3 Requerimientos

El funcionamiento de las aplicaciones, mantienen la característica de llevarse acabo de manera paralela, lo cual debe desarrollarse en maquinas distribuidas, ya que la creación de ticket restringe los nombres de los archivos que los contienen.

El funcionamiento de las aplicaciones, requieren de ciertas capacidades de hardware especialmente de Memoria Ram, con un mínimo de 128MB para asegurar un buen nivel de prestaciones, obvio con la salvedad que los equipos no estén desarrollando variadas aplicaciones y con un alto consumo de memoria.

En cuanto al uso del procesador, este elemento sigue en estrecha relación con las características del lenguaje de programación ya que se crean aplicaciones que no exigen en demasía la capacidad de los procesadores actuales. El tipo de procesador que se utiliza en las pruebas debe presentar una capacidad de procesamiento sobre 1.4Ghz, aunque con menores capacidades se obtienen desarrollos igualmente aceptables.

Disponer de una conexión de red que tenga la capacidad de poder aceptar procesos que administren o accedan a equipos ubicados dentro de la red y que sean identificados mediante direcciones IP. Cuyos quipos deben tener la capacidad de poder liberar el uso de puertos de comunicación.

Consideraciones de seguridad, se requiere además configurar aspectos de seguridad de equipos y redes, en cuanto a estas últimas es permitir las conexiones dinámicas entre equipos, ya que cada aplicación puede crear procesos de comunicación cuando se estime conveniente. Sobre los equipos es necesario determinar algunos permisos de funcionamiento de las aplicaciones por parte de antivirus y barreras de seguridad del sistema operativo, especialmente con sistemas operativos Windows (Firewall de Windows) esto se debe a las características de funcionamiento de las aplicaciones mediante comunicaciones intensas y que el firewall lo reconoce como posibles ataques o virus que están tratando de acceder a datos del equipo.

Se requiere que exista una carpeta compartida con el nombre de files, es necesario dicha carpeta para guardar los tickets, además de poder incluir la aplicación, archivos benchmark y donde se registren los resultados de cada experimento.en la ilustración siguiente se muestra como:

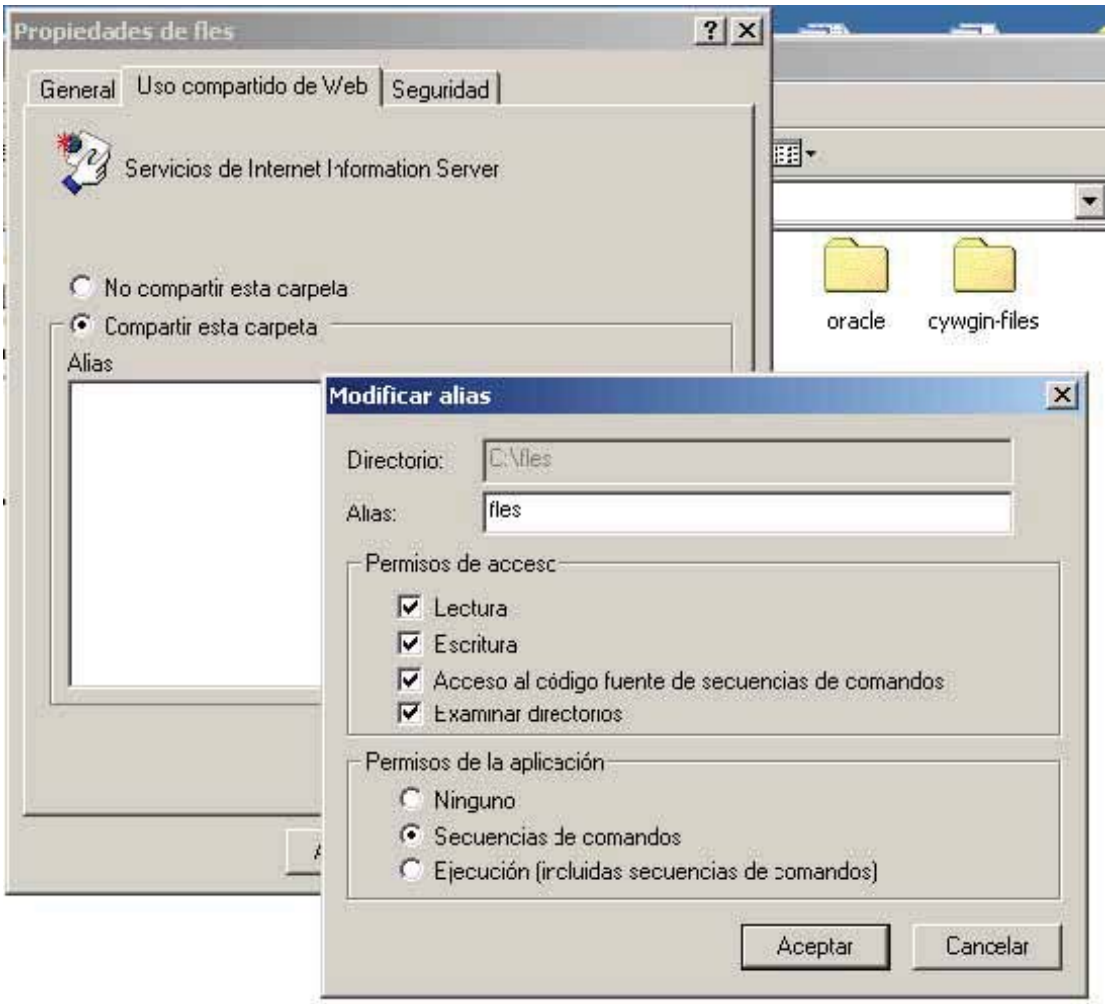


Ilustración 7.6: Pantallas compartir Carpeta “files”.

Cada uno de los equipos en los cuales se ejecutarán tanto en las colonias de características administradoras como buscadoras, deberán tener instalado y ejecutándose en el sistema, el motor de OZ; si bien este motor no es necesario para la totalidad de las aplicaciones desarrolladas en dicha plataforma, algunas de las características presentes en los softwares implementados en el presente desarrollo requieren de funcionalidades especiales, específicamente el de lectura y accesos remotos. Dicho motor puede ser adquirido gratuitamente desde la página: http://www.mozart-oz.org/. El ambiente de programación requiere de un editor llamado emacs, En la ilustración siguiente se aprecian las ventanas principales de instalación de la versión para Windows.

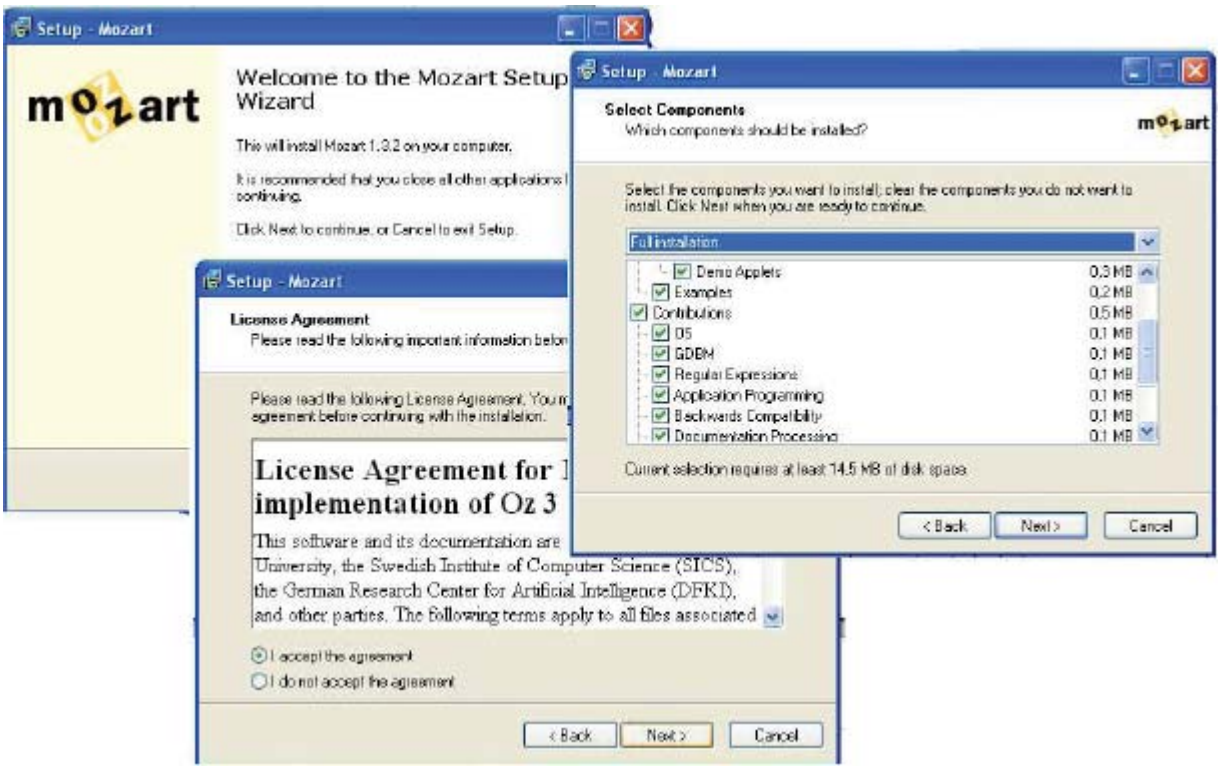


Ilustración 7.7: Pantallas Instalación Mozart-oz.

Los prototipos desarrollados se basaron en distintas versiones de software con el objetivo de seguir con un proceso iterativo incremental.

7.4 Pruebas

El prototipo fue desarrollado con las siguientes características: Resolución de un problema de optimización de Cobertura de Conjuntos. (SCP) mediante el algoritmo Ant Colony System (ACS), con una estructura de distribución configurable, forma de comunicación asíncrona, heurística configurable, principalmente se puede configurar cada parámetro de cada colonia.

Las pruebas fueron realizadas utilizando problemas de prueba extraídos de ORLIB de Beasley [34].

Al terminar este nuevo desarrollo quedaran claras ciertas diferencias:

El algoritmo puede ser configurado considerablemente lo que logra poder determinar las mejores configuraciones y los mejores parámetros de búsquedas. Permitiendo

verificar los valores de tendencia rápida tendiera rápidamente a valores de tendencia temprana, lo cual provocó al largo del proceso de varias iteraciones llegar a un mejor valor de resolución.

La paralelización logra que al no tender rápidamente hacia un valor, verifica el aporte de la paralelización al mejorar cada una de los corrimientos individuales.

El resultado mostró mejor funcionamiento que el corrimiento individual del algoritmo.

Eliminación mayoritaria del retardo por coordinación, el cual consistía en la espera que debían realizar algunas colonias que desarrollaron su búsqueda mucho más rápido, debiendo quedarse esperando que terminaran las demás colonias para poder continuar su proceso, además no es necesario que todas las colonias terminen al mismo tiempo.

Un mayor uso de memoria y procesador, por el uso de una interfaz gráfica con mayores requerimientos.

Continuidad en la utilización intensiva de redes y puertos de comunicación.

En general este desarrollo mostró resultados mucho mejores a los desarrollados con anterioridad y mostrando además la utilidad del proceso en paralelo. En lo referente a la calidad de la solución en el siguiente punto se muestran los resultados obtenidos.

7.5 Análisis de las Pruebas

A continuación se muestran en detalle algunas pruebas realizadas sobre esta plataforma de búsquedas. Dichos resultados están expuestos de manera de mostrar las diferentes configuraciones del proceso.

Cabe mencionar que solo se nombran algunas de las pruebas representativas, ya que la posibilidad de configuraciones es inmensa y abarcarían gran cantidad de hojas del informe.

Tabla 7.1: Resumen Resultados Versión Final Administrador-Buscador.

ADMINISTRADOR – BUSCADOR					
Problema	**Filas**	**Columnas**	**Optimo**	**ACS**	**ACS Paralelo**

Scp41	200	1000	429	463	440
Scp42	200	1000	512	590	542
Scp61	200	1000	138	154	155
Scp62	200	1000	146	163	157

7.6 Detalle de las Pruebas

Tabla 7.2: Resultados Prueba (SCP41; Ro: 0,1 0,2 0,3; Q0: 0,4 0,5 0,6; Paralelo Colaborativo)

Problema	**SCP41**
Filas	200
Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**
Ro	**0,1**	**Ro**	**0,2**	**Ro**	**0,3**
Qo	**0,4**	**Qo**	**0,5**	**Qo**	**0,6**
Migración	**1**	**Migración**	**1**	**Migración**	**1**
Heurística	**2**	**Heurística**	**2**	**Heurística**	**2**

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Solución	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**
1	495	7	495	478	6	478	472	6	472
2	479	6	479	509	7	478	473	7	472
3	491	7	479	471	7	471	472	6	472
4	465	7	465	478	7	471	469	6	469
5	470	7	465	478	6	471	470	6	469
6	468	7	465	466	7	466	467	7	467
7	457	7	457	463	7	463	464	6	464
8	465	7	457	465	6	463	467	6	464
9	465	7	457	474	7	463	451	6	451
10	460	7	457	461	7	461	456	7	451
11	461	7	457	447	6	447	456	6	451
12	457	7	457	466	7	447	451	7	451
13	451	7	451	454	6	447	451	6	451
14	461	7	451	447	6	447	451	6	451
15	457	7	451	447	7	447	451	6	451

16	451	7	451	447	6	447	451	6	451
17	450	7	450	447	7	447	451	7	451
18	450	6	450	447	6	447	451	6	451
19	449	7	449	447	7	447	451	6	451
20	453	7	449	447	6	447	451	6	451
21	453	7	449	446	7	446	451	6	451
22	447	7	447	447	6	446	448	7	448
23	446	6	446	453	7	446	447	7	447
24	447	7	446	446	6	446	448	6	447
25	447	7	446	446	7	446	450	6	447
26	446	6	446	446	6	446	447	6	447
27	446	7	446	446	6	446	447	6	447
28	446	7	446	446	7	446	447	6	447
29	452	7	446	446	6	446	447	6	447
30	446	7	446	446	7	446	447	6	447
31	446	7	446	446	6	446	446	6	446
32	446	7	446	446	7	446	447	6	446
33	446	7	446	446	6	446	446	7	446
34	449	6	446	446	7	446	446	6	446
35	446	7	446	446	7	446	446	6	446

36	446	7	446	446	6	446	446	7	446
37	447	7	446	446	7	446	447	6	446
38	446	7	446	446	6	446	446	6	446
39	446	7	446	446	7	446	446	7	446
40	446	7	446	446	7	446	446	6	446
41	447	7	446	446	6	446	446	7	446
42	446	7	446	446	7	446	446	6	446
43	446	6	446	446	7	446	446	6	446
44	446	7	446	446	6	446	446	6	446
45	446	7	446	446	7	446	446	7	446
46	446	6	446	446	6	446	446	6	446
47	446	7	446	446	6	446	447	6	446
48	446	7	446	446	7	446	446	7	446
49	446	7	446	446	6	446	446	6	446
50	446	7	446	446	7	446	446	6	446
51	446	6	446	449	6	446	446	6	446
52	446	7	446	446	7	446	446	7	446
53	446	7	446	447	7	446	446	6	446
54	443	7	443	446	6	446	446	6	446
55	449	6	443	446	7	446	446	6	446

56	444	7	443	446	6	446	446	7	446
57	443	7	443	446	6	446	446	6	446
58	443	7	443	446	7	446	446	6	446
59	446	7	443	446	6	446	446	7	446
60	444	6	443	446	7	446	443	6	443
61	447	7	443	446	6	446	446	6	443
62	443	7	443	443	7	443	443	6	443
63	443	7	443	443	6	443	443	6	443
64	443	7	443	446	7	443	447	6	443
65	443	6	443	446	6	443	443	6	443
66	443	7	443	441	6	441	443	6	443
67	446	7	443	443	7	441	443	6	443
68	443	7	443	443	7	441	443	6	443
69	443	7	443	443	6	441	443	7	443
70	443	6	443	441	7	441	446	6	443
71	443	7	443	441	6	441	443	6	443
72	443	7	443	441	7	441	443	6	443
73	443	7	443	441	6	441	443	7	443
74	443	7	443	441	7	441	443	6	443
75	443	7	443	442	6	441	443	6	443

76	441	6	441	441	7	441	443	6	443
77	443	7	441	443	6	441	443	6	443
78	443	7	441	441	7	441	441	6	441
79	441	7	441	441	6	441	441	7	441
80	441	6	441	440	7	440	441	6	441
81	441	7	441	441	6	440	441	6	441
82	441	7	441	441	7	440	441	7	441
83	441	6	441	440	6	440	441	6	441
84	441	7	441	442	7	440	441	6	441
85	441	7	441	441	6	440	441	6	441
86	441	6	441	440	7	440	441	6	441
87	441	7	441	440	6	440	441	7	441
88	442	7	441	440	6	440	441	6	441
89	441	6	441	440	7	440	443	6	441
90	442	7	441	442	6	440	441	6	441
91	440	7	440	441	7	440	441	6	441
92	440	7	440	440	6	440	440	6	440
93	440	7	440	440	6	440	440	7	440
94	440	6	440	441	7	440	440	6	440
95	440	7	440	440	6	440	440	6	440

96	440	6	440	440	7	440	440	6	440
97	440	7	440	440	6	440	440	6	440
98	440	6	440	440	7	440	440	7	440
99	440	6	440	440	6	440	440	6	440
100	440	7	440	440	7	440	442	6	440
RESULTADO	440								

Tabla 7.3: Resultados Prueba (SCP41; Ro: 0,1 0,2 0,3; Q0: 0,8 0,9 0,8; Paralelos Independientes)

Problema	**SCP41**
Filas	200
Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**
Ro	**0,1**	**Ro**	**0,2**	**Ro**	**0,3**
Qo	**0,8**	**Qo**	**0,9**	**Qo**	**0,8**
Migración	**5**	**Migración**	**5**	**Migración**	**5**

Heurística	2	Heurística	2	Heurística	2

	Colonias INDEPENDIENTES								
ITERACIONES	Col 1			Col 2			Col 3		
	Solución	Tiempo	Acumulado	Solución	Tiempo	Acumulado	Solución	Tiempo	Acumulad
1	464	6	464	464	6	464	464	6	464
2	493	6	464	463	6	463	493	6	464
3	464	6	464	461	5	461	458	7	458
4	457	6	457	458	6	458	462	6	458
5	464	6	457	461	6	458	461	5	458
6	453	6	453	458	5	458	458	6	458
7	462	6	453	458	6	458	458	6	458
8	457	6	453	458	6	458	458	6	458
9	454	6	453	458	5	458	454	6	454
10	453	6	453	458	6	458	455	6	454
11	453	6	453	458	6	458	454	6	454
12	453	6	453	458	5	458	454	6	454
13	453	6	453	458	6	458	454	6	454

14	453	6	453	455	6	455	454	6	454	455	6
15	453	6	453	458	5	455	454	6	454	455	6
16	453	6	453	455	6	455	454	6	454	455	6
17	453	6	453	455	6	455	454	6	454	455	6
18	453	6	453	455	6	455	454	6	454	455	6
19	453	6	453	455	5	455	454	5	454	455	6
20	453	6	453	455	6	455	454	6	454	455	6
21	453	6	453	455	5	455	454	6	454	455	6
22	453	6	453	455	6	455	454	6	454	455	6
23	452	6	452	455	5	455	448	6	448	455	6
24	453	6	452	455	5	455	454	6	448	455	6
25	452	6	452	455	6	455	448	6	448	455	6
26	452	6	452	455	6	455	448	6	448	455	6
27	452	6	452	455	5	455	448	5	448	455	6
28	449	6	449	455	5	455	448	6	448	455	6
29	452	6	449	455	6	455	448	6	448	455	6
30	452	6	449	455	5	455	448	6	448	455	6
31	449	6	449	455	6	455	448	6	448	455	6
32	449	6	449	455	6	455	448	6	448	455	6
33	449	6	449	455	5	455	448	6	448	455	6

34	449	6	449	455	6	455	448	5	448	455	6
35	449	6	449	455	6	455	448	6	448	455	6
36	449	6	449	455	5	455	448	6	448	455	6
37	449	6	449	455	6	455	448	6	448	455	6
38	449	6	449	455	6	455	448	5	448	455	6
39	449	5	449	455	5	455	445	6	445	455	6
40	449	6	449	455	6	455	448	6	445	455	6
41	449	6	449	455	6	455	445	6	445	455	6
42	449	6	449	455	6	455	448	6	445	455	6
43	449	6	449	455	6	455	447	5	445	455	6
44	449	6	449	455	5	455	445	5	445	455	6
45	449	6	449	455	6	455	445	6	445	455	6
46	449	6	449	455	6	455	445	6	445	455	6
47	449	6	449	455	5	455	445	5	445	455	6
48	449	6	449	455	6	455	445	6	445	455	6
49	449	6	449	455	6	455	445	6	445	455	6
50	449	6	449	455	5	455	445	6	445	455	6
51	449	6	449	455	6	455	445	6	445	455	6
52	449	6	449	455	5	455	445	6	445	455	6
53	449	7	449	455	6	455	445	6	445	455	6

54	449	6	449	455	6	455	445	6	445		
55	449	6	449	455	6	455	445	5	445		
56	449	6	449	455	5	455	445	6	445		
57	449	6	449	455	6	455	445	6	445		
58	449	5	449	455	6	455	445	6	445		
59	449	6	449	455	6	455	445	5	445		
60	449	6	449	455	5	455	445	6	445		
61	449	6	449	455	6	455	445	6	445		
62	449	6	449	455	5	455	445	6	445		
63	449	7	449	455	6	455	445	6	445		
64	449	6	449	455	6	455	445	5	445		
65	449	6	449	455	5	455	445	7	445		
66	449	6	449	455	6	455	445	6	445		
67	449	6	449	455	6	455	445	6	445		
68	449	6	449	455	6	455	445	6	445		
69	449	6	449	455	5	455	445	6	445		
70	449	5	449	455	6	455	445	7	445		
71	449	6	449	455	6	455	445	6	445		
72	449	6	449	455	6	455	445	6	445		
73	449	6	449	455	5	455	445	6	445		

74	449	6	449	455	6	455	445	6	445	
75	449	6	449	455	6	455	445	6	445	
76	449	6	449	455	5	455	446	6	445	
77	449	6	449	455	6	455	445	7	445	
78	452	6	449	455	6	455	445	6	445	
79	449	6	449	455	5	455	445	6	445	
80	449	6	449	455	6	455	448	6	445	
81	449	6	449	455	6	455	445	6	445	
82	449	6	449	455	6	455	446	6	445	
83	449	7	449	455	5	455	445	6	445	
84	449	6	449	455	6	455	445	7	445	
85	449	5	449	455	6	455	445	5	445	
86	449	6	449	457	5	455	445	5	445	
87	449	6	449	455	6	455	445	6	445	
88	449	6	449	450	6	450	445	6	445	
89	449	6	449	455	5	450	445	6	445	
90	449	6	449	450	5	450	445	6	445	
91	449	6	449	450	6	450	445	6	445	
92	449	6	449	450	6	450	445	6	445	

93	449	5	449	450	5	450	445	6	445	
94	449	7	449	450	6	450	445	5	445	
95	449	6	449	450	5	450	445	6	445	
96	449	6	449	450	6	450	445	6	445	
97	449	6	449	450	5	450	447	6	445	
98	449	5	449	450	6	450	445	5	445	
99	449	6	449	450	6	450	445	6	445	
100	449	7	449	450	5	450	445	6	445	
RESULTADO	449 – 449 -445									

Tabla 7.4: Resultados Prueba (SCP41; Ro: 0,1 0,2 0,3; Q0: 0,8 0,9 0,8; Paralelos Colaborativos)

Problema	**SCP41**
Filas	200
Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**

Ro	**0,1**	**Ro**	**0,2**	**Ro**	**0,3**
Qo	**0,8**	**Qo**	**0,9**	**Qo**	**0,8**
Migración	**5**	**Migración**	**5**	**Migración**	**5**
Heurística	**2**	**Heurística**	**2**	**Heurística**	**2**

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Solución	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**
1	464	6	464	464	6	464	464	5	464
2	493	6	464	463	6	463	493	6	464
3	464	6	464	461	6	461	458	6	458
4	457	6	457	458	6	458	462	6	458
5	464	6	457	461	6	458	461	6	458
6	453	6	453	458	6	458	449	5	449
7	462	6	453	458	5	458	459	6	449
8	457	6	453	458	6	458	449	6	449
9	454	6	453	458	6	458	449	6	449
10	453	6	453	458	5	458	449	6	449
11	453	6	453	461	6	458	449	6	449

12	453	6	453	457	6	457	449	6	449
13	453	6	453	457	6	457	449	6	449
14	453	5	453	454	6	454	449	6	449
15	453	7	453	457	6	454	449	6	449
16	452	6	452	457	5	454	449	6	449
17	452	6	452	453	6	453	449	6	449
18	452	5	452	453	6	453	449	6	449
19	452	6	452	453	6	453	449	6	449
20	452	6	452	450	5	450	449	6	449
21	452	6	452	452	5	450	449	6	449
22	452	6	452	450	6	450	449	6	449
23	449	6	449	450	6	450	448	6	448
24	449	6	449	450	5	450	449	6	448
25	449	6	449	450	6	450	445	6	445
26	449	6	449	452	6	450	448	6	445
27	449	6	449	449	6	449	445	6	445
28	449	5	449	449	6	449	445	6	445
29	449	6	449	449	7	449	445	5	445
30	449	6	449	449	6	449	445	5	445
31	449	6	449	449	7	449	445	6	445

32	445	6	445	448	7	448	445	6	445
33	449	6	445	448	6	448	445	6	445
34	445	5	445	448	6	448	445	6	445
35	448	6	445	448	7	448	445	6	445
36	445	6	445	448	6	448	445	5	445
37	445	6	445	445	5	445	445	6	445
38	445	6	445	445	7	445	445	6	445
39	445	6	445	445	7	445	445	6	445
40	445	5	445	445	7	445	445	6	445
41	445	6	445	445	7	445	445	6	445
42	445	6	445	445	6	445	445	6	445
43	445	6	445	446	6	445	445	5	445
44	445	5	445	445	7	445	445	6	445
45	445	5	445	445	6	445	445	6	445
46	445	6	445	445	6	445	445	6	445
47	445	6	445	445	7	445	445	5	445
48	445	6	445	445	6	445	445	6	445
49	448	6	445	445	5	445	445	6	445
50	445	5	445	445	6	445	445	6	445
51	445	6	445	445	6	445	445	6	445

52	445	6	445	445	6	445	445	6	445
53	445	6	445	445	5	445	445	6	445
54	445	6	445	445	7	445	445	6	445
55	445	6	445	445	7	445	445	6	445
56	445	6	445	445	6	445	445	5	445
57	445	6	445	445	7	445	445	6	445
58	445	5	445	445	6	445	445	6	445
59	445	6	445	445	8	445	445	6	445
60	445	6	445	445	6	445	445	6	445
61	445	6	445	445	7	445	445	5	445
62	445	6	445	445	6	445	445	6	445
63	445	5	445	445	6	445	445	5	445
64	445	6	445	445	6	445	445	5	445
65	448	6	445	445	6	445	445	6	445
66	445	6	445	445	6	445	446	6	445
67	445	6	445	446	7	445	445	6	445
68	445	6	445	445	6	445	445	6	445
69	445	6	445	445	5	445	445	5	445
70	445	6	445	445	6	445	445	6	445
71	445	5	445	445	6	445	445	6	445

72	445	6	445	445	6	445	445	6	445
73	445	6	445	445	5	445	445	6	445
74	445	5	445	445	6	445	445	5	445
75	446	5	445	445	6	445	445	6	445
76	445	5	445	445	5	445	445	6	445
77	445	6	445	445	6	445	445	6	445
78	445	6	445	445	6	445	445	6	445
79	448	6	445	445	5	445	445	6	445
80	445	6	445	445	6	445	446	6	445
81	446	6	445	445	6	445	448	5	445
82	445	5	445	445	5	445	445	6	445
83	445	6	445	445	6	445	445	6	445
84	445	6	445	445	6	445	445	6	445
85	445	6	445	445	6	445	445	6	445
86	445	6	445	445	5	445	445	6	445
87	445	6	445	445	6	445	445	6	445
88	445	6	445	445	6	445	445	5	445
89	445	6	445	445	5	445	445	6	445
90	445	5	445	445	6	445	445	6	445
91	445	6	445	445	6	445	445	6	445

92	445	6	445	445	5	445	445	6	445
93	445	5	445	445	6	445	445	6	445
94	445	6	445	445	6	445	445	5	445
95	445	6	445	445	5	445	445	6	445
96	446	6	445	445	6	445	448	6	445
97	445	6	445	445	5	445	445	6	445
98	445	5	445	445	6	445	445	5	445
99	445	6	445	445	6	445	445	5	445
100	445	6	445	445	5	445	445	6	445
RESULTADO	445								

Tabla 7.5: Resultados Prueba (SCP41; Ro: 0,4 0,5 0,4; Q0: 0,4 0,5 0,5; Paralelos Colaborativos)

Problema	**SCP41**
Filas	200
Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**
Ro	**0,4**	**Ro**	**0.5**	**Ro**	**0.4**
Qo	**0.4**	**Qo**	**0.5**	**Qo**	**0.5**
Migración	**5**	**Migración**	**5**	**Migración**	**5**
Heurística	**2**	**Heurística**	**2**	**Heurística**	**2**

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Solución	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**
1	495	7	495	478	7	478	478	7	478
2	479	7	479	509	6	478	509	7	478
3	491	7	479	471	7	471	471	7	471
4	474	7	474	478	7	471	478	7	471
5	477	7	474	473	6	471	473	6	471
6	461	7	461	471	6	471	471	7	471
7	473	8	461	473	7	471	473	7	471
8	461	7	461	471	7	471	471	6	471
9	458	7	458	463	6	463	463	6	463

10	461	7	458	471	7	463	471	7	463
11	447	7	447	463	7	463	456	7	456
12	456	7	447	468	6	463	454	6	454
13	447	7	447	462	7	462	463	7	454
14	447	7	447	463	7	462	454	7	454
15	447	7	447	460	6	460	454	7	454
16	446	7	446	463	7	460	449	6	449
17	447	7	446	462	7	460	454	7	449
18	448	6	446	460	6	460	449	6	449
19	446	7	446	462	7	460	451	7	449
20	447	7	446	457	6	457	446	6	446
21	446	7	446	452	7	452	450	7	446
22	447	7	446	457	6	452	446	7	446
23	446	7	446	455	7	452	448	6	446
24	446	7	446	452	7	452	446	7	446
25	446	6	446	451	6	451	448	6	446
26	446	7	446	452	6	451	450	6	446
27	446	7	446	448	7	448	446	6	446
28	446	7	446	451	6	448	448	6	446
29	446	7	446	448	7	448	446	7	446

30	446	7	446	445	6	445	446	6	446
31	447	7	446	447	6	445	446	7	446
32	446	7	446	447	7	445	446	7	446
33	446	7	446	445	7	445	446	6	446
34	446	7	446	448	6	445	446	7	446
35	446	7	446	445	7	445	446	6	446
36	445	7	445	444	6	444	447	6	446
37	445	7	445	445	7	444	445	7	445
38	447	7	445	444	6	444	444	6	444
39	448	7	445	444	7	444	444	7	444
40	445	7	445	444	7	444	445	7	444
41	448	7	445	444	6	444	444	6	444
42	447	7	445	444	7	444	444	7	444
43	444	7	444	444	7	444	444	7	444
44	444	7	444	445	6	444	445	6	444
45	444	7	444	445	6	444	445	7	444
46	444	6	444	444	7	444	444	6	444
47	444	7	444	444	6	444	444	7	444
48	447	7	444	444	7	444	444	6	444
49	444	7	444	444	6	444	444	6	444

50	444	7	444	444	7	444	444	7	444
51	444	7	444	444	6	444	447	7	444
52	444	7	444	444	7	444	444	6	444
53	444	6	444	444	6	444	445	7	444
54	444	7	444	444	7	444	444	6	444
55	444	7	444	444	6	444	444	6	444
56	447	7	444	444	7	444	444	6	444
57	445	6	444	444	6	444	444	6	444
58	444	8	444	444	6	444	444	6	444
59	444	7	444	444	7	444	444	7	444
60	447	6	444	445	6	444	444	6	444
61	444	6	444	444	7	444	444	6	444
62	444	7	444	444	6	444	444	6	444
63	444	7	444	444	7	444	444	7	444
64	444	7	444	447	6	444	452	6	444
65	444	7	444	444	6	444	444	7	444
66	444	7	444	444	6	444	444	6	444
67	444	7	444	444	7	444	444	7	444
68	444	7	444	444	7	444	444	7	444
69	444	7	444	444	6	444	444	6	444

70	447	7	444	444	6	444	444	7	444
71	444	6	444	444	7	444	444	6	444
72	444	7	444	444	7	444	444	7	444
73	444	7	444	444	6	444	444	6	444
74	444	6	444	444	7	444	444	7	444
75	444	7	444	445	6	444	445	6	444
76	444	7	444	444	7	444	444	7	444
77	444	7	444	447	6	444	449	7	444
78	445	7	444	444	6	444	444	7	444
79	444	7	444	444	7	444	444	6	444
80	444	7	444	453	6	444	444	7	444
81	444	7	444	444	7	444	444	6	444
82	444	7	444	444	6	444	444	7	444
83	444	7	444	445	7	444	446	6	444
84	444	6	444	444	7	444	447	7	444
85	444	7	444	445	6	444	445	6	444
86	442	6	442	444	6	444	444	7	444
87	444	7	442	444	7	444	444	6	444
88	446	7	442	444	6	444	444	7	444
89	445	7	442	444	7	444	444	7	444

90	442	6	442	445	7	444	446	6	444
91	443	8	442	444	6	444	444	6	444
92	442	6	442	444	6	444	444	7	444
93	442	7	442	444	7	444	444	6	444
94	442	7	442	444	6	444	444	6	444
95	442	7	442	444	7	444	444	7	444
96	442	6	442	444	6	444	444	7	444
97	442	7	442	444	7	444	447	7	444
98	442	7	442	444	6	444	442	6	442
99	442	6	442	444	7	444	442	6	442
100	442	7	442	444	6	444	442	7	442
RESULTADO	442								

Tabla 7.6: Resultados Prueba (SCP42; Ro: 0,1 0,2 0,3; Q0: 0,4 0,5 0,6; Paralelos Colaborativos)

Problema	**SCP42**
Filas	200
Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**
Ro	**0,1**	**Ro**	**0,2**	**Ro**	**0,3**
Qo	**0,4**	**Qo**	**0,5**	**Qo**	**0,6**
Migración	**1**	**Migración**	**1**	**Migración**	**1**
Heurística	**2**	**Heurística**	**2**	**Heurística**	**2**

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Solución	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**
1	628	7	628	626	7	626	620	7	620
2	616	7	616	612	7	612	587	7	587
3	621	7	616	604	7	604	618	6	587
4	612	7	612	608	7	604	587	7	587
5	627	7	612	602	7	602	590	6	587
6	583	7	583	579	7	579	582	7	582
7	600	7	583	594	7	579	585	6	582
8	601	7	583	573	6	573	582	7	582

9	568	7	568	573	8	573	582	6	582
10	595	8	568	568	7	568	582	7	582
11	580	7	568	573	7	568	582	7	582
12	576	7	568	570	7	568	582	6	582
13	565	7	565	568	6	568	582	7	582
14	585	7	565	568	7	568	582	6	582
15	560	7	560	568	7	568	582	7	582
16	572	7	560	563	7	563	570	7	570
17	559	7	559	581	7	563	575	7	570
18	560	7	559	575	7	563	570	6	570
19	555	6	555	566	7	563	564	6	564
20	565	7	555	563	7	563	570	7	564
21	559	7	555	563	7	563	564	6	564
22	562	8	555	563	7	563	564	7	564
23	562	6	555	570	7	563	560	6	560
24	555	7	555	563	6	563	564	7	560
25	559	7	555	563	7	563	564	6	560
26	551	7	551	556	7	556	569	6	560
27	555	7	551	562	6	556	555	7	555
28	555	6	551	556	7	556	555	6	555

29	555	8	551	556	7	556	555	6	555
30	551	6	551	556	7	556	555	7	555
31	551	8	551	560	7	556	555	6	555
32	551	7	551	562	7	556	555	6	555
33	551	7	551	555	6	555	555	7	555
34	551	6	551	556	6	555	555	6	555
35	551	7	551	555	7	555	555	6	555
36	551	6	551	555	7	555	555	7	555
37	551	7	551	551	7	551	555	6	555
38	551	7	551	551	7	551	551	6	551
39	551	7	551	551	6	551	551	6	551
40	555	8	551	551	7	551	551	7	551
41	551	7	551	555	7	551	551	6	551
42	551	7	551	551	7	551	551	6	551
43	558	7	551	551	7	551	546	6	546
44	555	7	551	551	6	551	551	7	546
45	551	6	551	555	7	551	546	6	546
46	551	7	551	551	7	551	546	7	546
47	555	7	551	551	6	551	546	6	546
48	546	7	546	546	7	546	550	7	546

49	558	7	546	546	6	546	546	6	546
50	546	7	546	546	7	546	546	6	546
51	546	7	546	546	7	546	546	6	546
52	546	6	546	546	6	546	546	6	546
53	546	7	546	553	7	546	546	6	546
54	546	7	546	546	7	546	546	6	546
55	546	7	546	546	7	546	546	6	546
56	546	7	546	546	6	546	546	7	546
57	546	7	546	546	7	546	546	6	546
58	546	6	546	546	6	546	546	6	546
59	546	7	546	546	6	546	546	7	546
60	553	7	546	553	6	546	546	6	546
61	546	7	546	546	6	546	546	6	546
62	546	7	546	546	7	546	546	6	546
63	551	7	546	546	6	546	546	6	546
64	546	7	546	546	7	546	546	6	546
65	546	7	546	553	7	546	546	7	546
66	546	7	546	546	6	546	546	6	546
67	546	7	546	546	7	546	553	6	546
68	546	6	546	546	7	546	546	6	546

69	546	7	546	546	7	546	546	7	546
70	546	7	546	546	6	546	542	6	542
71	546	7	546	546	7	546	546	6	542
72	546	7	546	546	7	546	542	6	542
73	546	7	546	546	6	546	542	6	542
74	546	7	546	546	7	546	542	6	542
75	546	7	546	546	7	546	546	7	542
76	546	7	546	546	7	546	542	6	542
77	546	7	546	546	7	546	542	7	542
78	546	7	546	546	7	546	542	6	542
79	546	6	546	542	6	542	542	6	542
80	546	6	546	548	6	542	542	7	542
81	546	7	546	542	7	542	542	6	542
82	546	7	546	546	7	542	542	7	542
83	546	7	546	542	7	542	542	6	542
84	546	7	546	542	7	542	542	7	542
85	548	7	546	542	7	542	542	6	542
86	542	6	542	542	6	542	546	7	542
87	546	6	542	542	6	542	542	6	542
88	542	7	542	542	7	542	542	7	542

89	542	7	542	542	7	542	542	6	542
90	542	6	542	542	6	542	542	6	542
91	542	7	542	542	7	542	542	6	542
92	542	7	542	542	7	542	542	7	542
93	542	7	542	542	7	542	542	6	542
94	542	7	542	542	7	542	546	7	542
95	542	7	542	542	7	542	542	6	542
96	542	7	542	542	7	542	542	6	542
97	542	6	542	542	6	542	542	6	542
98	542	7	542	542	7	542	542	7	542
99	542	7	542	542	7	542	542	6	542
100	542	7	542	542	7	542	542	7	542
RESULTADO	542								

Tabla 7.7: Resultados Prueba (SCP42; Ro: 0,1 0,2 0,3; Q0: 0,8 0,9 0,8; Paralelos Colaborativos)

Problema	**SCP42**
Filas	200
Columnas	1000
Optimo	426

ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**
Ro	**0,1**	**Ro**	**0,2**	**Ro**	**0,3**
Qo	**0,8**	**Qo**	**0,9**	**Qo**	**0,8**
Migración	**5**	**Migración**	**5**	**Migración**	**5**
Heurística	**2**	**Heurística**	**2**	**Heurística**	**2**

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Solución	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**
1	577	7	577	594	7	594	577	7	577
2	625	8	577	582	7	582	625	8	577
3	577	7	577	590	7	582	577	7	577
4	593	8	577	565	7	565	593	8	577
5	577	7	577	582	8	565	577	7	577
6	569	7	569	565	6	565	569	7	569
7	565	7	565	565	7	565	568	7	568

8	576	7	565	565	7	565	569	8	568
9	566	7	565	565	7	565	572	7	568
10	565	8	565	565	8	565	568	8	568
11	565	7	565	565	7	565	568	7	568
12	561	8	561	565	7	565	561	7	561
13	565	7	561	565	7	565	570	8	561
14	565	7	561	565	7	565	561	7	561
15	561	8	561	565	7	565	561	7	561
16	565	7	561	565	7	565	561	8	561
17	565	8	561	570	7	565	561	7	561
18	561	7	561	565	7	565	561	7	561
19	561	7	561	565	7	565	561	7	561
20	561	7	561	565	7	565	561	7	561
21	561	8	561	565	7	565	561	8	561
22	561	7	561	561	7	561	561	7	561
23	561	7	561	561	7	561	561	7	561
24	561	8	561	561	7	561	561	7	561
25	561	7	561	561	7	561	561	8	561
26	561	7	561	561	7	561	561	7	561
27	561	8	561	561	7	561	561	7	561

28	561	7	561	561	7	561	561	7	561
29	567	8	561	561	7	561	561	8	561
30	561	7	561	561	7	561	561	7	561
31	561	7	561	561	7	561	561	7	561
32	561	7	561	561	7	561	561	7	561
33	561	8	561	561	7	561	561	7	561
34	561	7	561	561	7	561	561	6	561
35	561	7	561	561	7	561	561	8	561
36	561	7	561	561	7	561	561	7	561
37	561	8	561	561	7	561	561	7	561
38	561	7	561	561	6	561	561	8	561
39	561	8	561	561	7	561	561	7	561
40	561	7	561	561	6	561	561	7	561
41	561	7	561	561	7	561	561	7	561
42	561	7	561	561	6	561	561	7	561
43	570	8	561	561	7	561	570	8	561
44	561	7	561	561	7	561	561	7	561
45	557	7	557	561	7	561	557	7	557
46	561	7	557	561	7	561	561	7	557
47	561	8	557	566	7	561	557	7	557

48	557	7	557	561	7	561	557	7	557
49	557	7	557	561	7	561	557	7	557
50	561	8	557	561	7	561	557	7	557
51	557	7	557	561	7	561	557	8	557
52	557	7	557	561	7	561	557	7	557
53	557	8	557	561	7	561	557	7	557
54	557	7	557	561	7	561	557	8	557
55	557	7	557	561	7	561	557	7	557
56	557	7	557	557	7	557	557	7	557
57	557	7	557	557	7	557	557	7	557
58	557	7	557	557	7	557	557	8	557
59	557	7	557	557	7	557	557	7	557
60	557	7	557	557	7	557	557	7	557
61	557	7	557	561	7	557	557	7	557
62	561	7	557	557	7	557	561	8	557
63	557	7	557	557	7	557	557	7	557
64	557	8	557	557	7	557	557	7	557
65	557	7	557	557	7	557	555	7	555
66	557	8	557	557	7	557	557	7	555
67	557	7	557	557	7	557	557	7	555

68	557	7	557	557	7	557	555	7	555
69	557	7	557	557	7	557	555	7	555
70	557	8	557	557	7	557	555	7	555
71	557	7	557	557	7	557	555	7	555
72	557	7	557	557	7	557	555	7	555
73	557	8	557	557	7	557	555	7	555
74	557	7	557	557	7	557	555	7	555
75	557	7	557	557	7	557	557	7	555
76	561	7	557	557	7	557	560	7	555
77	557	7	557	557	7	557	555	7	555
78	557	7	557	555	7	555	555	8	555
79	557	7	557	555	7	555	555	7	555
80	555	7	555	555	7	555	555	7	555
81	557	8	555	555	7	555	557	8	555
82	557	7	555	555	6	555	555	7	555
83	555	7	555	555	7	555	555	7	555
84	555	8	555	555	8	555	555	7	555
85	555	7	555	555	6	555	555	8	555
86	555	7	555	555	8	555	555	7	555
87	555	8	555	555	6	555	555	7	555

88	557	7	555	555	7	555	555	7	555
89	555	7	555	555	7	555	555	7	555
90	555	7	555	555	7	555	555	8	555
91	555	7	555	555	6	555	555	7	555
92	555	7	555	555	7	555	555	7	555
93	555	7	555	555	7	555	555	7	555
94	555	7	555	555	7	555	555	7	555
95	555	7	555	555	7	555	555	7	555
96	555	7	555	555	7	555	555	8	555
97	555	7	555	555	7	555	555	7	555
98	555	8	555	555	7	555	555	7	555
99	555	7	555	555	7	555	555	7	555
100	555	7	555	555	7	555	555	8	555
RESULTADO	555								

Tabla 7.8: Resultados Prueba (SCP42; Ro: 0,4 0,5 0,4; Q0: 0,4 0,5 0,5; Paralelos Colaborativos)

Problema	**SCP42**
Filas	200

Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**
Ro	**0,4**	**Ro**	**0.5**	**Ro**	**0.4**
Qo	**0.4**	**Qo**	**0.5**	**Qo**	**0.5**
Migración	**5**	**Migración**	**5**	**Migración**	**5**
Heurística	**2**	**Heurística**	**2**	**Heurística**	**2**

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Solución	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**
1	628	7	628	626	7	626	626	7	626
2	616	7	616	612	7	612	612	7	612
3	598	8	598	604	7	604	604	6	604

4	604	6	598	610	7	604	608	6	604
5	598	6	598	606	6	604	606	7	604
6	587	7	587	604	7	604	601	6	601
7	598	7	587	596	7	596	599	7	599
8	587	7	587	592	6	592	601	6	599
9	578	7	578	591	7	591	588	6	588
10	587	7	578	590	7	590	595	7	588
11	578	7	578	590	7	590	590	7	588
12	576	6	576	587	6	587	590	6	588
13	578	6	576	578	7	578	587	7	587
14	576	7	576	587	7	578	578	7	578
15	578	7	576	578	6	578	583	6	578
16	576	7	576	578	7	578	582	7	578
17	576	7	576	578	6	578	578	6	578
18	578	6	576	576	6	576	576	7	576
19	576	7	576	578	7	576	578	6	576
20	578	7	576	576	6	576	576	6	576
21	576	7	576	576	7	576	576	6	576
22	583	7	576	576	6	576	576	7	576
23	576	6	576	578	6	576	578	7	576

24	576	7	576	576	6	576	576	6	576
25	576	6	576	578	7	576	578	6	576
26	576	7	576	576	6	576	576	7	576
27	576	6	576	576	6	576	576	6	576
28	576	7	576	578	6	576	578	6	576
29	576	7	576	576	7	576	576	6	576
30	576	7	576	576	6	576	576	6	576
31	578	7	576	576	7	576	576	6	576
32	576	7	576	576	7	576	576	7	576
33	576	7	576	576	6	576	576	6	576
34	576	7	576	576	7	576	576	7	576
35	576	6	576	576	6	576	576	6	576
36	576	7	576	576	7	576	576	7	576
37	576	7	576	576	7	576	576	7	576
38	576	7	576	576	6	576	576	6	576
39	576	7	576	576	7	576	576	7	576
40	576	7	576	576	7	576	576	7	576
41	569	7	569	576	7	576	576	6	576
42	576	7	569	576	6	576	576	7	576

43	560	6	560	576	7	576	576	7	576		
44	560	7	560	578	7	576	578	6	576		
45	562	6	560	578	6	576	578	7	576		
46	560	6	560	576	6	576	576	6	576		
47	560	7	560	560	6	560	560	6	560		
48	562	7	560	560	6	560	560	7	560		
49	560	6	560	576	6	560	576	6	560		
50	562	7	560	560	7	560	560	7	560		
51	560	7	560	562	6	560	562	6	560		
52	560	6	560	560	7	560	560	7	560		
53	556	7	556	562	6	560	562	6	560		
54	565	7	556	560	7	560	560	7	560		
55	556	7	556	560	6	560	560	6	560		
56	556	6	556	560	7	560	560	7	560		
57	560	7	556	560	7	560	560	6	560		
58	556	6	556	556	6	556	556	6	556		
59	558	7	556	560	6	556	560	6	556		
60	556	7	556	558	7	556	558	6	556		
61	552	6	552	556	6	556	556	7	556		
62	558	7	552	556	7	556	556	6	556		

63	554	7	552	556	6	556	556	7	556		
64	552	7	552	556	7	556	556	6	556		
65	552	6	552	556	6	556	556	6	556		
66	552	6	552	556	6	556	556	7	556		
67	554	7	552	556	7	556	556	6	556		
68	552	6	552	552	7	552	552	7	552		
69	552	7	552	552	6	552	552	6	552		
70	552	7	552	552	7	552	552	7	552		
71	554	6	552	557	6	552	557	6	552		
72	556	7	552	552	7	552	552	7	552		
73	554	7	552	552	7	552	552	6	552		
74	552	7	552	552	6	552	552	7	552		
75	552	7	552	556	6	552	556	6	552		
76	556	6	552	552	7	552	552	6	552		
77	552	7	552	552	7	552	552	7	552		
78	552	6	552	552	6	552	552	7	552		
79	552	7	552	552	6	552	552	6	552		
80	552	7	552	552	6	552	552	7	552		
81	552	6	552	554	6	552	554	6	552		
82	552	7	552	552	6	552	552	6	552		

83	552	7	552	552	7	552	552	7	552
84	552	6	552	552	6	552	552	6	552
85	552	6	552	554	7	552	554	7	552
86	554	7	552	552	6	552	552	6	552
87	552	6	552	552	7	552	552	6	552
88	552	7	552	552	6	552	552	7	552
89	554	6	552	552	7	552	552	6	552
90	554	7	552	552	6	552	552	7	552
91	552	7	552	558	7	552	558	6	552
92	552	7	552	552	6	552	552	6	552
93	552	6	552	552	7	552	552	6	552
94	552	7	552	552	6	552	552	6	552
95	552	6	552	552	6	552	552	6	552
96	552	7	552	554	7	552	554	7	552
97	552	7	552	552	7	552	552	6	552
98	551	7	551	551	6	551	551	7	551
99	550	6	550	552	7	551	552	6	551
100	551	7	550	551	6	551	551	7	551
RESULTADO	550								

Tabla 7.9: Resultados Prueba (SCP61; Ro: 0,1 0,2 0,3; Q0: 0,4 0,5 0,6; Paralelos Colaborativos; Colonia 1: Heurística 1, Colonias 2y3: Heurística 2)

Problema	**SCP61**
Filas	200
Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**
Ro	**0,1**	**Ro**	**0,2**	**Ro**	**0,3**
Qo	**0,4**	**Qo**	**0,5**	**Qo**	**0,6**
Migración	**1**	**Migración**	**1**	**Migración**	**1**
Heurística	**1**	**Heurística**	**2**	**Heurística**	**2**

	Colonias								
ITERACIONES	**Col 1**			**Col 2**			**Col 3**		
	Solución	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**	**Solución**	**Tiempo**	**Acumulado**
1	343	7	343	170	7	170	162	6	162

2	737	9	343	183	7	170	165	7	162
3	314	6	314	163	6	163	162	6	162
4	289	7	289	162	7	162	160	7	160
5	336	8	289	163	7	162	160	6	160
6	313	6	289	164	7	162	159	6	159
7	289	6	289	161	6	161	158	7	158
8	301	7	289	160	7	160	159	6	158
9	260	7	260	160	6	160	160	6	158
10	301	7	260	160	7	160	157	6	157
11	260	6	260	161	6	160	156	6	156
12	260	7	260	165	7	160	158	6	156
13	260	6	260	160	7	160	156	6	156
14	221	6	221	159	6	159	156	7	156
15	197	6	197	156	7	156	156	6	156
16	230	7	197	159	6	156	156	6	156
17	230	6	197	156	7	156	156	6	156
18	229	6	197	156	6	156	156	6	156
19	216	6	197	156	6	156	156	6	156
20	216	6	197	156	7	156	156	6	156
21	205	6	197	156	6	156	157	6	156

22	190	5	190	158	7	156	156	7	156
23	192	6	190	156	6	156	156	6	156
24	196	6	190	158	7	156	156	6	156
25	193	5	190	156	6	156	157	7	156
26	190	6	190	156	6	156	156	6	156
27	196	6	190	156	7	156	156	6	156
28	189	6	189	156	6	156	156	7	156
29	192	6	189	156	7	156	156	6	156
30	205	6	189	156	6	156	156	6	156
31	205	6	189	156	7	156	156	7	156
32	205	6	189	156	6	156	156	6	156
33	196	6	189	156	7	156	156	6	156
34	196	7	189	156	6	156	156	6	156
35	166	5	166	156	6	156	156	5	156
36	205	7	166	156	6	156	156	7	156
37	183	6	166	156	6	156	156	6	156
38	171	6	166	156	6	156	155	6	155
39	180	6	166	156	7	156	156	6	155
40	196	6	166	156	6	156	155	7	155
41	196	6	166	155	7	155	155	6	155

42	189	6	166	156	6	155	155	6	155
43	187	6	166	156	7	155	155	7	155
44	170	5	166	155	6	155	155	6	155
45	183	6	166	155	6	155	155	6	155
46	191	6	166	156	7	155	155	6	155
47	167	5	166	155	6	155	155	6	155
48	191	7	166	155	7	155	155	6	155
49	176	5	166	155	6	155	155	7	155
50	198	6	166	155	6	155	155	6	155
51	185	6	166	155	7	155	155	6	155
52	176	6	166	155	6	155	155	6	155
53	192	5	166	155	6	155	155	6	155
54	195	6	166	155	7	155	155	6	155
55	182	6	166	155	6	155	155	6	155
56	182	6	166	155	7	155	155	6	155
57	182	6	166	155	6	155	155	6	155
58	191	6	166	155	7	155	155	6	155
59	188	6	166	155	6	155	155	6	155
60	188	6	166	155	7	155	155	6	155
61	170	5	166	155	6	155	155	6	155

62	186	6	166	155	7	155	155	6	155
63	167	6	166	158	6	155	155	7	155
64	170	6	166	155	7	155	155	6	155
65	182	6	166	155	6	155	155	7	155
66	176	6	166	155	7	155	155	6	155
67	185	6	166	155	6	155	155	7	155
68	167	5	166	155	6	155	155	6	155
69	167	6	166	155	7	155	156	6	155
70	170	6	166	155	7	155	155	6	155
71	173	6	166	155	6	155	155	6	155
72	185	6	166	155	6	155	155	6	155
73	170	6	166	155	7	155	155	6	155
74	182	6	166	155	6	155	155	7	155
75	167	5	166	155	6	155	155	6	155
76	173	6	166	155	7	155	155	6	155
77	179	6	166	155	6	155	155	7	155
78	185	6	166	155	6	155	155	6	155
79	165	6	165	155	6	155	155	6	155
80	179	6	165	155	7	155	155	7	155
81	176	5	165	156	6	155	155	6	155

82	176	6	165	155	7	155	155	6	155
83	179	6	165	155	6	155	155	7	155
84	185	6	165	155	6	155	155	6	155
85	179	6	165	155	7	155	155	6	155
86	170	6	165	155	6	155	155	7	155
87	167	5	165	155	6	155	158	6	155
88	185	6	165	155	6	155	156	7	155
89	185	6	165	155	6	155	155	6	155
90	173	6	165	155	6	155	155	6	155
91	176	6	165	155	6	155	155	6	155
92	167	6	165	155	7	155	155	6	155
93	179	5	165	155	6	155	155	6	155
94	185	6	165	155	6	155	155	6	155
95	170	6	165	155	7	155	158	6	155
96	179	6	165	155	6	155	155	7	155
97	170	5	165	155	7	155	155	6	155
98	185	6	165	155	6	155	155	6	155
99	173	6	165	155	7	155	155	7	155
100	176	6	165	155	6	155	155	6	155
RESULTADO	176 55								

Tabla 7.10: Resultados Prueba (SCP62; Ro: 0,1 0,2 0,3; Q0: 0,4 0,5 0,6; Paralelos Colaborativos, Colonia 3: Heurística 1, Colonias 1y2: Heurística 2)

Problema	**SCP62**
Filas	200
Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**	**Nº Hormigas**	**10**
Iteraciones	**100**	**Iteraciones**	**100**	**Iteraciones**	**100**
Ro	**0,1**	**Ro**	**0,2**	**Ro**	**0,3**
Qo	**0,4**	**Qo**	**0,5**	**Qo**	**0,6**
Migración	**1**	**Migración**	**1**	**Migración**	**1**
Heurística	**2**	**Heurística**	**2**	**Heurística**	***1***

	Colonias		
ITERACIONES	**Col 1**	**Col 2**	**Col 3**

	Solución	Tiempo	Acumulado	Solución	Tiempo	Acumulado	Solución	Tiempo	Acumulado
1	176	6	176	188	5	188	369	5	369
2	182	5	176	170	5	170	392	6	369
3	177	6	176	180	5	170	369	4	369
4	180	5	176	165	5	165	366	4	366
5	166	6	166	170	6	165	209	4	209
6	168	5	166	160	5	160	349	5	209
7	162	5	162	160	5	160	213	3	209
8	160	6	160	164	5	160	184	4	184
9	162	5	160	160	5	160	167	4	167
10	159	5	159	165	5	160	190	4	167
11	160	5	159	160	5	160	172	4	167
12	159	6	159	164	5	160	166	4	166
13	159	5	159	159	5	159	166	3	166
14	159	5	159	160	5	159	169	3	166
15	159	5	159	159	5	159	171	4	166
16	159	5	159	159	5	159	159	4	159
17	159	5	159	157	5	157	159	4	159
18	159	5	159	159	5	157	159	4	159
19	159	6	159	159	5	157	159	4	159

20	157	5	157	157	5	157	159	4	159
21	157	5	157	157	4	157	159	4	159
22	157	5	157	157	5	157	159	4	159
23	157	5	157	157	5	157	159	4	159
24	157	6	157	157	5	157	157	3	157
25	157	5	157	157	5	157	159	3	157
26	157	5	157	157	5	157	161	3	157
27	157	5	157	157	5	157	157	4	157
28	157	5	157	157	5	157	157	4	157
29	157	5	157	157	5	157	159	3	157
30	157	6	157	157	5	157	161	3	157
31	157	5	157	157	5	157	159	4	157
32	157	5	157	157	5	157	159	4	157
33	159	5	157	157	5	157	159	4	157
34	157	5	157	157	5	157	163	4	157
35	157	4	157	159	5	157	159	4	157
36	157	5	157	157	5	157	159	3	157
37	157	6	157	157	5	157	159	4	157
38	157	5	157	157	5	157	157	4	157
39	157	5	157	157	5	157	157	4	157

40	157	6	157	157	5	157	159	4	157
41	157	5	157	157	5	157	157	3	157
42	157	5	157	157	5	157	159	4	157
43	157	5	157	157	5	157	159	4	157
44	157	5	157	157	5	157	157	4	157
45	157	5	157	157	5	157	157	3	157
46	157	6	157	157	5	157	159	4	157
47	157	5	157	157	5	157	159	4	157
48	157	5	157	157	4	157	159	4	157
49	159	5	157	157	4	157	157	4	157
50	157	5	157	157	4	157	159	4	157
51	157	5	157	157	5	157	159	4	157
52	157	5	157	159	5	157	159	4	157
53	157	5	157	157	5	157	159	4	157
54	157	6	157	157	5	157	159	3	157
55	157	5	157	157	5	157	159	3	157
56	157	5	157	157	5	157	159	4	157
57	159	6	157	157	5	157	157	4	157
58	157	5	157	157	5	157	157	4	157
59	157	5	157	157	4	157	157	4	157

60	159	5	157	157	5	157	157	3	157
61	157	5	157	157	5	157	157	4	157
62	159	5	157	157	5	157	157	4	157
63	157	6	157	157	5	157	157	4	157
64	157	5	157	157	5	157	157	4	157
65	157	5	157	157	5	157	157	4	157
66	157	5	157	159	5	157	157	3	157
67	157	5	157	157	5	157	157	3	157
68	157	5	157	157	5	157	157	4	157
69	157	6	157	157	5	157	157	4	157
70	157	5	157	157	5	157	157	4	157
71	157	5	157	157	5	157	159	4	157
72	157	5	157	157	5	157	159	4	157
73	159	5	157	157	5	157	159	4	157
74	157	5	157	157	5	157	157	4	157
75	157	6	157	157	4	157	157	3	157
76	159	5	157	157	5	157	157	4	157
77	157	5	157	157	5	157	157	4	157
78	161	6	157	157	5	157	157	4	157
79	157	5	157	157	5	157	157	3	157

80	157	5	157	157	5	157	157	4	157
81	157	5	157	163	5	157	157	4	157
82	157	6	157	157	5	157	157	4	157
83	157	5	157	159	5	157	157	4	157
84	157	5	157	157	5	157	157	4	157
85	159	5	157	157	5	157	157	4	157
86	157	5	157	157	5	157	157	4	157
87	157	5	157	157	5	157	157	3	157
88	157	5	157	159	5	157	159	4	157
89	159	5	157	157	5	157	159	4	157
90	157	5	157	157	5	157	159	4	157
91	157	5	157	159	5	157	159	4	157
92	157	5	157	157	5	157	159	4	157
93	157	5	157	157	5	157	159	4	157
94	157	5	157	157	5	157	159	3	157
95	157	5	157	159	5	157	159	4	157
96	157	5	157	157	5	157	159	4	157
97	157	6	157	157	5	157	159	4	157
98	157	5	157	157	5	157	159	4	157
99	157	5	157	157	5	157	157	4	157

100	159	5	157	157	5	157	159	3	157
RESULTADO	157								

Tabla 7.11: Resultados Prueba (SCP41; Ro: 0,9 0,1; Q0: 0,1 0,9; Paralelos Colaborativos, Colonias Heurística 1)

Problema	**SCP41**
Filas	200
Columnas	1000
Optimo	426
ACS	463

Nº Hormigas	**10**	**Nº Hormigas**	**10**
Iteraciones	**50**	**Iteraciones**	**50**
Ro	**0.9**	**Ro**	**0,1**
Qo	**0.1**	**Qo**	**0,9**
Migración	**1**	**Migración**	**1**
Heurística	**1**	**Heurística**	**1**

Colonias HEURISTICA 1

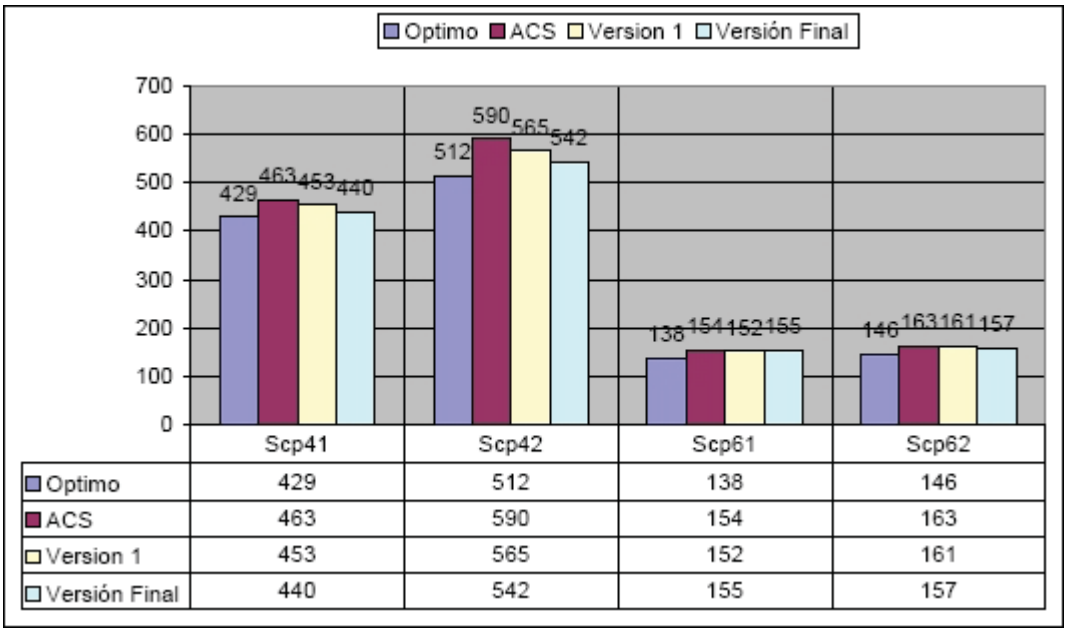
ITERACIONES	Col 1			Col 2		
	Solución	Tiempo	Acumulado	Solución	Tiempo	Acumulado
1	714	9	714	5502	4	5502
2	1373	14	714	5502	4,5	5502
3	714	9	714	1883	3,5	1883
4	714	10	714	5502	4	1883
5	714	10	714	2588	4	1883
6	714	8	714	1883	4	1883
7	714	10	714	1883	3,5	1883
8	724	9	714	1235	3,5	1235
9	728	10	714	1053	3,5	1053
10	714	8	714	1235	3,5	1053
11	714	9,5	714	1235	3,5	1053
12	740	9	714	1053	3,5	1053
13	714	8	714	1053	3,5	1053
14	727	8,5	714	817	3,5	817
15	727	8,5	714	817	4	817
16	727	9,5	714	817	3	817
17	711	11	711	817	3,5	817
18	727	9	711	817	3,5	817

19	709	9,5	709	817	3,5	817
20	711	9,5	709	811	3,5	811
21	709	9	709	784	3,5	784
22	709	8,5	709	811	3,5	784
23	723	8,5	709	784	3,5	784
24	709	8,5	709	784	3,5	784
25	698	8,5	698	784	3	784
26	709	6,5	698	784	3	784
27	712	8	698	784	3	784
28	680	7,5	680	784	4	784
29	712	8,5	680	784	3,5	784
30	680	8	680	784	3,5	784
31	680	7,5	680	784	3,5	784
32	680	9	680	784	3,5	784
33	680	9	680	724	3,5	724
34	694	9,5	680	784	3,5	724
35	680	7,5	680	784	3,5	724
36	680	7,5	680	784	3,5	724
37	671	8,5	671	734	3,5	724
38	680	8	671	784	3,5	724

39	671	8	671	784	3,5	724
40	671	7	671	774	3,5	724
41	671	8,5	671	784	3,5	724
42	671	7,5	671	784	3,5	724
43	671	7,5	671	784	3,5	724
44	671	7	671	784	3,5	724
45	671	8	671	784	3,5	724
46	671	8	671	784	3,5	724
47	671	7	671	797	3,5	724
48	671	7,5	671	797	3,5	724
49	671	9,5	671	781	3,5	724
50	645	8,5	645	197	4	197
RESULTADO	197					

7.7 Conclusiones de las Pruebas

Gráfico de Resumen de Resultado de las mejores Prestaciones:



	Scp41	Scp42	Scp61	Scp62
Optimo	429	512	138	146
ACS	463	590	154	163
Version 1	453	565	152	161
Versión Final	440	542	155	157

Ilustración 7.8: Gráfico Comparación de Resultados Finales.

7.8 Aportes Generales del Proyecto

A partir del estudio de cada uno de los procesos desarrollados, se deben mencionar cuales han sido los conceptos que son de aporte a los actuales desarrollos y estado del arte del ámbito del proyecto.

El principal concepto es el desarrollo de una plataforma de búsqueda configurable y distribuida, la cual permite que si comparamos la situación antes del presente desarrollo se presenta un aporte en tiempo y esfuerzo necesario para encontrar la mejor solución a un problema de optimización combinatorial.

Por ejemplo, si trataramos de resolver cualquier problema, tendríamos que determinar una serie de parámetros, donde cada uno de ellos tiene una serie de posibles valores a elegir, en detalle sería algo como:

- *Número de hormigas* que desarrollarán la búsqueda, donde solo si acotamos a múltiplos de 5, entre 5 y 100, nos encontraríamos con ***20*** posibles valores.

- *Parámetro Ro*, debería determinarse entre 0 y 1, donde si tomamos en consideración solo las opciones con una décima deberíamos elegir entre 0,1 y 1 las opciones serían de ***10*** posibles valores.

- *Parámetro Qo*, tiene un número de posibilidades igual a Ro, por lo tanto las opciones son igualmente de ***10*** posibles valores.

- *Número de iteraciones*, parámetro que podría variar en un número ilimitado, ya que solo dependería de cuanto uno determinaría como un número adecuado de iteraciones del proceso, el cual además dependerá de las características de complejidad del problema. Solo acotando las opciones a números múltiples de 10, las opciones entre 10 y 200, nos encontraríamos con ***20*** posibles valores.

- *La Heurística*, es otro parámetro que además de deber cambiar la configuración se debería cambiar la codificación de la aplicación. En nuestro caso, para buscar una analogía con el proceso desarrollado, se podrían elegir entre ***2*** heurísticas base para el desarrollo del proceso.

Entonces, en resumen, si alguien quisiera determinar cual es la mejor solución para un problema determinado, tendría que realizar tantos experimentos como combinaciones posibles de parámetros se mencionaron antes, solo con los valores acotados podrían llegar a 80.000 combinaciones, lo cual claramente sería de gran dificultad y perdida de tiempo.

Dicho problema, se aborda con procesos paralelos y distribuidos, donde si bien llegar a un número tal de combinaciones sigue siendo dificultoso, por el proceso de determinación y configuración de parámetros inicial. Pero claramente se puede optimizar el proceso, ya que en vez de realizar por ejemplo 50 procesos de búsquedas consecutivos, se puede realizar un solo proceso, pero con 50 instancias en paralelo. Tal como se puede apreciar en la ilustración siguiente. Por otra parte, por las características del proceso, este permite que por conceptos de procesos colaborativos, se logre llegar a un mejor resultado del obtenido por procesos

independientes. Es decir, solo por conceptos de paralelismo y colaboración se logra obtener mejoras al proceso.

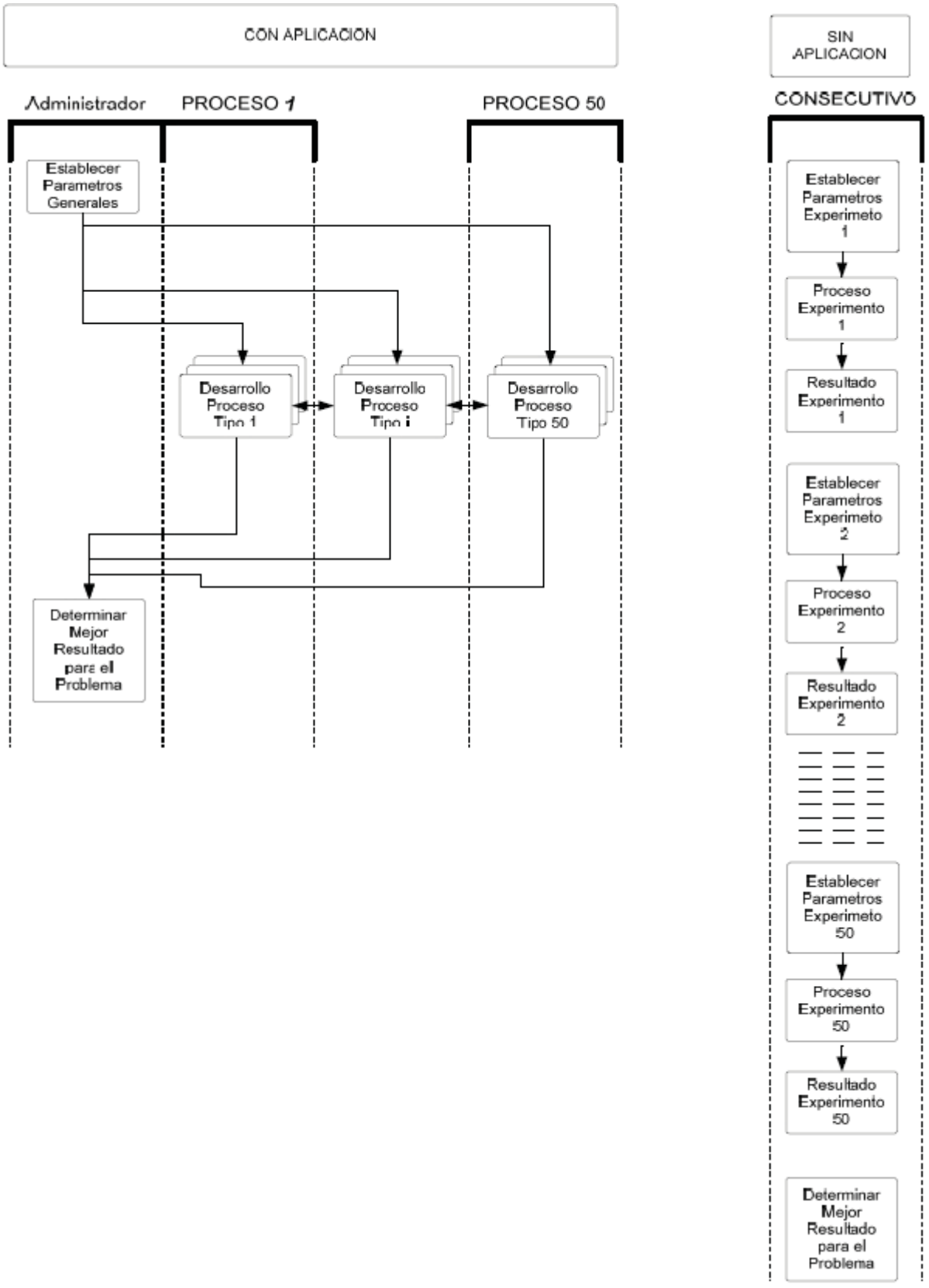


Ilustración 7.9: Esquema Comparación con y sin Paralelismo.

Conclusiones

En un sistema de búsqueda de soluciones a problemas combinatoriales, se puede optar por diversas técnicas, pero usando las técnicas incompletas no se asegura el efectivo logro de encontrar el óptimo, además se ve altamente condicionado por las características de los algoritmos utilizados y lo complejo del problema.

En los últimos tiempos se han desarrollado este tipo de sistemas, pero aplicándolo a problemas concretos, como por ejemplo del “vendedor viajero”, y de manera tradicional, algoritmo aislado a un problema aislado. Sin embargo, estas soluciones no presentan mayores características de nuevos aportes.

Desde esta perspectiva es que se buscó incorporar nuevas técnicas y herramientas que permitan involucrar otro tipo de problemas y métodos de desarrollo. Específicamente ligado a la perspectiva del proceso de desarrollo, es que la utilización de distintos tipos de algoritmos ACO genera tomar en cuenta sus características de funcionamiento y configuración.

De una forma más concreta, el trabajo aquí presentado desarrolla nuevas técnicas y métodos que permiten que un problema combinatorial sea resuelto de manera colaborativa y distribuida, en el cual a partir de múltiples colonias de hormigas se lleva a cabo un proceso de búsqueda en paralelo. Para ello, se ha extendido el funcionamiento colaborativo de las colonias de manera interna a un funcionamiento colaborativo entre colonias, donde a partir de la migración y comunicación de los mejores resultados obtenidos por cada colonia se mejoran los tiempos y resultados globales.

A partir del desarrollo de las características entes mencionadas, es que fue posible observar tras el desarrollo, las siguientes características generales:

Al momento de desarrollar pruebas con idénticos parámetros de entrada, pero una de ellas en paralelo y la otra de manera independiente, se pudo observar que en el primer caso todas las colonias llegaron al mismo óptimo costo, mientras que en el segundo

caso solo una de ellas llegó al óptimo encontrado en paralelo y las otras colonias tuvieron resultados de peor calidad.

En relación al punto anterior se puedo observar, que a pesar del efecto de los distintos parámetros de entrada para colonia, si el proceso se desarrolla en paralelo, los resultados igual tienden a converger a un mismo valor.

Si se evalúan exclusivamente los resultados desde el punto de vista de la calidad de ellos, claramente queda de manifiesto una mejora estable en la mayoría de los experimentos realizados, inclusive así uno inferir que si realizan un número mayor de iteraciones y más experimentos se logrará determinar puntualmente cuales son los mejores parámetros de entrada al proceso.

Dentro de los parámetros configurables de la plataforma de búsquedas, el que marco una diferencia ostensible en la calidad de los resultados, fue la elección de que heurística se utilizaba en el proceso de selección de la siguiente columna a agregar a la solución, siendo la heurística que esta determinada por: “Cantidad de filas que se le agregan a la solución actual, dividido por el costo que tiene sumar dicha columna” la que demostró y entregó los resultados de calidad antes comentados. Mientras que la heurística que solo toma en consideración el costo de las columnas mostró resultados bastante más bajos.

La característica asíncrona del proceso, provocó el mayor avance en relación al tiempo de resolución de los problemas, ya que se pudo aprovechar de mejor manera las distintas capacidad de los equipos participantes del proceso, además provocó que si por alguna razón alguno de los equipos que contenían a una colonia *caían,* no provocaba la caída del experimento.

En relación al punto anterior hay que mencionar un aspecto llamativo, ya que al momento de desarrollar el proceso en paralelo de colonias utilizando heurísticas distintas, durante el transcurso de un número ya interesante de iteraciones, más de 30, la constante influencia de la(s) colonia(s) que trabajan con la mejor heurística provoca que la colonia con heurística de peor desempeño se acerquen o inclusive lleguen a los mismos resultados.

El tiempo, que es uno de los factores a evaluar como resultado del desarrollo mostró que el tiempo promedio de desarrollo de cada iteración es de aproximadamente 6 segundos y medio.

Si bien es posible realizar un estimativo de tiempo, es necesario estipular que existen factores que igual lo afectan y los cuales no pueden ser controlados por el proceso, como por ejemplo, las características de la red de comunicaciones, donde por ejemplo la saturación o uso intensivo de la red puede provocar retardo en los procesos.

Siguiendo con el punto anterior, otro aspecto no manejable directamente se refiere a las características internas de los equipos computacionales, ya que si bien uno puede realizar un análisis si cumplen con los requerimientos mínimos, también hay que considerar el uso o carga del equipo mediante los desarrollos de las pruebas, esto se refleja principalmente en la cantidad de aplicaciones que se están ejecutando al mismo tiempo: antivirus, escritorio remoto, actualizaciones, anti spyware, etc, etc.

Si bien, como se mencionó anteriormente hay factores que afectan el rendimiento, el desarrollo de las pruebas no marcó por parte de la aplicación un uso excesivamente alto de los recursos de memoria ni procesador.

Otro aspecto que afecta la utilización de redes de comunicación son variables del entorno como por ejemplo, políticas de seguridad y confiabilidad de la red.

Al momento de llevar a cabo una evaluación sobre el desarrollo, Se deben destacar algunos aspectos importantes, el principal se refiere a la mejora obtenida luego de los cambios efectuados al proceso, especialmente los relacionados a la nueva forma de desarrollo y la nueva plataforma.

A partir de las aplicaciones y evaluaciones desarrolladas con anterioridad, también provocaron que el último desarrollo permitiera una plataforma de búsquedas configurable.

En resumen, en este documento se presentaron las mejoras obtenidas en cuanto a calidad y velocidad de obtención de soluciones a problemas combinatoriales SCP mediante la utilización de un algoritmo ACO ACS, implementado en una estructura distribuida utilizando comunicación asíncrona.

A modo de análisis de resultados y de lógica de funcionamiento, se pueden inferir los siguientes aspectos:

Los parámetros de entrada afectan considerablemente el proceso de búsqueda de la solución.

La heurística que toma en consideración las filas que se le agregarán a la solución, es claramente superior y se acerca bastante a los resultados esperados como obtenidos del desarrollo.

El funcionamiento en paralelo si afecta el desarrollo de búsqueda de cada colonia, si bien los resultados no presentan grandes diferencias con funcionamiento independiente, son marginalmente mejores y no presentan un castigo por retardo de la comunicación.

Sobre la arquitectura de distribución de los procesos, no existió diferencia en cuanto a resultados, pero si se presenta un uso mayor de la red.

A continuación presento las consideraciones y evaluaciones personales sobre el desarrollo, las cuales se refieren principalmente al logro de los objetivos.

Como postulante al titulo de ingeniero civil informático, es que me encontré ante un problema muy poco estudiado y obviamente con mínimo desarrollo práctico. Pero en este momento final me encuentro con un conocimiento del ámbito y características del proceso mucho más detallado, donde me es posible determinar que aspectos son críticos para el desarrollo de este tipo de problemas.

Además, se logró cumplir con un objetivo que cualquier informático siempre busca y que además se contemplaba como objetivo del desarrollo, que es obtener una aplicación Software que cumpliera con las características de ser configurable, de interfaz dinámica y que además, a partir del estudio conceptual previo, entregara buenos resultados.

Por lo tanto es posible resumir, en cuanto a mi evaluación. Que se logró obtener un software que permite configurar algoritmos ACO para que resuelvan problemas de cobertura de conjuntos de manera colaborativa y paralela. Que además entregaron buenos resultados en cuanto a tiempo y que me permitieron determinar mejoras en la calidad de los mismos, determinados en gran manera por la configuración del algoritmo.

En fin, el proyecto y específicamente el software, permite obtener los resultados en un menor tiempo y que además puede asegurar que se obtendrá el mejor resultado posible al configurar los parámetros de búsqueda de la mayoría de las combinaciones de parámetros posibles.

Pero este trabajo aquí presentado no es un punto final a esta línea de investigación, sino un punto seguido que deja pendientes algunas cosas y abre la puerta a otras más. A continuación se nombran algunos de los temas posibles a desarrollar e investigar en el futuro:

Otorgar mayores características de intefaz y procesos de evaluación de los resultados, por parte del mismo software.

Aumentar las características, para que abarque la mayoría de los tipos de algoritmos ACO, especialmente aquellos que también se manejen con parámetros de configuración.

Resolver otro tipo de problema combinatorial, como por ejemplo Set Partitioning o Knapsack.

Incorporación de nuevas técnicas o hibridación de los procesos.

Referencias

[1] E. Alba, C. Blum, A. Roli. “An introduction to metaheuristic techniques”. In E. Alba, editor, Parallel Metaheuristics, Wiley Series on Parallel and Distributed Computing. Wiley, 2005

[2] E Alba, G.Leguizamón, G. Ordoñez “Analyzing the Behavior of Parallel Ant Colony Systems for Large Instances of the Task Scheduling Problem” 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) -Workshop 6 p. 191b. 2005.

[3] E. Alba, J. M. Troya. "An Analysis od Synchronous and Asynchronous Parallel Distributed Genetic Algorithms with Structured and Panmictic Islands", Proceedings of the BioSP3 Workshop, Zomaya A.Y., Ercal F., Olariou S. (eds.), Springer-Verlag, 1999.

[4] U. Aickelin “A New Genetic Algorithm for Set Covering Problems“, Annual Operational Research Conference 42, Swansea, UK, 2000.

[5] B. Barán, M. Almirón, “Colonia de Hormigas en un Ambiente Paralelo Asíncrono” XXVIII Conferencia Latinoamericana de Informática CLEI’2002, Montevideo Uruguay 2002.

[6] B. Barán, E. Chaparro y N. Cáceres., “A-Teams en la Optimización del Caudal Turbinado de una Represa Hidroeléctrica”, *Conf. Iberoam. Intelig. Arti.l IBERAMIA’98*, Portugal, 1998.

[7] B. Bullnheimer, R. F. Hartl, y C. Strauss. “A new rank-based version of the Ant System: A computational study”. Central European Journal for Operations Research and Economics, 7:1, páginas 25-38, 1999.

[8] B. Bullnheimer., G. Kotsis. y C. Strauss. “Parallelization Strategies for the Ant System”, Reporte Técnico POM 9/97, Universidad de Viena, Viena – Austria, 1997.

[9] A. Caprara, H. Kellerer, U. Pferschy, and D. Pisinger. Approximation algorithms for knapsack problems with cardinality constraints. European Journal of Operational Research, 123:333--345, 2000.

[10] S. Chu, J.F. Roddick, and Pan, J. 2004. Ant colony system with communication strategies. Inf. Sci. Inf. Comput. Sci. 167, 1-4 , 63-76. DOI= http://dx.doi.org/10.1016/j.ins.2003.10.013, Diciembre 2004

[11] O. Cordón, I. Fernández de Viana, F. Herrera, y L. Moreno. “A new ACO model integrating evolutionary computation concepts: The Best-Worst Ant System”. En M. Dorigo, M. Middendorf, y T. Stützle, editores, Abstract proceedings of ANTS2000 -From Ant Colonies to Artificial Ants: A series of International Workshops on Ant Algorithms, páginas 22-29. IRIDIA, Université Libre de Bruxelles, Belgium, 2000.

[12] O. Cordón, F. Herrera, L. Moreno. “Integración de Conceptos de Computación Evolutiva en un Nuevo Modelo de Colonias de Hormigas”. VIII Conferencia de la Asociación Española para la Inteligencia Artificial, (Seminario Especializado en Computacion Evolutiva), Murcia (España), Vol. II, páginas 98-105.1999.

[13] B. Crawford, C.Castro. "Integrating Lookahead and Post Processing Procedures with ACO for Solving Set Partitioning and Covering Problems", Eighth International Conference on Artificial Intelligence and Soft Computing, ICAISC, 2006

[14] B. Crawford, C. Castro. “Combination of Constraint Programming Techniques and ACO for the Set Partitioning and Covering Problems” I Workshop in Constraint Programming. Facultad Ingeniería, Universidad de Concepción, Concepción, Chile. 2006.

[15] P. Delisle, M. Gravel, M. Krajecki, C. Gagné., W.L. Price, "A shared memory parallel implementation of Ant Colony Optimization", Proceedings of the 6th Metaheuristics International Conference (MIC'2005), Vienne, Autriche, 22-26, pp. 257-264. 2005

[16] M. Dorigo, L. M. Gambardella, “Ant colony system: A cooperative learning approach to the traveling salesman problem”. IEEE Transactions on Evolutionary Computation, 1(1):53–66, 1997.

[17] M. Dorigo y L. Gambardella, “A study of some properties of Ant-Q”. IV Int. Conf. on Parallel Problem from Nature, Berlin – Alemania: Springer-Verlag, pp. 656-665, 1996.

[18] M. Dorigo y G. Di Caro, The ant colony optimization meta-heuristic, in D. Corne, M. Dorigo and F. Glover (eds), New Ideas in Optimization, McGraw-Hill, London, pp. 11–32. 1999.

[19] M. Dorigo, T. Stutzle. “Ant Colony Optimization”. MIT Press, USA, 2004.

[20] M. Dorigo, T. Stutzle. “The ant colony optimization metaheuristic: Algorithms, applications and advances”. In F. Glover and G. Kochenberger, editors, Handbook of Metaheuristics, pages 251–285. Kluwer, 2002. Also available as Technical Report IRIDIA/2000-32, IRIDIA Universite Libre de Bruxells, Belgica. ftp://iridia.ulb.ac.be/pub/mdorigo/tec.reps/TR.11-MetaHandBook.pdf

[21] I. Foster. “Designing and Building Parallel Programs”. Addison-Wesley Publishing Company, Reading, MA, 1995.

[22] L. Gambardella y M. Dorigo, “Ant-Q: a reinforcement learning approach to the traveling salesman problem”. In:Proc.ML-95,12th Intnl.Conf. Machine Learning, pp. 252260 (Morgan Kaufmann, Palo Alto, CA). 1995.

[23] X. Gandibleux, V. Delorme, T'Kindt “An Ant Colony Algorithm for the Set Packing Problem” In Ant Colony Optimization and Swarm Intelligence (M. Dorigo, M. Birattari, Ch. Blum, L. Gambardella, Fr. Mondada, and Th. Stutzle Edts).ANTS 2004, Fourth International Workshop, Brussels, Belgium, Sep 5-8, 2004.

[24] R. Hadji, M. Rahoual, E. Talbi, V. Bachelet, “Ant colonies for the set covering problem”. In M. Dorigo et al., editors, Proceedings of ANTS2000, pages 63–66.2000.

[25] G. Leguizamón, “Current Applications of Ant Systems for Subset Problems” Proyecto: Sistemas Inteligentes para Scheduling y Control. Universidad Nacional de San Luis – Argentina.
http://www.educ.ar/educar/superior/biblioteca_digital/verdocbiblio.jsp?url=S_BD_WICK2000/TRA3_06.PDF&contexto=superior/biblioteca_digital/

[26] G. Leguizamon, Z. Michalewicz. “Ant Systems for subset problems”. Unpub. manuscript, 2000.

[27] L. Lessing. “Ant colony optimization for the set covering problem”. Master’s thesis, Fachgebiet Intellektik, Fachbereich Informatik, TU Darmstadt, Germany, 2004.

[28] L. Lessing, I. Dumitrescu, T. Stutzle, “A Comparison Between ACO Algorithms for the Set Covering Problem”, M. Dorigo et al. (Eds.): ANTS 2004, LNCS 3172, pp. 1–12, 2004, Springer-Verlag Berlin Heidelberg 2004.

[29] D. Levine. “Application of a hybrid genetic algorithm to airline crew scheduling“. Computers & Operations Research, 23(6):547-558, 1996

[30] D. Levine. “A Parallel Genetic Algorithm for the Set Partitioning Problem“, Report ANL-94/23, Argonne National Laboratory, May, 1994.

[31] M. Manfrin and M. Birattari and T. Stützle and M. Dorigo. “Parallel Ant Colony Optimization for the Traveling Salesman Problem”. Technical Report TR/IRIDIA/2006-007. IRIDIA, Université Libre de Bruxelles, Brussels, Belgium. 2006. http://iridia.ulb.ac.be/IridiaTrSeries/IridiaTr2006-007r001.pdf

[32] V. Maniezzo, M. Milandri, “An ant-based framework for very strongly constrained problems”, in M. Dorigo, G. Di Caro and M. Sampels (eds), 3rd International Workshop on Ant Algorithms, ANTS2002, Brussels, Belgium, Vol. 2463 of Lecture Notes in Computer Science, Springer-Verlag, pp. 222–227. 2002

[33] M. Middendorf, F. Reischle, H. Schmeck. Information Exchange in Multi Colony Ant Algorithms. In Proceedings of the Workshop on Bio-Inspired Solutions to Parallel Processing Problems, LNCS 1800, pages 645-652. Springer Verlag, 2000

[34] Set covering problems, http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/

[35] C. Solnon y Derek Bridge, ”An Ant Colony Optimization Meta-Heuristic for Subset Selection Problems” Capítulo Nº1 del Libro “System Engineering using Particle Swarm Optimization”, paginas 7-29, Marzo2006. http://www710.univ-lyon1.fr/~csolnon/publications/PSO2006.pdf

[36] T. Stutzle, “MAX − MIN ant system for the quadratic assignment problem”, Technical Report AIDA-97-04, Darmstadt University of Technology, Computer Science Department, Intellectics Group. 1997

[37] T. Stutzle, M. Dorigo, “ACO algorithms for the traveling salesman problem”, in K. Miettinen, M. Makela, P. Neittaanmaki and J. Periaux (eds), Evolutionary Algorithms in Engineering and Computer Science, Wiley, pp. 163–183. 1999

[38] T. Stutzle, M. Dorigo, “A short convergence proof for a class of ant colony optimization algorithms”, IEEE Transactions on Evolutionary Computation 6(4): 358– 365. 2002.

[39] T. Stutzle, H. Hoos, “Improving the ant system: A detailed report on the MAX − MIN ant system”, Technical Report AIDA-96-12 -Revised version, Darmstadt University of Technology, Computer Science Department, Intellectics Group. 1996

[40] T. Stutzle, H. Hoos, “The MAX − MIN ant system and local search for combinatorial optimization problems: Towards adaptive tools for combinatorial global optimization”, in S. Voss, S. Martello, I. Osman and C. Roucairol (eds), Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, Kluwer Academic Publishers, Boston, MA, pp. 313–329. 1998.

[41] T. Stutzle, H. Hoos, “MAX − MIN ant system, Future Generation Computer Systems” 16: 889–914. 2000.

[42] T. Stützle, “Parallelization Strategies for Ant Colony Optimization”, Proc. of Parallel Problem Solving from Nature – PPSN-V, Springer Verlag, Vol. 1498, pp. 722-731, 1998.

[43] A. Tanenbaum. “Organización de computadoras, un enfoque estructurado”, 4ta Edición. Prentice Hall, 2000.

[44] Training and Education Documents (EPCC, the Edinburgh Parallel Computing Centre). http://www.epcc.ed.ac.uk/computing/training/document_archive/

[45] D. Van Veldhuizen, J. B. Zydallis, y G. B. Lamont. “Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation, 7(2):144–173, 2003.

[46] I. Osman, J. P. Kelly, “Meta-heuristics: Theory & Applications”, Kluwer Academic Publishers, Boston. 1996.

ANEXOS

Texto Anexos en PDF, en: