Universidad Nacional de Asunción

Facultad Politécnica

OPTIMIZACIÓN MULTI-OBJETIVO BASADA EN COLONIAS DE HORMIGAS. TEORÍA Y ESTRATEGIAS DE PARALELIZACIÓN

JULIO PACIELLO

HÉCTOR MARTÍNEZ

PROYECTO DE TRABAJO DE GRADO PRESENTADO EN CONFORMIDAD A LOS REQUISITOS PARA OBTENER EL GRADO DE INGENIERO EN INFORMÁTICA.

ASESOR DEL TRABAJO
D. Sc. BENJAMÍN BARÁN CEGLA

SAN LORENZO, 2006

Agradecimientos

A nuestras familias, por habernos comprendido y apoyado en todo este tiempo de mucho trabajo. Sin sus apoyos no hubiera sido posible realizar esta tesis.

Al profesor Benjamín Barán, por sus aportes y sugerencias brindados a lo largo de todo el desarrollo este trabajo, y por todo el tiempo que nos dedicó para mejorarlo.

A las demás personas que de alguna forma colaboraron con nosotros en la realización de este trabajo, ya sea motivando o colaborando con ideas. No especificamos nombres, pero estas personas saben a quienes nos referimos.

Dedicatoria

En memoria de mis padres, quienes fueron mi sustento incondicional mientras completaba mis estudios universitarios y quienes hubieran estado orgullosos de este trabajo.

En especial, a mi hija Bianca, quien simplemente fundamenta mis ganas de salir adelante en la vida. Le dedico este primer gran paso para mí.

Julio Paciello Coronel

A mis padres, quienes a lo largo de toda mi vida me dieron todo lo necesario para que yo siga adelante. Gracias a su comprensión, cariño y enseñanzas me dieron fuerzas suficientes para terminar esta carrera universitaria y empezar así a intentar brindar un legado similar al que ellos me brindaron.

Héctor Martínez Santacruz

Resumen

Las colonias de hormigas reales fueron estudiadas por los científicos en décadas anteriores, muchos de ellos buscaron comprender el comportamiento natural de estos insectos en su búsqueda de alimentos. Claramente, este tipo de insectos así como otros, presenta un comportamiento natural que a partir de mecanismos simples consigue objetivos muy llamativos que pueden ser considerados "inteligentes" dado que para las personas requerirían de razonamiento.

Un ejemplo llamativo de este comportamiento son las hormigas en su búsqueda de alimentos, que por medio de una sustancia química que segregan son capaces, trabajando como colonia, de encontrar el camino más corto desde su nido a una fuente de alimento. Cabe resaltar que las hormigas son insectos ciegos, y logran el comportamiento mencionado con el trabajo grupal y una comunicación indirecta producida al segregar una sustancia química que las demás hormigas pueden percibir.

La teoría de la meta-heurística ACO ("*Ant Colony Optimization"*) fue propuesta como una teoría que modela el comportamiento natural de las hormigas reales, y es utilizada con éxito en la resolución de problemas de optimización multi-objetivo. Este trabajo describe la teoría ACO y presenta distintos algoritmos existentes e inspirados en esta teoría.

Experimentalmente, se realiza una comparación entre diversos algoritmos ACO utilizando tres problemas de prueba bi-objetivos, el "*Quadratic Assignment Problem"* (problema de asignación cuadrática), el "*Traveling Salesman Problem"* (problema del cajero viajante) y el "*Vehicle Routing Problem with Time Windows"* (problema de ruteo de vehículos con ventanas de tiempo), considerando diversos algoritmos ACO que constituyen el estado del arte en la teoría basada en colonias de hormigas.

Como propuesta original de este trabajo, se presenta un equipo de algoritmos ACO basado en el paralelismo para resolver problemas multi-objetivos. Además, se propone un nuevo algoritmo multi-objetivo, el "*Multiobjective Ant System"*, y se verifica un buen comportamiento empírico en los problemas mencionados. Se verifica el efecto cooperativo del equipo de algoritmos y la robustez del mismo ante los diferentes problemas utilizados. Empíricamente se demuestra que una estrategia de utilizar una única tabla de feromonas y múltiples visibilidades supera a otras propuestas como alternativa de diseño para un algoritmo ACO multi-objetivo.

Abstract

Natural Ant colonies have been studied by scientists in the past decades. Some works intended to understand the natural behavior of these insects when they search for food. Clearly, ants and other insects have the ability of achieving "considerable" objectives (objectives that humans could consider to require reasoning) using very simple mechanisms.

An interesting example of the previously mentioned behavior is the way ants, working as a colony, can find the shortest path between their nest and a food source by means of a chemical substance called pheromone. It is worth mentioning that ants are blind insects, and that they accomplish the mentioned behavior by working together communicating with each other indirectly using a chemical substance that other ants can perceive.

The theory of ACO (Ant Colony Optimization) meta-heuristic was proposed as a theory that models the behavior of natural ants. Currently, this theory is successfully applied to some multiobjective optimization problems. The present work describes the ACO theory and presents several existing algorithms based on this theory.

An experimental comparison using a benchmark of three bi-objective problems, Traveling Salesman Problem (TSP), Quadratic Assignment Problem (QAP) and Vehicle Routing Problem with Time Windows (VRPTW), of the existing state-of-art algorithms in the resolution of multiobjective problems using ant colony theory is made.

This work proposes a Team of ACO algorithms based on parallel processing to solve multiobjective problems. In addition, a new approach, the Multiobjective Ant System, is proposed proving its good behavior. For the above mentioned multiobjective problems a good robustness was observed when applying the Team of ACO algorithms to the benchmark problems. Experimental results prove that the strategy of having only one table of pheromones and multiple visibilities empirically outperforms other strategies.

Índice de contenido

Índice de Figuras

Índice de Tablas

Símbolos Utilizados

$\vec{x}$	Vector de decisión.
ω	Cantidad de restricciones.
$g_i(\vec{x})$	Funciones de restricción.
$f_1(\vec{x}), f_2(\vec{x}), \ldots, f_b(\vec{x})$	Funciones objetivo.
$\vec{f}(\vec{x})$	Vector de funciones objetivo.
b	Cantidad de objetivos a optimizar.
Ω	Dominio de soluciones factibles.
Ω_o	Conjunto imagen de Ω.
w_j	Masa del aspa *j*.
ϵ_j	Valores seleccionados como límites para las restricciones.
δ_i	Pesos relativos.
$G=(N, Ar)$	Grafo.
N	Número de nodos.
Ar	Número de arcos.
c_{ij}	Costo asociado al arco *i*, *j*.
$A=\{a_{ij}\}\in\mathbb{R}^{n\times n}$	Matriz de distancias.
a_{ij}	Distancia entre localidades *i* y *j*.
$B^q=\{b_{ij}^q\}\in\mathbb{R}^{n\times n}, q=1,\ldots, b$	Matriz de flujos.
b_{ij}^q	*q*-ésimo flujo entre instalaciones ubicadas en las localidades *i* y *j*
d_i	Vector de posición.
M_i	Momento de un aspa ubicada en la posición *i*.
v	Cantidad de vehículos necesarios para el VRPTW.
n	Cantidad de clientes.
$C=\{c_0, c_1, \ldots, c_n\}$	Conjunto de *n* clientes, c_0 representa el depósito y c_i el cliente *i*.
ψ_i	Ruta del *i*-ésimo camión.
$t_{j,j+1}$	Tiempo de viaje entre el cliente c_j y el siguiente cliente c_{j+1}.

Q	La capacidad máxima de un vehículo.
q_i	Demanda del cliente c_i , con $q_i \leq Q$ y $q_0 = 0$.
s_i	Representa el tiempo de servicio en el cliente c_i .
$[b_i, e_i]$	Ventana de tiempo aceptable del cliente *i*, donde b_i es el tiempo más temprano de servicio del cliente c_i y e_i es el máximo tiempo de atención del cliente c_i .
τ	Rastros de feromonas.
η	Visibilidad.
τ_0	Valor inicial de feromonas.
m	Cantidad de hormigas.
J_i	Vecindario de estados alcanzables desde *i.*
α	Importancia relativa de las feromonas.
β	Importancia relativa de la visibilidad.
$p_{i,j}$	Probabilidad de utilización de estado *j* estando en el estado *i.*
ρ	Coeficiente de evaporación.
$\Delta\tau$	Cantidad de feromonas a depositar.
l	Numero de hormigas elitistas.
μ	*Ranking* de una solución elitista.
Z	Factor de normalización del AS.
$\Delta\tau'$	Cantidad de feromonas a depositar en una actualización elitista.
gb	Mejor solución global.
ib	Mejor solución de iteración.
fs	Factor de suavizamiento MMAS.
S	Mejor solución obtenida utilizando solo la visibilidad.
gw	Peor solución global.
P	Población del OA.
γ	Paso de aprendizaje.
λ	Importancia relativa entre objetivos.
τ_{max}	Valor máximo impuesto a la cantidad de feromonas.
τ_{min}	Valor mínimo impuesto a la cantidad de feromonas.

$\hat{f}_1$	Evaluación promedio de soluciones utilizando la primera función objetivo.
$\hat{f}_2$	Evaluación promedio de soluciones utilizando la segunda función objetivo.
K	Número de iteraciones.
O	Cantidad total de feromona a depositar (MOA).
UP	Unidad de Procesamiento.
NPr	Número de Unidades de Procesamiento.
$F(X)$	Problema a dividir.
$\zeta_i(x)$	Subproblema a resolver.
G_i	Algoritmo a utilizar.
$d(p,q)$	Distancia euclídea entre dos puntos p y q.
Y_{true}	Frente Pareto óptimo.
Y'	Frente Pareto generado por un algoritmo particular.
$M1'$	Métrica de desempeño 1.
$M2'$	Métrica de desempeño 2.
σ	Distancia mínima entre dos puntos para considerarlos distribuidos.
$M3'$	Métrica de desempeño 3.
E	Métrica de Error.

Siglas y Abreviaturas

A-Team	*Asynchronous Team*. Equipo asíncrono de algoritmos.
ACO	*Ant Colony Optimization*. Optimización basada en colonias de hormigas.
ACS	*Ant Colony System*. Sistema basado en colonias de hormigas. Ver [Dorigo1997].
AS	*Ant System*. Sistema basado en hormigas. Ver [Dorigo1996].
ASrank	*Ranked Ant System*. Sistema de hormiga basado en "*ranking*". Ver [Bullnheimer1999].
BIANT	*Bicriterion Ant*. Hormigas bi-criterio. Ver [Iredi2001].
BIMC	*Bicriterion Multi-Colony*. Multi-colonias bi-criterio. Ver [Iredi2001].
COMP	CompetANTs. Ver [Doerner2003].
MAS	*Multiobjective AS*. Ant System multi-objetivo. Propuesta de este trabajo.
MMAS	*Max-Min Ant System*. Sistema Max-Min basado en hormigas. Ver [Stutzle2000].
M-MMAS o M3AS	*Multiobjective MMAS*. Max-Min Ant System multi-objetivo. Ver [Pinto2005].
MOA	*Multiobjective OA*. Omicron ACO multi-objetivo. Ver [Gardel2005].
MOACO	*Multiobjective Ant Colony Optimization*. Optimización multi-objetivo basada en colonias de hormigas.
MOACS	*Multiobjective ACS*. Ant Colony System multi-objetivo. Ver [Barán2003].
MOAQ	*Multiobjective Ant Q*. Ant Q multiobjetivo. Ver [Mariano1999].
OA	Omicron ACO. Ver [Gómez2004].
PACO	Pareto ACO. Ver [Doerner2002].

QAP	*Quadratic Assignment Problem*. Problema de asignación cuadrática.
TA	*Team Algorithm*. Equipo de algoritmos.
TSP	*Traveling Salesman Problem*. Problema del cajero viajante.
VRPTW	*Vehicle Routing Problem with Time Windows*. Problema de ruteo de vehículos con ventanas de tiempo.

Capítulo 1

Introducción

En este capítulo se presentan los antecedentes y la importancia de este trabajo, así como las propuestas que se realizan. Conceptos como optimización multi-objetivo, las colonias de hormigas y los equipos de algoritmos, son de gran importancia para el trabajo por lo que son brevemente introducidos en este capítulo. No obstante, en capítulos siguientes, se tratarán todos estos temas en detalle.

1.1. Antecedentes y Relevancia

Las aplicaciones reales de ingeniería y otras áreas generalmente involucran la optimización simultánea de varios objetivos, y la toma decisiones puede estar relacionada a la alternativa que optimice al mismo tiempo varios criterios generalmente contradictorios (ver [Coello1999, Deb1999]). De esta manera, la optimización multi-objetivo se basa en la teoría de la optimilidad Pareto y es un área de creciente interés. Esta teoría de Pareto es presentada en detalle en [Veldhuizen1999] y menciona conceptos y definiciones matemáticas precisas de manera a poder tratar problemas desde el punto de vista multi-objetivo "puro" sin utilizar métodos tradicionales de optimización mono-objetivo.

La optimización multi-objetivo es sin duda un importante tópico de investigación, no solo por la naturaleza multi-objetiva de muchos problemas reales sino también por el hecho de existir muchas cuestiones todavía pendientes en esta área. De hecho, como se menciona en [Coello1999], no existe hasta el momento una definición universalmente aceptada de "óptimo" en el contexto multi-objetivo, como si lo existe en la optimización mono-objetivo. Esto hace difícil comparar los

resultados entre distintos métodos de optimización utilizados, y generalmente la decisión sobre la "mejor" respuesta corresponde a la persona que toma las decisiones (*decisión-maker*).

Al existir una variedad de objetivos, generalmente en conflicto, aparece el concepto de un "conjunto" de soluciones óptimas antes que una única mejor solución. Este concepto es introducido en la teoría de Pareto y estas soluciones óptimas son conocidas como soluciones Pareto óptimas, en honor al trabajo publicado por Vilfredo Pareto en el año 1896.

Como ninguna solución Pareto óptima puede ser clasificada como mejor que otra solución Pareto óptima, sin intervención del *decisión maker*, el objetivo de la optimización multi-objetivo es encontrar la mayor cantidad de soluciones Pareto óptimas [Deb1999].

En la actualidad, la optimización multi-objetivo es tratada por diversos paradigmas de programación, entre los cuales se pueden citar los métodos estocásticos como los algoritmos evolutivos [Coello1999, Coello2002] y los algoritmos basados en el comportamiento de las hormigas reales [Denenbourg1990, Goss1989], cuya técnica es denominada optimización basada en colonia de hormigas ("*Ant Colony Optimization", ACO*) [Dorigo1999]. A la fecha fueron ya propuestos diferentes algoritmos ACO multi-objetivos denominados MOACO *("MultiObjective ACO"),* [Barán2003, Doerner2002, Doerner2003, Gardel2005, Iredi2001, Mariano1999, Pinto2005], y se debe esto al creciente interés científico ante la necesidad de optimizar estos algoritmos de manera a poder resolver problemas de alta complejidad, muchos de ellos clasificados como NP-Completos [Sahni1976]. Como ejemplo de este tipo de problemas podemos citar diversos problemas de ingeniería [Gardel2005, Pinto2005], logística [Barán2003, Doerner2003] y otras áreas.

Como se menciona en [Sahni1976], los problemas NP-Completos no pueden ser tratados de manera eficaz con métodos de búsqueda exhaustiva que exploren completamente el espacio de soluciones del problema, y es en estos casos donde actualmente se aplican técnicas estocásticas como las mencionadas anteriormente también clasificadas como computación evolutiva, basada tanto en algoritmos genéticos como en colonias de hormigas, entre otros. La computación evolutiva se fundamenta en mejoras iterativas a partir de soluciones iniciales, generalmente aplicando métodos estocásticos, para luego utilizar estas soluciones de manera a guiar el proceso de búsqueda y encontrar aun soluciones mejores en una siguiente iteración (denominada generación). De esta forma el proceso es iterativo y encuentra soluciones aplicando métodos estocásticos, "guiado" por las buenas soluciones encontradas previamente.

Continuando con la optimización basada en colonia de hormigas, un problema estudiado por científicos es comprender la manera en que animales ciegos como las hormigas consiguen establecer el camino más corto entre su colonia y las fuentes de alimento. Fue demostrado [Denenbourg1990, Goss1989] que el medio utilizado para comunicar información entre individuos, acerca del camino utilizado y para decidir por donde ir, consiste en rastros (huellas) de *feromonas* (una sustancia química que las hormigas pueden oler).

Una hormiga deposita feromonas (en cantidades variables) en el suelo a medida que se mueve, dejando rastros de esta sustancia en el camino utilizado. El movimiento de una hormiga aislada es esencialmente aleatorio [Denenbourg1990, Goss1989]; sin embargo, una hormiga ubicada en una zona donde previamente fueron depositadas feromonas, puede detectar estos rastros y decidir con alta probabilidad seguir el mismo camino, reforzando las feromonas a medida que avanza. El comportamiento colectivo resultante es una forma de autocatálisis, en el cual mientras más hormigas siguen un rastro de feromonas, más atractivo se convierte el camino utilizado por estas hormigas. Entonces, la probabilidad de una hormiga de escoger un camino crece a medida que se incrementa el número de hormigas que previamente escogieron dicho camino. En el capítulo 4 se muestra como este comportamiento natural logra encontrar el camino más corto entre el nido de las hormigas y las fuentes de alimento.

Por su naturaleza iterativa, la computación evolutiva también requiere ser optimizada en cuanto a tiempo de respuesta de las ejecuciones. Un paradigma utilizado en la mejora del desempeño de los algoritmos es la computación paralela, aprovechando el poder de cómputo disponible de manera a obtener resultados en tiempos razonables [Kumar2003]. Entre las técnicas basadas en el paralelismo aparece el equipo de algoritmos *(Team Algorithm)* [Barán2002, Talukdar1983], el cual es una técnica basada en el uso de los recursos computacionales disponibles, dada la amplia disponibilidad de recursos computacionales y redes de computadoras actuales, generalmente asíncronas, de manera a tratar problemas complejos.

Generalmente los algoritmos disponibles para optimización y satisfacción de restricciones poseen debilidades. En lugar de intentar diseñar un nuevo algoritmo sin debilidades, una tarea difícil de realizar, se podría utilizar una forma de organización de distintos algoritmos de manera a disminuir las debilidades a través de la cooperación entre los mismos. Así, se pretende conseguir resultados en forma conjunta, que posiblemente no hubiera podido conseguir ninguno de los

algoritmos trabajando de manera individual. Este tipo de organización es denominado equipo de algoritmos.

La técnica del equipo de algoritmos combina la utilización de diferentes algoritmos para la resolución del mismo problema. De esta manera, cada algoritmo trabaja en equipo para resolver el problema y típicamente es asignado a un procesador distinto en una red de computadoras. Además, normalmente se utiliza un proceso *Administrador* encargado de recolectar las soluciones aportadas por los distintos algoritmos del equipo que generalmente guía a los mismos desde una perspectiva global [Barán2002]. Un equipo de algoritmos evolutivos basados en los algoritmos genéticos fue propuesto por Fernández *et al.* en [Fernández2005].

Como ejemplo práctico, cabe mencionar el trabajo realizado por Talukdar *et al.* en [Talukdar1991], en el cual fue implementado un *A-Team (Asynchronous Team)* combinando Algoritmos Genéticos con reconocidos métodos numéricos, como el Método de Newton, aplicados a la resolución de sistemas algebraicos de ecuaciones no lineales. De esta forma, se obtuvo una combinación asíncrona de algoritmos reduciendo significativamente los recursos computacionales que demandaría la sola utilización de métodos numéricos tradicionales, reportándose muy buenas aceleraciones (*speedup*), y un interesante efecto de sinergia entre los distintos métodos.

1.2. Propuesta de esta tesis y Objetivos

Esta tesis propone por primera vez un equipo distribuido de algoritmos MOACO para resolver problemas multi-objetivos. Se resuelven tres reconocidos problemas de prueba bi-objetivos, el "*Traveling Salesman Problem"* (TSP) [Gómez2004] , el "*Quadratic Assignment Problem"* (QAP) [Çela1998], y el "*Vehicle Routing Problem with Time Windows"* (VRPTW) [Barán2003]. Estos problemas son considerados clásicos en la literatura de optimización combinatoria y del tipo NP-completos [Sahni1976]. Los resultados experimentales son comparados con los obtenidos por cada algoritmo MOACO que compone el equipo.

Otro objetivo de esta tesis es presentar una comparación entre los diversos algoritmos MOACO que constituyen el estado del arte en la optimización multi-objetivo basada en colonia de hormigas. El enfoque utilizado está orientado a comparar experimentalmente los algoritmos MOACO y el equipo de algoritmos.

En el contexto de realizar comparaciones experimentales entre algoritmos MOACO; este trabajo puede ser considerado como una extensión del trabajo realizado por García-Martínez *et al.* en

[Garcia2004], incluyendo algoritmos propuestos recientemente como el M-MMAS [Pinto2005], el MOA [Gardel2005] y dos nuevas propuestas de esta tesis, el MAS *("Multiobjective Ant System")* y el "*Team Algorithm"*. Además se realizaron pruebas con un mayor conjunto de problemas.

Al resolver los tres reconocidos problemas mencionados, se pretende finalmente contar con un *framework* de algoritmos MOACO implementados, además de los procesos *Administrador y Esclavo* necesarios para el equipo de algoritmos. De esta forma, las técnicas de modularización de software utilizadas facilitarán la escalabilidad y adaptación de los algoritmos a nuevos problemas. Se espera además, que problemas reales y con aplicaciones prácticas que puedan ser modelados como un *TSP*, *QAP*, o *VRPTW*, sean adaptables para ser resueltos con alguno de los algoritmos implementados a un costo de esfuerzo casi nulo.

Como será mencionado en los siguientes capítulos, el conjunto de problemas resueltos posee diversas aplicaciones prácticas importantes, entre las cuales se pueden destacar la optimización de rutas para aviones, y otros medios de transporte, el programa *Starlight* de la NASA que recolecta imágenes de cuerpos celestiales, el problema del balanceo de turbinas ampliamente aplicado en motores con turbinas, entre otros que serán mencionados más adelante.

Debido a estas importantes aplicaciones reales citadas previamente, capaces de ser modeladas como uno de los problemas resueltos en esta tesis, se considera también de importancia este trabajo, ya que facilita la utilización de distintos métodos (algoritmos), además de poder ser aplicable a casos reales y prácticos.

1.3. Organización del trabajo

El trabajo está organizado como sigue: en el capítulo 2 se trata la formulación matemática de la optimización multi-objetivo. Se presentan definiciones y otros métodos clásicos en la optimización multi-objetivo.

El capítulo 3 presenta la formulación de los tres problemas a ser resueltos en este trabajo y sus aplicaciones prácticas modeladas como los problemas resueltos en este trabajo.

El capítulo 4 presenta la teoría de la metaheurística ACO, y se muestra la forma en que el modelo de comportamiento de las hormigas reales es utilizada en la computación. En el capítulo 5 se presentan algunos algoritmos ACO propuestos además de todos los algoritmos MOACO utilizados en este trabajo y que constituyen el estado del arte en esta teoría.

El capítulo 6 introduce la computación paralela y describe algunas estrategias de paralelización aplicadas a los algoritmos basados en colonias de hormigas. En el capítulo 7 se presenta la teoría relacionada a los algoritmos en equipo, así como también se presenta el equipo de algoritmos propuesto en esta tesis.

En el capítulo 8 se presenta la implementación computacional, considerando las métricas de desempeño utilizadas al comparar los algoritmos y los resultados experimentales obtenidos.

Finalmente el capítulo 9 presenta las conclusiones del trabajo y se sugieren trabajos futuros de interés que puedan surgir a partir de esta tesis.

Capítulo 2

Optimización Multi-objetivo

La optimización multi-objetivo puede ser definida como el problema de encontrar un vector de variables de decisión que satisfacen restricciones y optimiza un vector de funciones cuyos elementos representan las funciones objetivo. Estas definiciones aparecen en los trabajos de Coello [Coello1999] y Deb [Deb1999].

Las funciones objetivo representan los criterios que el diseñador desea optimizar simultáneamente, generalmente en conflicto entre sí. Por ejemplo, en la elección de rutas se podrían tener dos funciones objetivo como el costo de los caminos utilizados y el tiempo total de viaje. Por lo tanto, la optimización multi-objetivo significa encontrar un conjunto de soluciones óptimas, considerando simultáneamente todas las funciones objetivo.

2.1. Definición formal

Formalmente, en la optimización multi-objetivo se desea encontrar un vector de decisión

$$\vec{x} = [x_{1,} x_{2,}, \ldots, x_n]^T \tag{2.1}$$

generalmente con $\vec{x} \in \mathbb{R}^n$, que deberá satisfacer ω restricciones de desigualdad

$$g_i(\vec{x}) \geq 0 \qquad i = 1,2,\ldots,\omega \tag{2.2}$$

y optimizar el vector de funciones objetivo

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_b(\vec{x})]^T \tag{2.3}$$

que generalmente cumple con $\vec{f}(\vec{x}) \in \mathbb{R}^b$. El conjunto de todas las soluciones que atienden 2.2 es conocido como dominio de soluciones factibles, y se representa como Ω, en general $\Omega \subset \mathbb{R}^n$.

El correspondiente conjunto imagen Ω_o se define como:

$$\Omega_o = \{ \vec{f}(\vec{x}) \in \mathbb{R}^b \mid \vec{x} \in \Omega \} \tag{2.4}$$

2.2. Optimalidad Pareto

La teoría de la optimalidad Pareto proporciona definiciones matemáticas precisas de los conceptos multi-objetivos generales, como las soluciones no comparables y las soluciones dominadas. Estos conceptos se ilustran en la tabla 1.1, donde cada situación posible entre dos soluciones dadas $u, v \in \Omega$ genera una consecuencia acerca de la comparación entre las mismas.

Tabla 1.1. Teoría de la optimalidad Pareto

Situación	Consecuencia
u domina a *v*	*u* es mejor solución que *v*
v domina a *u*	*v* es mejor solución que *u*
Ninguna de las anteriores	*u* y *v* son soluciones no comparables

Basados en la teoría de la optimalidad Pareto podemos dar una definición formal acerca del concepto de dominancia mencionado en la tabla 1.1 y denominada en la optimización multi-objetivo como dominancia Pareto.

Definición 1: *Dominancia Pareto:* Sean dos soluciones $u, v \in \Omega$. Se dice que *u* domina a *v* (denotado como $u \succ v$) si es mejor o igual que *v* en cada uno de los objetivos y estrictamente mejor en al menos un objetivo.

Como ejemplo, en un contexto de minimización $u \succ v$ si y solo si:

$$f_i(u) \leq f_i(v) \; \forall i \in \{1,2,\ldots,b\} \; \wedge \; \exists j \in \{1,2,\ldots,b\} \mid f_j(u) < f_j(v) \tag{2.5}$$

Considerando una función de dos objetivos a minimizar y dado un punto específico $\vec{f}(u) \in \Omega_0$, podemos dibujar una semirrecta paralela al eje de las abcisas y otra paralela al eje de las ordenadas que parten del punto $\vec{f}(u)$ orientadas hacia el infinito en caso de minimizar y hacia menos infinito en caso contrario. La región formada se denomina región de dominancia del punto *u,* como se ilustra en la figura 2.1.

Todas las soluciones ubicadas en la región de dominancia de *u* son soluciones dominadas por *u*. Conociendo la definición de dominancia Pareto se pueden mencionar otras definiciones utilizadas en el contexto de la optimización multi-objetivo.

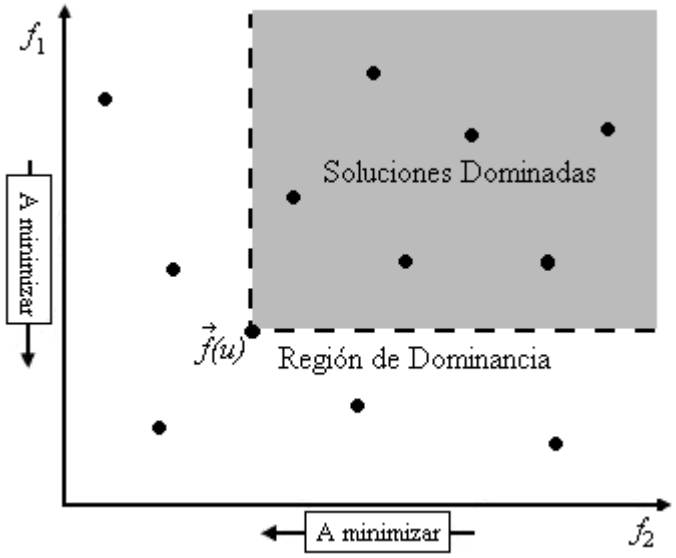

Figura 2.1. Región de dominancia del punto $f(u) \in \Omega_0$

Definición 2: *Soluciones no comparables:* Dados $u, v \in \Omega$, si $u \not\succ v$ ni $v \not\succ u$, se dice que son soluciones no comparables, lo que se denota como $u \sim v$.

Definición 3: *Conjunto Pareto:* El conjunto de todas las soluciones $\vec{x}$ no dominadas en Ω se denomina Conjunto Pareto, lo que se denota como *CP*. Las soluciones $\vec{x}$ que pertenecen a *CP* se denotarán como x^*.

Definición 4: *Frente* Pareto*:* La imagen del Conjunto Pareto a través de la función $\vec{f}$ se denomina Frente Pareto, denotado por *Y*.

Como ejemplo podemos minimizar la función $\vec{f}_2$ de Schaffer [Zitzler1998] definida como:

$$\vec{f}_2(x) = \{x^2, (x-2)^2\} \tag{2.6}$$

En la figura 2.2 se muestran ambas funciones objetivo trazadas en el dominio acotado para $x \in [-3,4]$, y su imagen a través de la función $\vec{f}_2$ se muestra en la figura 2.3.

Como podemos observar en el ejemplo, las soluciones Pareto óptimas de este problema corresponden al intervalo $x \in [0,2]$. Estos puntos producen la sección resaltada de la curva en la

figura 2.3, que constituyen así al Frente Pareto del problema. El espacio de búsqueda de soluciones también es denominado dominio de decisión, que en el ejemplo corresponde al dominio de la variable de decisión *x*. Mientras que el dominio de la función objetivo también es denominado espacio solución del problema.

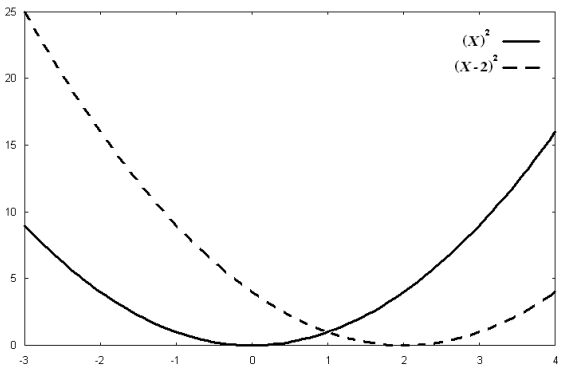

Figura 2.2. Funciones objetivo en el dominio$x \in [-3,4]$.

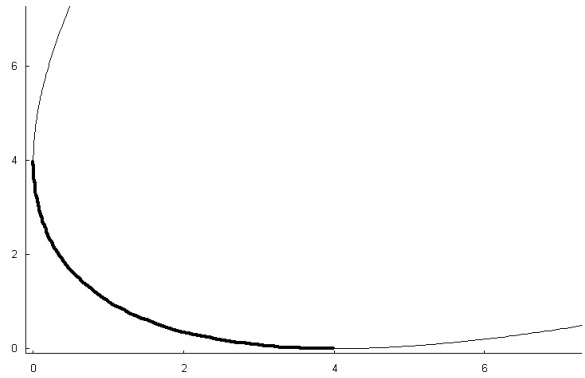

Figura 2.3. Frente Pareto para la función de Schaffer.

En consecuencia, podemos afirmar que la optimización multi-objetivo se centra en la búsqueda de un conjunto de soluciones óptimas denominado Conjunto Pareto, y en el cual todas las soluciones son no comparables entre sí. Estas soluciones no comparables representan para el diseñador un conjunto de alternativas de decisión donde cada una es mejor que las demás en al menos un objetivo del problema.

2.3. Métodos clásicos no basados en la Optimalidad Pareto

En la literatura existen métodos alternativos de optimización multi-objetivo que generalmente generan una versión mono-objetivo a partir del problema multi-objetivo a resolver. Estos métodos son clasificados como métodos clásicos [Deb1999] y a continuación se presentan dos métodos reconocidos y generalmente muy utilizados en la minimización y maximización de funciones matemáticas, dentro del contexto de la ingeniería.

2.3.1. Método de suma ponderada

Las múltiples funciones objetivo son combinadas en una función general $\vec{f}$. En un contexto de minimización, esto se define como sigue:

$$\begin{array}{cc} \textit{Minimizar} & \vec{f} = \sum_{j=1}^{b} \delta_j \cdot f_j(\vec{x}) \\ \textit{donde} & \vec{x} \in \Omega \\ \textit{y generalmente} & \sum_{j=1}^{b} \delta_j = 1 \end{array} \tag{2.7}$$

Al agregar las funciones objetivo en una única función se obtiene una única solución óptima, correspondiente a la combinación lineal de valores asignados a los pesos δ_j. Para obtener diferentes soluciones Pareto óptimas, se pueden asignar diferentes pesos δ_j a las funciones objetivo y volver a optimizar la función $\vec{f}$.

Una consideración a tener en cuenta al utilizar este método es la necesidad de normalización de los valores de las funciones objetivo en caso que las mismas posean distintos dominios de definición e inclusive estén medidas con distintas magnitudes. Como ejemplo consideremos el caso de utilizar una función objetivo de costo medida en dólares y en el intervalo de 1.000 a 100.000 dólares, y otra función de distancia recorrida medida en kilómetros y en el intervalo de 800 a 2000 kilómetros. Claramente puede notarse la necesidad de normalizar estos valores para posteriormente poder agregarlos de manera ponderada, en caso contrario se estarían agregando valores inconsistentes entre sí.

2.3.2. Método de perturbación ϵ

Se construye un problema de optimización mono-objetivo en el cual una función objetivo es tomada como la función a optimizar y las demás son utilizadas como restricciones. En un contexto de minimización se define formalmente como:

$$\begin{array}{cc} \textit{Minimizar} & f_k(\vec{x}) \quad , k \in \{1,2,\ldots,b\} \\ \textit{Sujeto a} & f_j(\vec{x}) \leq \epsilon_j \quad , \forall\, j \neq k \\ \textit{donde} & \vec{x} \in \Omega \end{array} \tag{2.8}$$

Para encontrar una solución Pareto óptima, valores apropiados de ϵ_j son escogidos para la *j*-ésima función objetivo con $j \neq k$, donde $k \in \{1,2,\ldots,b\}$ y f_k representa la *k*-ésima función objetivo a optimizar. Estos valores apropiados de ϵ requieren un conocimiento a priori del problema. Para encontrar diferentes soluciones Pareto óptimas, el procedimiento se repite asignando diferentes valores a cada ϵ_j con $j \in \{1,2,\ldots,b\}$.

Capítulo 3

Formulación de los Problemas Considerados

Muchos problemas reales pueden ser modelados mediante la teoría de grafos [Stutzle2002]. Generalmente, un problema real de ingeniería u otras áreas implica una compleja optimización combinatoria, y por ende es clasificado como NP-Completo [Sahni1976]. En este capítulo se formulan tres de este tipo de problemas NP-Completos y que son generalmente modelados mediante grafos con pesos asociados a cada arista que interconecta los nodos.

En el caso particular de los problemas formulados en este trabajo, las soluciones corresponden a permutaciones de los nodos correspondientes al grafo que modela el problema. Por este motivo, estos problemas son referidos como de "alta" complejidad, ya que la optimización consiste en una búsqueda de las permutaciones factibles y óptimas, de acuerdo a la optimalidad Pareto, de entre todas las permutaciones.

Se puede notar fácilmente que estando las permutaciones en función al número de nodos del grafo (n), el espacio de búsqueda de permutaciones crece con una tasa no polinómica con respecto a n. Así, computacionalmente, podemos estimar una cota inferior para este espacio en función al número de nodos mediante la expresión $n!$ (factorial de n). Claramente, resulta impráctico utilizar métodos de búsqueda exhaustiva para resolver este tipo de problemas considerando una gran cantidad de nodos (*n grande*), y resulta razonable utilizar otros métodos heurísticos, como los mencionados en este trabajo.

A continuación se presenta la formulación matemática de los siguientes problemas:

1- TSP (*Traveling Salesman Problem*).
2- QAP *(Quadratic Assignment Problem)*.
3- VRPTW *(Vehicle Routing Problem)*.

3.1. Traveling Salesman Problem (TSP)

El TSP o problema del cajero viajante puede ser representado como un grafo completo $G=(N, Ar)$, donde N es el número de nodos, o ciudades, y Ar es el conjunto de arcos que interconectan completamente los nodos. Cada arco $(i, j)\in Ar$ posee un valor c_{ij} que representa la distancia entre la ciudad i y la ciudad j. El problema consiste en encontrar el camino más corto que visite todas las ciudades exactamente una vez y vuelva a la ciudad de origen. Este camino se denomina camino Hamiltoniano, y por ende el TSP puede ser definido como encontrar el camino Hamiltoniano más corto. Para los TSP simétricos, la distancia $c_{ij}=c_{ji}$. La definición detallada del problema puede encontrarse en [Gómez2004] y pertenece al conjunto de problemas NP-completos [Sahni1976].

En el caso bi-objetivo, se asocia a cada arco (i,j) un par de valores, uno para cada objetivo, que representan las distancias entre las ciudades i y j de acuerdo a cada objetivo. El problema consiste en minimizar:

$$\vec{f}(\vec{x})=\begin{cases}\sum_{i=1}^{N} c^{1}_{\phi_i \phi_{i+1}} \\ \sum_{i=1}^{N} c^{2}_{\phi_i \phi_{i+1}}\end{cases} \tag{3.1}$$

donde minimizar se refiere a la optimalidad Pareto, y ϕ_i representa la i-ésima ciudad visitada. De manera a agregar el costo de retorno a la ciudad de origen, se considera a ésta como la siguiente, luego de la N-ésima ciudad visitada.

El *bi-objective TSP* posee aplicaciones prácticas en la optimización de rutas considerando, por ejemplo, tiempo de viaje y distancia recorrida.

Aplicaciones Prácticas

Para enfatizar la importancia de este paradigmático problema, a continuación se presentan algunas aplicaciones prácticas.

3.1.1. Programa espacial *Starlight* de la NASA

El propósito del programa espacial es mostrar imágenes de cuerpos celestiales mediante un par de satélites involucrados en la misión [Bailey2000]. El objetivo es minimizar el consumo de combustible utilizado por ambos satélites en su posicionamiento y la obtención de imágenes de los cuerpos celestiales.

En este caso las ciudades del TSP son las posiciones de los dos satélites para obtener información de los cuerpos celestiales, y el costo de viaje está representado por la cantidad de combustible requerida para volver a posicionar a ambos satélites de un punto adecuado para un cuerpo celeste, a otro apropiado para otro cuerpo celeste.

3.1.2. Optimización de rutas aéreas

Este problema consiste en determinar la ruta aérea de menor distancia que recorra todos los aeropuertos que un avión debe visitar. Es un ejemplo clásico del TSP en donde los aeropuertos están ubicados en distintos puntos geográficos y existe una distancia asociada a cada par de aeropuertos.

La importancia de la optimización de las rutas aéreas es la disminución de los costos de viaje, como el ahorro de combustible y el tiempo total de viaje que redunda en la satisfacción del cliente.

3.1.3. Determinación de una topología de anillo

El diseño de una red en topología de anillo requiere claramente pasar por cada punto de acceso a la red exactamente una vez y volver al punto de partida. En el caso práctico, el laboratorio de investigación de la compañía *Bell Communications* diseñó en una ocasión una herramienta basada en el TSP para encontrar la topología virtual en anillo de menor longitud para una red mallada de fibras ópticas. Es fácil notar la importancia de minimizar la longitud del anillo de manera a disminuir el costo total de la red. Las ciudades son representadas por los puntos de acceso a la red que se desean conectar y el costo de viaje viene dado por las distancias entre cada punto de acceso.

3.1.4. Otros

Encontrar el camino de menor longitud para un repartidor de diarios que debe entregar el diario a *n* clientes. En este caso las ciudades son los clientes y el costo de viaje está determinado por la distancia entre cada cliente y/o el tiempo de traslado.

Otra aplicación similar es la recolección de monedas de los teléfonos públicos. En este caso los teléfonos públicos representan las ciudades y el costo de viaje está dado por la distancia y/o entre cada teléfono. Se desea determinar la ruta de menor longitud y/o tiempo que visite todos los teléfonos exactamente una vez.

La recolección de basura y la ruta de transporte escolar son otros ejemplos de aplicaciones reales del TSP.

3.2. Quadratic Assignment Problem (QAP)

El QAP o problema de asignación cuadrática es un problema NP-completo [Sahni1976], y consiste en asignar un conjunto de instalaciones a un conjunto de localidades con distancias conocidas entre cada par de localidades, dados los flujos entre cada par de instalaciones, de manera a minimizar la suma del producto entre los flujos y las distancias [Çela1998].

El QAP multi-objetivo (mQAP), propuesto por Knowles y Corne [Knowles2003] utiliza diferentes matrices de flujo, y mantiene la misma matriz de distancia. Dadas *n* instalaciones y *n* localidades, una matriz $A=\{a_{ij}\}\in\mathbb{R}^{n\times n}$ donde a_{ij} representa la distancia entre las localidades *i* y *j*, y *b* matrices $B^q=\{b_{ij}^q\}\in\mathbb{R}^{n\times n}$, $q=1,\ldots,b$ donde b_{ij}^q representa el *q*-ésimo flujo entre las instalaciones ubicadas en las localidades *i* y *j,* el mQAP se define como:

$$Minimizar\ \vec{f}(\vec{x})=\begin{cases}\sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}b_{ij}^{1}\\ \vdots\\ \sum_{i=1}^{n}\sum_{j=1}^{n}a_{ij}b_{ij}^{b}\end{cases} \tag{3.2}$$

donde minimizar se refiere a la optimalidad Pareto. La aplicabilidad real de este problema aparece en la ubicación de hospitales e instituciones sociales. Para este trabajo se consideró el bQAP (QAP bi-objetivo).

Aplicaciones Prácticas

A continuación se presentan algunas aplicaciones prácticas del QAP.

3.2.1. Ubicación de edificios en hospitales y facultades dentro de una ciudad universitaria

Instituciones de gran tamaño como hospitales y universidades poseen generalmente varios edificios, y normalmente existen diferentes flujos de movimiento de personas entre cada par de edificios. Por este motivo se busca ubicar físicamente los mismos de tal forma a minimizar las distancias entre aquellos pares de edificios con elevado flujo de movimiento de personas. Esta es una aplicación directa del QAP.

3.2.2. Diseño del teclado para escritura

El teclado de un computador o máquina de escribir posee muchas posiciones posibles y muchas teclas distintas a ubicar. En este caso, las localidades del QAP son las posiciones físicas posibles para una tecla dentro del teclado y las instalaciones están representadas por las teclas distintas a colocar. Normalmente se conoce el flujo de utilización de cada par de teclas posibles para un idioma dado. El objetivo es encontrar la distribución de teclas que optimice la distancia entre los pares de teclas frecuentemente utilizados. De esta forma se busca minimizar el movimiento de los dedos.

3.2.3. Balanceo de turbinas

Este problema se presenta en [Mason1997]. Un motor con turbina, sea del tipo hidráulico, a gas o a vapor, está compuesto por hélices que giran alrededor del eje central de la turbina, ver figura 3.1. Cada hélice está compuesta por aspas colocadas en la periferia de un disco en puntos equidistantes. Las masas de estas aspas generalmente no son idénticas debido a errores en el proceso de fabricación o por desgaste de utilización. Por este motivo el centro de masas de la hélice depende de la ubicación de estas aspas en su periferia. Cuando el centro de gravedad de la hélice no coincide con el eje central de rotación se produce desbalanceo estático, denominado desbalanceo de la turbina. Esto no es deseable ya que produce vibraciones y reduce el tiempo de vida útil de la turbina. Normalmente la ubicación óptima de las aspas se aproxima mediante métodos de prueba y error o mediante la intuición de una persona experta pudiendo tomar días o semanas dependiendo del tipo de turbina.

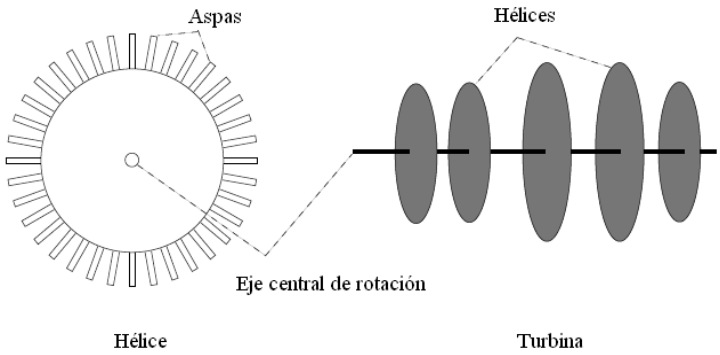

Figura 3.1. Representación gráfica de una turbina, ver [Mason1997]

Cada posición posible para un aspa está representada por un vector de posición d_i, y cada aspa posee una masa o peso w_i. El momento generado al colocar un aspa de peso w_i en la posición *i* está dado por $M_i = w_i \cdot d_i$. El momento de la hélice que contiene a todas las aspas está representado por la suma de los momentos individuales de cada aspa. El problema consiste en asignar las aspas a posiciones de manera a minimizar la suma de los momentos individuales de cada aspa. Una magnitud elevada para el momento de la hélice representa desbalanceo estático [Mason1997]. En este caso el problema puede plantearse como un QAP donde las aspas representan las instalaciones y se conoce la distancia entre cada posición posible para cada aspa, y el objetivo es minimizar la suma de los momentos de cada aspa.

3.3. Vehicle Routing Problem with Time Windows (VRPTW)

El VRPTW (*Vehicle Routing Problem with Time Windows*), cuya definición puede encontrarse en [Barán2003], busca encontrar rutas de costo mínimo para una flota de vehículos, que se originan y terminan en un depósito central, minimizando además el número de vehículos a utilizar. Cada cliente debe ser visitado exactamente una vez y posee una demanda que debe ser cubierta por los vehículos sin exceder la capacidad de los mismos. Además, cada cliente posee una ventana de tiempo que representa el tiempo de recepción permitido que debe ser respetado por los camiones. En cada cliente se tiene en cuenta un tiempo de servicio durante el cual el vehículo permanece en el mismo. Matemáticamente el problema bi-objetivo consiste en minimizar el vector:

$$\vec{f}(\vec{x}) = \begin{cases} v \\ \sum_{i=1}^{v} \sum_{j=1}^{|\psi_i|} t_{j,j+1} \end{cases} \tag{3.3}$$

sujeto a:

$$\sum_{j=1}^{|\psi_i|} q_j \leq Q \qquad \forall i \in \{1,2,...,v\}$$

$$b_k \leq \sum_{j=1}^{k-1} (t_{j,j+1} + s_j) \leq e_k \quad \forall k \in \{1,2,...,|\psi_i|\}$$

donde:

v	representa la cantidad de vehículos necesarios;
n	es la cantidad de clientes; esto es, $n = \sum_{i=1}^{v} \|\psi_i\|$;
$C = \{c_0, c_1, ..., c_n\}$	representa el conjunto de *n* clientes y el depósito c_0 , donde c_j representa al cliente *j*;
ψ_i	representa el conjunto de clientes visitados por el *i*-ésimo vehículo;
$t_{j,j+1}$	es el tiempo de viaje entre un cliente c_j y el siguiente cliente c_{j+1} , visitado por un mismo vehículo;
Q	la capacidad máxima de un vehículo, que se asume constante para todos los vehículos;
q_j	demanda del cliente c_j , con $q_j \leq Q$ y $q_0 = 0$;
$[b_j, e_j]$	ventana de tiempo aceptable del cliente *j*, donde b_j es el instante más temprano de servicio del cliente c_j y e_j es el instante más tarde;
s_j	representa el tiempo de servicio en el cliente c_j .

Aplicaciones Prácticas

Encontrar la manera más eficiente de distribuir mercaderías en una red logística es un objetivo primordial en una cadena de proveedores. Incluso si la mejora en las rutas de camiones es mínima, en el sentido económico resulta importante, ya que la distribución de mercaderías normalmente se realiza constantemente, inclusive diariamente, con lo cual las ganancias obtenidas mediante el ahorro son acumuladas en el tiempo. En consecuencia, se presenta a continuación un ejemplo de distribución de mercaderías.

Optimización de rutas en una cadena de supermercados

La optimización de rutas en una cadena de supermercados representa una aplicación directa del problema de ruteo de vehículos con ventanas de tiempo. En la realidad, una cadena de supermercado normalmente tiene la necesidad de distribuir sus mercaderías. Para el efecto, generalmente cuenta con una flota de vehículos y distintos puntos de distribución que normalmente representan supermercados pertenecientes a la cadena. Estos puntos de distribución establecen un horario de recepción de mercaderías, que representa la ventana de tiempo del cliente.

Es de interés para la cadena de supermercados minimizar el tiempo total de viaje y/o la distancia total recorrida de los camiones de manera a obtener ahorros en el coste logístico de la distribución de mercaderías. Así mismo se desea minimizar el tamaño de la flota de camiones, que lógicamente representa un significativo ahorro en costos de adquisición y mantenimiento de vehículos. Además, se debe cumplir con las ventanas de tiempo impuestas por el cliente, de forma a organizar el horario del personal y no incurrir en gastos extras. En la tabla 3.1 se presentan algunas otras aplicaciones reales del VRPTW según distintas áreas económicas. Puede notarse mediante la tabla 3.1 que éste problema es directamente aplicable a una amplia gama de situaciones en la vida real.

Tabla 3.1. Aplicaciones reales del VRPTW para distintas áreas económicas

Área económica	**Aplicación Real**
Industria del automóvil	Entrega de piezas de repuesto
Prensa	Distribución de periódicos y revistas
Banca	Reparto y recolección de dinero en efectivo
Sector público	Reparto de correo, Limpieza de calles
Agricultura	Recolección de cereales, leche
Educación	Rutas de transportes escolares
Defensa	Rutas de aviones espías, logística militar
Salud	Distribución de medicamentos a farmacias
Industria	Distribución de mercaderías entre almacenes
Servicios	Reparación de electrodomésticos a domicilio
Transporte	Planificación de rutas de aviones, trenes, camiones.

Capítulo 4

Optimización Basada en Colonia de Hormigas

Los algoritmos basados en colonias de hormigas son sistemas multi-agentes donde el comportamiento de cada hormiga particular, denominada hormiga artificial, está inspirado en el comportamiento de las hormigas reales. Estos algoritmos son ejemplos satisfactorios de los sistemas basados en la Inteligencia en Enjambre (*Swarm Intelligent Systems*), como se presenta en el trabajo de Tarasewich *et al*. [Tarasewich2002].

Recientemente, la comunidad científica comenzó a analizar el comportamiento de insectos en busca de ideas que puedan ser utilizadas como meta-heurísticas. Muchos aspectos de las actividades colectivas de los insectos sociales, como las hormigas, son auto-organizadas (*self-organizing*), lo que significa que un comportamiento grupal complejo emerge de la interacción de individuos que poseen un comportamiento muy simple. El ejemplo de esta situación, en el caso de los insectos, es la búsqueda de alimentos y la construcción de nidos.

Los resultados de esta auto-organización son globales (se deben a la colonia) por naturaleza, y se fundamentan en interacciones basadas enteramente en información local. Estas interacciones pueden ser directas (vía contacto físico, visual o químico) o indirectas. El contacto indirecto se caracteriza por un concepto que será explicado más adelante "*stigmergy*", que básicamente se refiere a que un individuo realiza una acción basado en las acciones realizadas anteriormente por otros individuos.

La expresión “inteligencia en enjambre” (*swarm intelligence*) fue utilizado para describir este comportamiento auto-organizativo en el contexto de los sistemas robóticos, y fue extendido para incluir el diseño de algoritmos inspirados en el comportamiento general de las colonias de insectos y otros animales sociales. Así, se incluyen en esta categoría a las hormigas, escarabajos, termitas y abejas, entre otros.

En este capítulo se presenta la teoría denominada meta-heurística ACO (*Ant Colony Optimization*), que define una clase particular de algoritmos ACO, además de describir varias implementaciones de este tipo de algoritmos basados en las colonias de hormigas.

4.1. Colonias de Hormigas

Las hormigas son insectos sociales, cuyo comportamiento está orientado a la supervivencia de la colonia como un todo antes que de un único individuo. Basados en los experimentos realizados por Denenbourg *et al.* [Denenbourg1990] y Goss *et al.* [Goss1989] se demostró su capacidad de poder encontrar el camino más corto desde su nido a las fuentes de comida. Este comportamiento se obtiene por medio de una sustancia química llamada feromona, que las hormigas depositan al caminar en busca de alguna fuente de comida, dejando de esta manera rastros de feromonas que las demás hormigas pueden oler.

Estos rastros de feromonas sirven a las hormigas para volver a su nido, y también para seguir el camino a una fuente de alimento encontrado por otra hormiga. De esta manera una hormiga que debe escoger entre más de un camino a seguir, escogerá con mayor probabilidad el camino que posea más rastros de feromonas.

Así, las hormigas que siguieron inicialmente el camino más corto a una fuente de alimento llegarán más rápido, y como además fueron depositando feromonas, entonces al empezar su retorno al nido los rastros de feromonas de ese camino serán mayores a los demás, incrementando la probabilidad de una hormiga de escoger este camino más corto. Luego de un tiempo, casi la totalidad de las hormigas utilizarán el camino más corto, que poseerá más rastros de feromonas, para llegar a la fuente de alimento. En la figura 4.1 se muestra gráficamente el proceso natural de búsqueda de las hormigas reales considerando el caso de contar con caminos de distintas longitudes.

Las feromonas representan para las hormigas una forma de comunicación indirecta denominada “*stigmergy”*, que se caracteriza por: i) el insecto comunica su información modificando el estado del ambiente físico visitado, y ii) la información revelada es de naturaleza local y puede ser

accedida solo por otros insectos que visitan el estado donde fue depositada. Utilizando como base este modelo de comportamiento, se desarrolló la teoría de la meta-heurística ACO, donde se utiliza una colonia de hormigas artificiales.

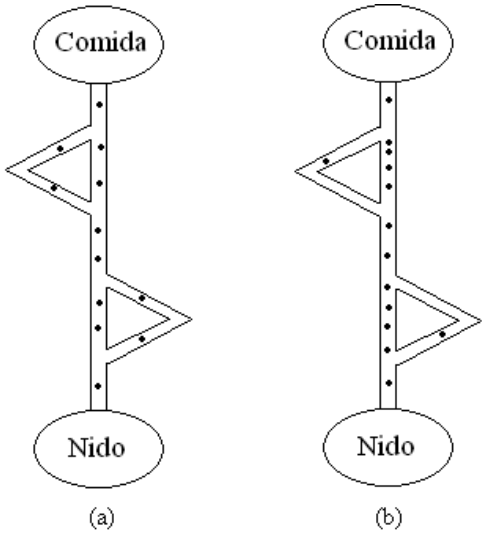

Figura 4.1. Comportamiento natural de las hormigas. (a) En un instante t_0 inicial existe una distribución uniforme de las hormigas en los distintos caminos. (b) Estado de distribución de las hormigas para un instante $t > t_0$, el camino más corto posee más feromonas y por lo tanto es utilizado por más hormigas.

En la siguiente sección presentamos una versión simple de un algoritmo ACO, que modela el comportamiento real de las hormigas en su objetivo de converger al camino más corto entre dos puntos, generalmente el nido y alguna fuente de alimento.

4.2. Modelo computacional simple de las hormigas reales

Presentamos en esta sección un algoritmo ACO simple [Dorigo1999] que ilustra el mecanismo natural explicado anteriormente acerca del comportamiento de las hormigas. Para el ejemplo, el algoritmo está enfocado en buscar el camino de costo mínimo entre un par de nodos, dado un grafo completamente conexo[1], y que representa el problema.

En este caso, la principal tarea de las hormigas artificiales, al igual que las hormigas naturales, es encontrar el camino más corto entre un par de nodos del grafo. Consideramos como longitud de un *tour* (o solución) a la cantidad de saltos entre nodos realizado por la hormiga. En este caso, tenemos un solo objetivo, minimizar la longitud del camino que conecta el par de nodos origen y destino,

1 Significa que existe una arista que interconecta a cada par de nodos.

entonces estamos tratando con una optimización mono-objetiva. De igual manera, estos conceptos en el contexto mono-objetivo, serán referenciados más adelante en este capítulo al realizar las extensiones apropiadas para el caso multi-objetivo.

A cada arco (*i,j*) del grafo se asocia una variable τ_{ij} denominada rastro artificial de feromonas o rastro de feromonas. Estas variables son consultadas y actualizadas por las hormigas. La cantidad de feromonas es proporcional a la cantidad de hormigas que ya utilizaron el referido arco.

Cada hormiga aplica una regla de decisión paso a paso (de salto en salto) al construir una solución. En cada nodo se utiliza información local de manera estocástica para decidir el siguiente nodo a visitar. Se utiliza el rastro de feromonas asociado a cada arco de salida del nodo actual en que se encuentra una hormiga.

Formalmente, la regla de decisión de la hormiga *k* ubicada en el nodo *i* utiliza los rastros de feromonas τ_{ij} para computar la probabilidad de escoger el nodo $j \in J_i$ como el siguiente nodo a visitar, donde J_i es el conjunto de vecinos ubicados a un salto del nodo *i*. La regla viene dada por la expresión:

$$p_{i,j} = \begin{cases} \dfrac{\tau_{i,j}}{\sum_{x \in J_i} \tau_{i,x}} & si\ j \in J_i \\ 0 & si\ j \notin J_i \end{cases} \qquad (4.1)$$

Al construir una solución, las hormigas depositan feromonas en los arcos que utilizan. Entonces, cada hormiga deposita una cantidad $\Delta\tau$ de feromonas al pasar de una ciudad *i* a otra *j*. De esta forma, la hormiga actualiza el valor τ_{ij} según la expresión:

$$\tau_{ij} = \tau_{ij} + \Delta\tau \qquad (4.2)$$

Con esta regla, se estimula a las siguientes hormigas a utilizar el arco (*i,j*) recientemente visitado por la hormiga que realizó la actualización. Como en el caso de las hormigas reales, se pretende mediante autocatálisis y la existencia de caminos de diferentes longitudes, converger al camino más corto, o sea, encontrar la solución "óptima". Para $\Delta\tau$ se puede utilizar, por ejemplo, un valor constante definido a priori.

Para evitar convergencias a óptimos locales, se agrega un mecanismo de evaporación de feromonas, al igual que en el caso real. Las intensidades (valores) de los rastros de feromonas

disminuyen paulatinamente, favoreciendo la exploración de arcos distintos durante el proceso de búsqueda. Esta evaporación se realiza de una forma simple, disminuyendo el valor de cada τ_{ij} en cada iteración del algoritmo según:

$$\tau_{ij} = (1-\rho)\cdot\tau \quad , \quad \rho \in (0,1)$$

donde ρ representa el coeficiente de evaporación de feromonas y se define a priori.

El algoritmo presentado en esta sección es una versión simple que modela el comportamiento de las hormigas reales en su tarea de búsqueda de alimentos. Un pseudo-código del mismo se presenta en la figura 4.2.

A pesar de su simplicidad (y quizás por este motivo), este modelo posee limitaciones, y puede ser extendido de manera a mejorar su desempeño. Por ejemplo, podemos depositar una cantidad de feromonas proporcional a la evaluación de la calidad de la solución encontrada, favoreciendo así a los arcos que componen "buenas" soluciones. En los problemas reales, generalmente se conoce alguna información heurística particular del problema, que podemos tener en cuenta en nuestro modelo de hormigas artificiales.

```
procedure ACO_SIMPLE
inicializar_parametros()
while not condicion_parada()
    generacion=generacion + 1
    for ant=1 to m // m es la cantidad de hormigas
       construir_solucion(origen,destino)
       evaluar_solucion()
       calcular_mejor_solucion() // según su evaluación
    end for
end while
print mejor_solucion
end

procedure construir_solucion (origen,destino)
sol={∅}
estado_actual=origen
while not estado_actual = destino
  siguiente=seleccionar_siguiente_estado() // según (4.1)
  sol=sol ∪ {siguiente}
  actualizar_feromonas_paso_a_paso(estado_actual,
                          siguiente)       // según (4.2)
  estado_actual=siguiente
end while
end
```

Figura 4.2. Pseudo-código de un Algoritmo ACO Simple

Es importante mencionar que en problemas reales normalmente se tienen restricciones. Como ejemplo de restricción aplicable a nuestro modelo simple, podemos restringirnos a encontrar

soluciones que no posean ciclos, con lo cual debemos agregar a nuestras hormigas artificiales alguna forma de memoria interna, donde puedan almacenar situaciones anteriores.

4.3. Meta-heurística ACO

La teoría de la meta-heurística ACO mencionada en esta sección fue presentada por Dorigo y Di Caro [Dorigo1999]. También aparece en el trabajo de Maniezzo *et al.* [Maniezzo2004]. En los trabajos de Dorigo *et al.* [Dorigo1996] y Dorigo y Di Caro [Dorigo1999] fueron mencionadas ideas iniciales que luego constituyeron la teoría de la meta-heurística ACO.

Como ya fuera mencionado, un algoritmo ACO constituye un conjunto de agentes computacionales concurrentes (la colonia de hormigas artificiales), que se mueven a través de estados del problema, correspondientes a soluciones parciales del mismo, buscando las mejores soluciones factibles.

Las hormigas artificiales se mueven aplicando una política estocástica de decisión local basada en dos parámetros, denominados rastros de feromonas τ y visibilidad (o atractividad) η . Los rastros de feromonas reflejan que los estados más visitados son altamente deseables, implementando de esta manera el proceso de auto-catálisis, mencionado anteriormente, mientras que la visibilidad refleja que estados menos costosos (evaluados según los objetivos a optimizar) son también deseables. De esta forma, agregamos información específica del problema por medio de la visibilidad a la política de decisión estocástica de las hormigas.

La visibilidad η_{ij} en el arco (*i,j*), en un contexto de minimización generalmente se calcula como $\eta_{ij}=1/c_{ij}$, donde c_{ij} representa el costo de utilizar el arco (*i,j*) de acuerdo a un objetivo particular. Así los arcos menos costosos y por ende más deseables poseen un valor más elevado de visibilidad.

Generalmente los elementos de la matriz de feromonas son inicializados a un valor τ_0 definido a priori. La política de transición [Dorigo1999] se basa en asignar la probabilidad de ir del estado *i* al estado *j* según la ecuación:

$$p_{i,j}=\begin{cases}\dfrac{\tau_{i,j}^{\alpha}\eta_{i,j}^{\beta}}{\sum_{x\in J_i}\tau_{i,x}^{\alpha}\eta_{i,x}^{\beta}} & si\ j\in J_i \\ 0 & en\ caso\ contrario\end{cases} \tag{4.3}$$

donde J_i representa el vecindario de estados alcanzables desde el estado *i* y válidos atendiendo las restricciones del problema, α y β son parámetros definidos a priori que reflejan la importancia relativa de las feromonas y la visibilidad respectivamente. El algoritmo ACO simple, mencionado en la sección anterior, puede ser considerado un caso especial en el cual $\beta = 0$, o sea sin utilizar la visibilidad.

Al completar una solución o durante la construcción de la misma, la hormiga evalúa la solución y modifica los rastros de feromonas en las componentes de la matriz de feromonas $\tau = \{\tau_{ij}\}$ que de esta forma guarda el conocimiento de las áreas ya exploradas. Esta información de feromonas guiará la búsqueda de futuras hormigas. El algoritmo puede incluir un proceso de evaporación de rastros de feromonas, y otras acciones como realizar optimizaciones locales sobre soluciones encontradas o actualizar la información global para guiar el proceso de búsqueda desde una perspectiva no local. La actualización y evaporación de feromonas se realiza según la ecuación generalizada:

$$\tau_{i,j} = (1-\rho)\cdot\tau_{i,j} + \rho\cdot\Delta\tau \tag{4.4}$$

donde ρ representa el coeficiente de evaporación, y $\Delta\tau$, en un contexto de minimización, se calcula como:

$$\Delta\tau = \frac{1}{f_k(\vec{x})} \tag{4.5}$$

con $k \in \{1,2,\ldots,b\}$ en el caso de realizar una actualización basada en un solo objetivo. Para el caso de actualización considerando los *b* objetivos al mismo tiempo [Pinto2005], se puede utilizar, por ejemplo, la ecuación:

$$\Delta\tau = \frac{1}{\sum_{k=1}^{b} f_k(\vec{x})} \tag{4.6}$$

donde por motivos de normalización los valores f_k son divididos por un valor máximo calculado a priori. Así, la cantidad de feromonas depositada en los arcos componentes de una solución es proporcional a la evaluación de la solución según los objetivos del problema. Puede notarse que la ecuación 4.4 agrupa la actualización y evaporación de feromonas mencionadas en el modelo simple de la sección anterior.

Corresponde mencionar que en la definición de las ecuaciones presentadas anteriormente, asumimos un contexto de minimización. Entonces, atendiendo las ecuaciones 4.5 y 4.6, las soluciones con “mejores evaluaciones” presentarán valores más pequeños en las funciones objetivo,

y como la cantidad de feromonas a depositar se calcula en ambas ecuaciones como la inversa de la evaluación correspondiente, la cantidad aportada será mayor. Se obtiene así el comportamiento deseado, en el cual soluciones mejor evaluadas aumentan en mayor proporción los rastros de feromonas en los arcos utilizados.

En la figura 4.3 se muestra el pseudo-código de un algoritmo ACO genérico.

```
procedure ACO
inicializar_parametros()
while not condicion_parada()
      generacion=generacion + 1
      for ant=1 to m       // m es la cantidad de hormigas
            construir_solucion()
            evaluar_solucion()     //según objetivo a optimizar
            actualizar_feromonas() //según ecuación (4.4)
            calcular_mejor_solucion()
      end for
end while
print mejor_solucion
end

procedure construir_solucion
sol={∅}
while existen_estados_no_visitados()
      siguiente=seleccionar_siguiente_estado() // según (4.3)
      sol=sol ∪ {siguiente}
      marcar_como_visitado(siguiente)
      if(actualizacion_paso_a_paso)
            actualizar_feromonas_paso_a_paso()// según (4.4)
end while
end
```

Figura 4.3. Pseudo-código de un algoritmo ACO genérico [Dorigo1999].

4.4. Alternativas de diseño multi-objetivo

Las principales alternativas para implementaciones de un algoritmo ACO multi-objetivo se extienden de los algoritmos ACO tradicionales y se refieren principalmente al número de colonias y a la forma de manejar la información de feromonas y las funciones heurísticas [López-Ibañez2004]. Las combinaciones de estas características son las utilizadas por las implementaciones actuales de los algoritmos ACO multi-objetivos y por este motivo merecen ser mencionadas.

Cabe mencionar que en un contexto multi-objetivo, la hormiga que completó una solución debe actualizar el conjunto Pareto *CP* si la solución encontrada es no dominada con respecto a las

existentes en *CP* y luego debe eliminar las soluciones dominadas de *CP*, a diferencia del contexto mono-objetivo, donde simplemente se actualiza una única mejor solución cada vez que se encuentra una solución mejor a la conocida hasta ese momento.

4.4.1. Número de Colonias

La alternativa es contar con múltiples colonias, donde el conjunto de todas las hormigas es dividido en conjuntos disjuntos de menor tamaño, y cada grupo conforma una colonia. Cada colonia se especializa en una región del frente Pareto (por ejemplo, un objetivo particular), y utiliza su propia información de feromonas. En un enfoque cooperativo, pueden intercambiarse soluciones entre colonias de manera que la actualización de feromonas en una colonia sea afectada también por soluciones de otras colonias.

4.4.2. Información de feromonas y visibilidad

Existen dos alternativas, la primera utiliza una única tabla de feromonas con la información relacionada a todos los objetivos. La segunda alternativa, utiliza múltiples tablas de feromonas, donde cada tabla posee la información de feromonas con respecto a un objetivo del problema. La regla de transición puede basarse en un solo objetivo, o puede basarse en la agregación de las diferentes tablas de feromonas asignando un peso a cada objetivo. La información heurística o visibilidad del problema posee análogas implementaciones alternativas a la información de feromonas.

Completando nuestra implementación de un algoritmo ACO genérico, lo extendemos de manera a tratar múltiples objetivos, agregando el manejo de un conjunto soluciones Pareto óptimas en vez de una única mejor solución. El pseudo-código se presenta en la figura 4.4., y en adelante denominamos *MOACO (MultiObjective Ant Colony Optimization)* a un algoritmo ACO multi-objetivo.

Otros trabajos que presentan varios algoritmos ACO multi-objetivos pueden encontrarse en [García-Martínez2004], mientras que en [López-Ibañez2004] se presentan varios diseños alternativos.

En resumen, partimos del algoritmo ACO simple, y desarrollamos algoritmos ACO con mayores capacidades y menores limitaciones que la versión simple. Agregamos las mejoras mencionadas anteriormente como la actualización basada en la calidad de las soluciones encontradas, permitir

establecer restricciones de factibilidad a las soluciones, y la transición de estados está basada tanto en los conocimientos aprendidos previamente (rastros de feromonas) como en información heurística específica del problema. Finalmente extendimos las ecuaciones y los algoritmos de manera a tratar problemas multi-objetivos.

```
procedure MOACO
inicializar_parametros()
while not condicion_parada()
      generacion=generacion + 1
      for ant=1 to m       // m es la cantidad de hormigas
            construir_solucion()
            evaluar_solucion()      //según la función (2.3)
            actualizar_feromonas() //según ecuación (4.4)
            actualizar_conjunto_pareto()
      end for
end while
print ConjuntoPareto
end

procedure construir_solucion
sol={∅}
while existen_estados_no_visitados()
      siguiente=seleccionar_siguiente_estado() // según (4.3)
      sol=sol ∪ {siguiente}
      marcar_como_visitado(siguiente)
      if(actualizacion_paso_a_paso)
            actualizar_feromonas_paso_a_paso()// según (4.4)
end while
end
```

Figura 4.4. Pseudo-código de un algoritmo ACO multi-objetivo.

En el siguiente capítulo se presentan distintas implementaciones de algoritmos ACO en el contexto de optimización mono-objetivo y multi-objetivo, basados en gran parte a los algoritmos genéricos presentados en este capítulo.

Capítulo 5

Algoritmos ACO

En capítulos anteriores se presentó la teoría de la meta-heurística ACO que modela el comportamiento de las hormigas reales en el contexto computacional. Hemos introducido algunos pseudo-códigos genéricos a partir de los cuales podemos derivar diversos algoritmos ACO mono-objetivos y multi-objetivos propuestos hasta el momento de la elaboración de este trabajo, y que por ende constituyen hasta la fecha el estado del arte en la optimización basada en colonias de hormigas.

También en este capítulo se propone un nuevo algoritmo MOACO, el *Multiobjective Ant System (MAS*), que será explicado en detalle en las siguientes secciones. Este capítulo presenta el siguiente orden temático, primeramente se describen los algoritmos ACO mono-objetivos, y a continuación se describen los algoritmos MOACO que son básicamente algoritmos basados en algún algoritmo ACO mono-objetivo particular y con las adaptaciones correspondientes para volverlo multi-objetivo.

Además, cada implementación particular incorpora algunos mecanismos específicos para tratar el problema de la convergencia a óptimos locales, la exploración de distintas regiones del espacio de soluciones del problema a resolver, entre otros que serán explicados a continuación. En todos los casos presentados en este capítulo, se asume nuevamente un contexto de minimización, sin que esto se constituya en pérdida de generalidad.

5.1. Algoritmos ACO mono-objetivos

5.1.1 Ant System (AS)

Este algoritmo se basa en el algoritmo ACO genérico de la Figura 4.3 presentado en el capítulo anterior, y fue propuesto en [Dorigo1996]. Define como un ciclo de computación, los pasos que van desde el momento en que cada hormiga empieza la construcción de su solución hasta que todas las hormigas completan una solución. Si todas las hormigas generan las mismas soluciones en un ciclo, se dice que el algoritmo entró en un estado de estancamiento *(stagnation behavior),* entonces, el AS define el criterio de parada del algoritmo como un número máximo de ciclos (generaciones) o un estado de estancamiento.

La probabilidad p_{ij} de la hormiga k, que se encuentra en el estado i, de pasar al estado j viene dada por la ecuación 4.3, tomada del algoritmo ACO genérico. Realiza una actualización de feromonas al finalizar cada ciclo, según la ecuación 4.4 tomada también del algoritmo ACO genérico. El valor para $\Delta\tau$ se calcula como:

$$\Delta\tau = \frac{Z}{f(\vec{x})} \tag{5.1}$$

donde Z es un parámetro definido a priori. Esta ecuación es similar a la ecuación genérica 4.5 ponderada por el factor Z mencionado. De esta forma, mediante el parámetro Z se controla la cantidad de feromonas a depositar, actuando como un factor de normalización.

Este algoritmo fue uno de los primeros trabajos en la optimización basada en colonia de hormigas, y básicamente es análogo al algoritmo ACO genérico presentado anteriormente. Podemos decir entonces que aporta las ideas básicas para los demás algoritmos de hormigas y su importancia es más bien histórica que práctica.

5.1.2 Rank-based Ant System (ASrank)

Este algoritmo es una extensión del Ant System aplicando el concepto de "*ranking*" (ordenamiento) [Bullnheimer1997]. Luego de que las m hormigas generan sus respectivas soluciones, éstas se ordenan de acuerdo a la evaluación de sus soluciones. La diferencia con el AS aparece al actualizar las feromonas, donde la cantidad de feromonas se pondera por la posición μ de la hormiga, y solo las l mejores hormigas, denominadas elitistas, son consideradas. De esta manera,

se evita que las hormigas con soluciones sub-óptimas incrementen el nivel de feromonas de ciertos estados.

La actualización de feromonas se define según la ecuación 4.4. La cantidad de feromonas a depositar se basa en la ecuación 5.1. con la restricción de permitir solo a las *l* mejores hormigas, con las mejores *l* soluciones de la iteración, actualizar los rastros de feromonas en los arcos que utilizaron. Está adaptación se muestra en la siguiente ecuación:

$$\Delta\tau=(l-\mu)\cdot\frac{Z}{f(\vec{x})} \tag{5.2}$$

Z es el mismo factor de normalización utilizado en el AS. Además, este algoritmo introduce la actualización elitista de feromonas, que se realiza luego de cada iteración (ciclo) del algoritmo, y consiste en incrementar los rastros de feromonas en los arcos correspondientes a la mejor solución encontrada hasta el momento (solución elitista) en una cantidad constante $\Delta\tau'$, definida como:

$$\Delta\tau'=l\cdot\frac{Z}{f(\vec{x})} \tag{5.3}$$

Este algoritmo utiliza el concepto de elitismo, en el cual solo las mejores soluciones depositan rastros de feromonas, también se realiza una actualización basada en la mejor solución encontrada hasta el momento. A esta solución la denominaremos en adelante *global best* (denotado por *gb*), y a la mejor solución de la iteración la llamaremos *iteration best (*denotado por *ib*). Mediante esta estrategia elitista se busca concentrar la búsqueda en la cercanía de las mejores soluciones encontradas, donde potencialmente se encuentran buenas soluciones [Gómez2004a].

5.1.3 Max-Min Ant System (MMAS)

El MMAS se basa también en el AS, pero presenta varias diferencias. Este algoritmo fue propuesto en [Stutzle2000]. La probabilidad de transición de estado es la misma del AS (ecuación genérica 4.3). Utiliza el concepto de elitismo, y permite solo a la mejor solución encontrada en la iteración o hasta el momento, actualizar los rastros de feromonas en sus arcos componentes, utilizando la misma regla que el AS. El uso de esta forma de actualización de feromonas favorece la explotación de buenos caminos encontrados en la búsqueda (concepto conocido como *exploitation*).

Al utilizar la solución global *gb* para actualizar las feromonas, la búsqueda se concentra rápidamente en esta solución y se limita la exploración de posibles mejores soluciones. Utilizando

la solución de la iteración *ib*, considerando que las mejores soluciones por iteración difieren generalmente unas de otras, un gran número de soluciones recibirán feromonas, posibilitando la exploración de varios caminos. Se pueden mezclar las estrategias, utilizando, por ejemplo, *ib* por defecto y *gb* después de un número fijo de iteraciones en la actualización de las feromonas. La estrategia sugerida en [Stutzle2000] es utilizar una estrategia mixta dinámica, que incremente la frecuencia de uso de *gb* a medida que corren las iteraciones.

Inicialmente se favorece la exploración de distintos caminos, y a medida que avanzan las iteraciones y se van encontrando mejores soluciones, se favorece más a la mejor solución encontrada hasta el momento, implementando el concepto de *exploitation*.

Para abordar el problema del estancamiento (*stagnation behavior*) se propuso utilizar límites en los niveles de feromonas que un estado puede tomar, o sea:

$$\tau_{min} \leq \tau_{ij} \leq \tau_{max} \tag{5.4}$$

Si $\tau_{ij} > \tau_{max}$, entonces $\tau_{ij} = \tau_{max}$, con $\tau_{max} = \Delta\tau/(1-\rho)$, y si $\tau_{ij} < \tau_{min}$, entonces $\tau_{ij} = \tau_{min}$, con

$$\tau_{min} = \frac{\Delta\tau}{2 \cdot m \cdot (1-\rho)} \tag{5.5}$$

donde $\Delta\tau$ se calcula con la ecuación 4.4. De esta manera se impone una cota inferior y otra superior al nivel de feromonas en los arcos. Los valores especificados para τ_{min} y τ_{max} fueron tomados de [Stutzle2000]. Como puede notarse, tanto τ_{max} como τ_{min} están en función al $\Delta\tau$ de la mejor solución encontrada hasta el momento, es por eso que estos valores son actualizados cada vez que se encuentra una mejor solución global *gb*.

Los lectores interesados pueden encontrar, en el trabajo citado, una demostración formal de la existencia de una cota superior implícita para el nivel de feromonas producida por el efecto de evaporación de feromonas. [Stutzle2000] utiliza esta cota implícita como valor para τ_{max}. En el caso de τ_{min} se realizó un planteamiento similar y empíricamente se demostró un buen comportamiento al utilizar el valor de τ_{min} presentado anteriormente.

Como nivel inicial de feromonas se utiliza $\tau_0 = \tau_{max}$, empíricamente los autores del algoritmo demostraron que esto favorece la exploración inicial y mejora el rendimiento del MMAS. También se menciona el concepto de convergencia del MMAS, el cual define que el MMAS converge, si

para cada estado de decisión, uno de los estados alternativos a visitar posee el nivel de feromonas τ_{max} , mientras que las demás alternativas poseen un valor τ_{min} .

Una vez que el MMAS converge o está cerca de converger, se busca favorecer la exploración de otros caminos proporcionando más feromonas en los estados con bajo nivel de feromonas, esta acción se conoce como suavizamiento del nivel de feromonas (*Pheromone Trail Smoothing*), y se calcula según la ecuación:

$$\tau_{ij} = \tau_{ij} + fs \cdot (\tau_{max} - \tau_{ij}) \tag{5.6}$$

donde *fs* se denomina factor de suavizamiento. De esta forma los estados con inferiores niveles de feromonas recibirán mayor cantidad feromonas mediante el suavizamiento. Este mecanismo permite una gran exploración del espacio de búsqueda.

Los autores de este algoritmo demostraron en su trabajo, la eficiencia del MMAS en problemas de optimización mono-objetivo, y a la fecha, esta propuesta está considerada como uno de los mejores algoritmos ACO basados en la teoría de las hormigas.

5.1.4 Ant-Q Algorithm

El algoritmo Ant-Q está basado en reglas de Q-Learning, que se proponen en [Gambardella1995]. Las principales diferencias que presenta esta propuesta con respecto a los algoritmos basados en el Ant System son las siguientes: utiliza una regla distinta para escoger el siguiente estado a visitar, denominada regla proporcional pseudo-aleatoria *(pseudo-random proportional rule)*. Asumamos que nos encontramos en el estado *i*, entonces se escoge *j* como siguiente estado a visitar según:

$$j = \begin{cases} \max\limits_{j \in J_i} \left\{ \tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta} \right\} & si\ q \le q_0 \\ \hat{i} & en\ caso\ contrario \end{cases} \tag{5.7}$$

donde *q* es un valor escogido aleatoriamente con probabilidad uniforme en el intervalo [0,1], y q_0 ($0 \le q_0 \le 1$) es un parámetro que indica la probabilidad de utilizar la regla pseudo-aleatoria; $\hat{i}$ es una variable aleatoria seleccionada según la distribución de probabilidades establecida en la ecuación 4.3.

Además, introduce una regla de actualización de feromonas basada en el AS, pero con una diferencia. Se agrega otro componente a la regla, que se define como:

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij} + \rho\cdot(\Delta\tau + \gamma\cdot \max_{z\in J_j} \tau_{jz}) \quad (5.8)$$

donde γ se denomina paso de aprendizaje, y $\max_{z\in J_j} \tau_{jz}$ representa el nivel de feromonas del arco que posea mayor cantidad de feromonas de entre todos los arcos que se originan en *j*. De esta manera, se busca favorecer a los arcos que pertenecen a caminos altamente visitados, depositando más feromonas mediante el componente explicado.

La cantidad $\Delta\tau$ se calcula según la regla del AS (ecuación 5.1), y puede definirse de dos maneras, según se utilice la mejor solución encontrada (*gb*) o la mejor solución de la iteración (*ib*). Solo la hormiga que encontró esta mejor solución tiene permitido actualizar las feromonas.

Los autores del Ant-Q, compararon dos formas de escoger el siguiente estado en la regla pseudo-aleatoria y lo denominaron *con heurística* y *sin heurística*, que se refieren a utilizar o no la visibilidad en la regla respectivamente.

Los resultados experimentales de los autores del algoritmo demostraron que la visibilidad es útil al comienzo de las iteraciones. Sin embargo, luego de un número grande de iteraciones presenta resultados similares a utilizar solo las feromonas Esto puede explicarse considerando que inicialmente las feromonas no poseen ningún significado especial, pero luego de correr varias iteraciones se van aprendiendo buenos valores para las mismas.

5.1.5 Ant Colony System (ACS)

ACS es una modificación del Ant-Q citado anteriormente, y propuesto en [Dorigo1997]. Utiliza la misma regla proporcional pseudo-aleatoria presentada en la sección anterior (ecuación 5.7).

Las diferencias aparecen al actualizar las feromonas, ACS realiza dos actualizaciones, una global realizada cada vez que se completa una iteración, y otra local realizada por cada hormiga cada vez que transita de un estado a otro. La primera regla de actualización se conoce como actualización retardada *(delayed pheromone update)*, mientras que la última se conoce como actualización paso a paso *(step by step pheromone update)*. Generalmente la regla retardada se realiza en función a la calidad de la solución encontrada y no posee analogías con las hormigas reales (concepto artificial). Por su parte, la regla paso a paso fue introducida en el modelo simple del capítulo 4, y generalmente deposita una cantidad constante de feromonas, en analogía al comportamiento real de las hormigas.

La actualización global la realiza la hormiga que encontró la mejor solución global (*gb*), y se realiza según la ecuación genérica 4.4 y la cantidad a depositar se calcula como en el AS, según la ecuación 5.1. La actualización paso a paso se realiza también utilizando la ecuación 4.4., pero se deposita $\Delta\tau=\tau_0$, una cantidad constante de feromonas en el correspondiente arco utilizado.

τ_0 representa, además, el valor inicial para las feromonas, y los autores sugieren calcular su valor como:

$$\tau_0=\frac{1}{n\cdot S} \tag{5.9}$$

donde *n* es el número de componentes del problema a optimizar y *S* representa el costo de la mejor solución obtenida ejecutando una iteración de ACS sin feromonas, solo guiado por la visibilidad (equivalente a ejecutar un algoritmo avaro).

5.1.6 Omicron ACO (OA)

OA fue propuesto en [Gómez2004] y está inspirado en el Max-Min Ant System (MMAS) explicado anteriormente. La principal diferencia entre el MMAS y el OA es la manera de actualizar la tabla de feromonas. En OA se mantiene una población *P* con las *l* mejores soluciones o individuos desde que se inició la búsqueda. El mejor individuo de *P* en cualquier momento, es la solución *gb* mientras que el peor individuo en *P* se denomina *gw* (*global worst*). No se mantienen individuos repetidos en *P*. La actualización de feromonas utiliza un parámetro definido a priori *O*, denominado *Omicron* por los autores del algoritmo.

Inicialmente se inicializa la matriz de feromonas a 1, esto es $\tau_{ij}=1, \forall i, j\in 1,2,\ldots,n$. En cada iteración se genera una solución, que reemplaza a *gw* en *P*, si la nueva solución es mejor que *gw*. Luego de *K* iteraciones, se actualiza la matriz de feromonas. Para esto, primero se reinicia la matriz de feromonas al valor inicial 1, luego, se suma una cantidad *O*/*l* a cada elemento τ_{ij} cada vez que el arco (*i,j*) aparece en alguno de los *l* individuos de *P*. Este proceso se repite cada *K* iteraciones hasta alcanzar una condición de parada. Se puede notar que $1\leq\tau_{ij}\leq(1+O)$, donde $\tau_{ij}=1$ si el arco (*i,j*) no es utilizado por ningún individuo en *P*, y $\tau_{ij}=1+O$, si el arco (*i,j*) aparece en todos los individuos de *P*.

El siguiente estado a visitar se calcula según la ecuación genérica 4.3, al igual que el MMAS. Resulta interesante mencionar que el OA basa su actualización de feromonas en soluciones elitistas

pertenecientes a una población, buscando explorar explícitamente el espacio de soluciones acotado por las mejores *l* soluciones encontradas hasta el momento. Las soluciones que no forman parte de la población *P*, son olvidadas. No se realiza un proceso de evaporación de feromonas, y al igual que en el MMAS el nivel de feromonas en cada arco posee una cota inferior y otra superior, esta última en función al parámetro *O* definido a priori.

5.2. Algoritmos MOACO

Ya explicados los algoritmos ACO mono-objetivos, podemos describir en esta sección los distintos algoritmos MOACO existentes. Normalmente estos algoritmos son extensiones de algún algoritmo ACO mono-objetivo. Consecuentemente en esta sección presentaremos estas extensiones en detalle para cada algoritmo MOACO.

Otro punto interesante a mencionar es que en las siguientes secciones no se aplica el concepto de mejor solución *gb*, como se mencionaba antes, sino el concepto de un conjunto de soluciones Pareto óptimas y que constituye el Conjunto Pareto del algoritmo. Es importante recordar que se tratará de una optimización multi-objetivo, e inclusive, dependiendo de la estrategia, pueden existir varias visibilidades y/o varias tablas de feromonas.

Mantenemos en las siguientes secciones el contexto de minimización, sin perder con ello la generalidad.

5.2.1. Multiple Objective Ant Q Algorithm (MOAQ)

Este algoritmo se propuso en el trabajo de Mariano y Morales en [Mariano1999]. La regla de transición de estados y la actualización de feromonas están basadas en el algoritmo Ant-Q, mencionado. Se mantiene una colonia de hormigas para cada objetivo. De esta manera, en un problema con *b* objetivos tendremos *b* colonias de hormigas optimizando cada una un objetivo específico.

La regla de transición de estados se basa en los valores de la matriz de feromonas τ, y la visibilidad η, la transición en cada colonia se realiza probabilísticamente según la ecuación 4.3. La actualización de los valores τ_{ij} se realiza basada en la ecuación 5.8 del Ant-Q.

La idea del MOAQ es utilizar colonias especializadas en objetivos particulares, y compartir los conocimientos, representados por las soluciones encontradas e inclusive por los rastros de feromonas.

5.2.2. Bicriterion Ant (BIANT)

Este algoritmo fue propuesto por Iredi *et al.* [Iredi2001] para la resolución del TSP con dos objetivos y mantiene dos tablas de feromonas τ y τ' para los dos objetivos considerados. La distribución de probabilidades para seleccionar el siguiente estado está dada por:

$$p_{i,j}=\begin{cases}\dfrac{\tau_{i,j}^{\lambda\alpha}\cdot\tau'^{(1-\lambda)\alpha}_{i,j}\cdot\eta_{i,j}^{\lambda\beta}\cdot\eta'^{(1-\lambda)\beta}_{i,j}}{\sum_{x\in J_i}\tau_{i,x}^{\lambda\alpha}\cdot\tau'^{(1-\lambda)\alpha}_{i,x}\cdot\eta_{i,x}^{\lambda\beta}\cdot\eta'^{(1-\lambda)\beta}_{i,x}} & si\ j\in J_i \\ 0 & en\ caso\ contrario\end{cases} \tag{5.10}$$

Para cada hormiga *t* para $t\in\{1,2,\ldots,m\}$, λ_t se calcula a partir de:

$$\lambda_t=\frac{t-1}{m-1} \tag{5.11}$$

De esta manera se espera que las diferentes hormigas realicen búsquedas en distintas regiones del frente Pareto. La actualización y evaporación de feromonas se realiza para cada tabla según la ecuación 4.4. Solo las hormigas que encontraron soluciones no dominadas pueden actualizar las tablas de feromonas, y depositan una cantidad $\Delta\tau=1/l$, donde *l* es el número de hormigas de la iteración que generaron soluciones no dominadas. Estas soluciones no dominadas se mantienen en un conjunto externo, el conjunto Pareto del algoritmo.

5.2.3. Bicriterion Multi Colony (BicriterionMC o BIMC)

Este algoritmo extiende el BicriterionAnt y también se presenta en [Iredi2001], introduciendo la generalización de tener *b* colonias, cada una con su propia tabla de feromonas y visibilidad.

Utiliza una actualización por región para realizar la actualización de feromonas, en donde la secuencia de soluciones en el conjunto de soluciones no dominadas de la iteración se ordena con respecto a uno de los objetivos y se divide en *b* regiones de igual tamaño. Las hormigas que encontraron soluciones en la *i*-ésima región actualizan la colonia *i*, para $i\in[1,b]$. Este método fuerza de manera explícita a las colonias a buscar en diferentes regiones del frente Pareto.

El parámetro λ , mencionado en BicriterionAnt, se calcula de manera distinta. Cada colonia *i* mantiene un intervalo de valores para λ que se solapan con los valores para la colonia $i-1$ e $i+1$, el intervalo de la colonia *i* se calcula como: $[(i-1)/(b+1),(i+1)/(b+1)]$.

En el trabajo de los autores del algoritmo se propusieron otros valores para λ , pero empíricamente se obtuvo una mejor distribución del Frente Pareto obtenido utilizando la estrategia mencionada.

5.2.4. Pareto Ant Colony Optimization (PACO)

Este algoritmo se propuso en [Doerner2002], y se basa en la utilización de *b* tablas de feromonas (τ^b), una para cada objetivo. En cada iteración una hormiga computa una serie de pesos, que en [Doerner2002] fueron seleccionados uniformemente aleatorios, $\delta=(\delta_1,\delta_2,\ldots,\delta_b)$, y los utiliza al calcular la regla de transición de estados. Estando en el estado *i* se selecciona el estado *j* según:

$$j=\begin{cases}\max\limits_{j\in J_i}\left[\left(\sum\limits_{k=1}^{b}\delta_k\cdot\tau_{i,j}^{k}\right)^{\alpha}\cdot\eta_{i,j}^{\beta}\right] & si\ q<q_0\\ \hat{i} & en\ caso\ contrario\end{cases} \tag{5.12}$$

donde $\hat{i}$ es una variable aleatoria seleccionada según la distribución de probabilidades calculada con la ecuación:

$$p_{ij}=\begin{cases}\dfrac{(\sum\limits_{k=1}^{b}\delta_k\cdot\tau_{i,j}^{k})^{\alpha}\cdot\eta_{i,j}^{\beta}}{\sum\limits_{x\in J_i}[(\sum\limits_{k=1}^{b}\delta_k\cdot\tau_{i,x}^{k})^{\alpha}\cdot\eta_{i,x}^{\beta}]} & si\ j\in J_i\\ 0 & en\ caso\ contrario\end{cases} \tag{5.13}$$

Las dos mejores hormigas para cada objetivo actualizan la tabla de feromonas correspondiente a dicho objetivo según la ecuación 4.4, realizando una actualización elitista.

Cada vez que una hormiga avanza a otro estado, se realiza una actualización local paso a paso de las *b* tablas de feromonas según la ecuación 4.4, considerando un valor constante para $\Delta\tau=\tau_0$, que representa el valor inicial para las feromonas (definido a priori).

Básicamente podemos decir que este algoritmo extiende el ACS para volverlo multi-objetivo, con las adaptaciones de tener una tabla de feromonas por objetivo, y agregarlas de forma ponderada en el cálculo del siguiente estado a visitar.

5.2.5. Multi-Objective Ant Colony System (MOACS)

MOACS, propuesto por Barán y Schaerer en [Barán2003], es una extensión del MACS-VRPTW, este último propuesto en [Gambardella1999]. Fue implementado considerando dos objetivos, utiliza una matriz de feromonas y dos visibilidades, una para cada objetivo del problema. Utiliza la regla pseudo-aleatoria y se escoge el estado *j* como siguiente a visitar, estando en el estado *i*, según:

$$j = \begin{cases} \underset{j \in J_i}{max} \left\{ \tau_{i,j} \left[\eta^0_{i,j} \right]^{\lambda\beta} \left[\eta^1_{i,j} \right]^{(1-\lambda)\beta} \right\} & si\ q < q_0 \\ \hat{i} & en\ caso\ contrario \end{cases} \tag{5.14}$$

El cálculo de la variable aleatoria $\hat{i}$ se realiza utilizando la probabilidad p_{ij} dada por:

$$p_{ij} = \begin{cases} \dfrac{\tau_{i,j} \left[\eta^0_{i,j} \right]^{\lambda\beta} \left[\eta^1_{i,j} \right]^{(1-\lambda)\beta}}{\sum_{x \in J_i} \tau_{i,x} \left[\eta^0_{i,x} \right]^{\lambda\beta} \left[\eta^1_{i,x} \right]^{(1-\lambda)\beta}} & si\ j \in J_i \\ 0 & en\ caso\ contrario \end{cases} \tag{5.15}$$

Cada vez que una hormiga se mueve del estado *i* al estado *j*, realiza la actualización local de feromonas según la ecuación 4.4, con $\Delta\tau = \tau_0$, el valor inicial para las feromonas.

En el caso de encontrar una solución no dominada, se actualiza el conjunto Pareto *CP* y se reinicializa la tabla de feromonas, considerando que la información fue aprendida por medio de soluciones dominadas [Barán2003]. Si la solución encontrada es dominada se realiza la actualización de feromonas según la ecuación 4.4, depositando una cantidad de feromonas según la ecuación multi-objetiva 4.6.

Como el nombre del algoritmo menciona, es una extensión multi-objetiva del ACS. Es interesante notar que el concepto de multi-objetivo se basa en la utilización de varias visibilidades, manteniendo una única tabla de feromonas, que contiene la información aprendida, considerando todos los objetivos al mismo tiempo.

Inicialmente parece ser una estrategia más escalable con respecto al número de objetivos, ya que no se necesitan muchas tablas de feromonas y tampoco se requiere tiempo de procesamiento para agregar varias tablas de feromonas al decidir que camino utilizar. En [García-Martínez2004] se demuestra experimentalmente que a la fecha es el mejor algoritmo para la resolución del TSP bi-objetivo.

5.2.6. Multiobjective Max-Min Ant System (M-MMAS o M3AS)

Este algoritmo, propuesto en [Pinto2005], extiende el Max-Min Ant System [Stutzle2000] para resolver problemas multi-objetivos. Se utilizó inicialmente para resolver el problema de enrutamiento *multicast* multi-objetivo. Pinto *et al.* optimizaron cuatro objetivos. Se mantiene una tabla de feromonas τ global que mantiene información de feromonas considerando todos los objetivos a optimizar. Una hormiga estando en el estado *i* escoge el siguiente estado a visitar, de acuerdo con la probabilidad *p* dada según la ecuación 5.15 del MOACS.

Las soluciones no dominadas actualizan la tabla de feromonas según la ecuación 4.4, y depositan una cantidad de feromonas según la ecuación 4.6. Si $\tau_{ij} > \tau_{max}$, entonces $\tau_{ij} = \tau_{max}$, con $\tau_{max} = \Delta\tau/(1-\rho)$, y si $\tau_{ij} < \tau_{min}$, entonces $\tau_{ij} = \tau_{min}$, con $\tau_{min} = \Delta\tau/2m(1-\rho)$. De esta manera se impone una cota inferior y otra superior al nivel de feromonas en los arcos, al igual que en el MMAS.

5.2.7. COMPETants (COMP)

Propuesto en [Doerner2003], fue utilizado en problemas bi-objetivos, con dos tablas de feromonas y dos visibilidades. El número de hormigas para cada colonia no es fijo, y se asigna de manera dinámica durante la ejecución del algoritmo. Cada vez que todas las hormigas han construido sus soluciones, las colonias con las mejores soluciones reciben más hormigas para la siguiente iteración. Así el número de hormigas para cada colonia es adaptado dinámicamente de acuerdo al rendimiento de cada colonia.

La probabilidad de transición de estados utiliza la ecuación 4.3. Se mantienen hormigas espías en cada colonia cuyas probabilidades de transición de estados utilizan información de ambas tablas de feromonas según la ecuación:

$$p_{ij}=\begin{cases}\dfrac{[0{,}5\cdot\tau_{i,j}+0{,}5\cdot\tau'_{i,j}]\cdot\eta_{i,j}^{\beta}}{\sum_{x\in J_i}[0{,}5\cdot\tau_{i,x}+0{,}5\cdot\tau'_{i,x}]\cdot\eta_{i,x}^{\beta}} & si\ j\in J_i \\ 0 & en\ caso\ contrario\end{cases} \tag{5.16}$$

De esta manera se implementa un enfoque cooperativo entre las colonias. El algoritmo está basado en el *ASrank*, del cual mantuvo el enfoque de realizar un *ranking* y permitir a las mejores l hormigas de cada objetivo actualizar la correspondiente tabla de feromonas. La cantidad $\Delta\tau$ que deposita cada hormiga se calcula como:

$$\Delta\tau=1-\frac{\mu-1}{l} \tag{5.17}$$

donde μ es la posición de la hormiga en el *ranking* de acuerdo al objetivo asignado a la colonia de la hormiga en cuestión.

La adaptación dinámica de las hormigas se realiza en base al costo medio, de acuerdo al objetivo asignado a cada colonia. Luego de una iteración, el número de hormigas de la primera colonia (asignada al primer objetivo) será:

$$\frac{\hat{f}_2}{\hat{f}_2\cdot\hat{f}_1} \tag{5.18}$$

donde $\hat{f}_1$ representa la evaluación promedio de las soluciones no dominadas de la colonia utilizando la primera función objetivo, y $\hat{f}_2$ se define análogamente pero utilizando la segunda función objetivo.

Estos costos medios se calculan promediando el costo de todas las soluciones no dominadas encontradas por cada colonia. El número de hormigas espías de la primera colonia se calcula como:

$$\frac{f_1(gb^1)}{4\cdot f_1(gb^1)\cdot f_1(gb^2)} \tag{5.19}$$

donde gb^1 representa la mejor solución encontrada por la primera colonia y gb^2, la mejor solución de la segunda colonia. Análogamente a lo realizado con la primera colonia, se puede obtener el número de hormigas y el número de espías para la segunda colonia.

5.2.8. Multiobjective Omicron ACO (MOA)

Este algoritmo fue propuesto en [Gardel2005] y está basado en el algoritmo OA propuesto en [Gómez2004]. El algoritmo fue implementado considerando dos objetivos, utiliza una tabla de feromonas y dos visibilidades, una para cada objetivo del problema.

El algoritmo posee una implementación muy similar al OA. Al igual que el OA, utiliza una población *P* de soluciones no dominadas que pueden actualizar el nivel de feromonas, generalmente se utiliza el conjunto Pareto como población *P*. Como probabilidad de transición de estados se utilizó la extensión multi-objetiva del MOACS, ecuación 5.15.

La actualización de feromonas se basa en depositar una cantidad constante en todos los arcos que componen todas las soluciones de la población *P*. La regla se basa en el parámetro *O (Omicron)* al igual que en el OA. De esta manera, se mantiene el nivel de feromonas con una cota inferior y otra superior, como se muestra en la sección dedicada al OA.

En su trabajo, los autores del algoritmo resolvieron el problema de la compensación de potencia reactiva en el contexto multi-objetivo, y denominaron a la versión multi-objetiva del OA como *Omicron Eléctrico*. En este trabajo se utiliza la nomenclatura más genérica MOA, representando al OA multi-objetivo.

5.2.9. Multiobjective Ant System (MAS)

El *MAS* que propone por primera vez este trabajo es una extensión sencilla del *AS* (*Ant System*) [Dorigo1996] para manejar múltiples objetivos. Las modificaciones realizadas se centran en la selección del siguiente estado y la actualización de las feromonas. La selección estocástica del siguiente estado a visitar utiliza la ecuación 5.15, basado en el *MOACS*. De esta manera, se mantiene una única tabla de feromonas con una visibilidad para cada objetivo a optimizar, y las hormigas se distribuyen en regiones del espacio de búsqueda por medio del valor λ utilizado.

λ toma valores discretos y distancias uniformes entre 1 y el número *m* de hormigas, o sea $\lambda \in \{1,2,\ldots,m\}$. Así se obtiene una distribución en el espacio de soluciones del problema, donde por ejemplo, en el contexto bi-objetivo, la hormiga con $\lambda = 1$ se especializará en el segundo objetivo y un valor $\lambda = m$ significa una especialización hacia el primer objetivo. Claramente los demás valores para λ representan valores intermedios entre estos casos extremos de especialización.

La actualización de feromonas se realiza luego de construir todas las soluciones de la iteración y es realizada por las soluciones no dominadas encontradas en la iteración. La regla de actualización utilizada es la ecuación genérica 4.4 con $\Delta\tau$ según la ecuación 4.6, de esta manera la actualización queda en función a la evaluación de la solución atendiendo todos los objetivos del problema.

Se agregó un mecanismo de control de convergencia al algoritmo, que consiste en reiniciar la tabla de feromonas si durante K' generaciones, con K' definido a priori, no se encontraron soluciones no dominadas. De esta manera, se favorece la exploración de nuevos caminos. Este mecanismo ayuda a continuar la búsqueda ante situaciones de convergencia a óptimos locales o el problema denominado *stagnation behavior*, mencionado en la implementación del *AS*, en el cual se terminaba el algoritmo ante esta situación. En la nueva propuesta, se reinicializa la tabla de feromonas de manera a continuar la ejecución. El pseudo-código del MAS se muestra en la figura 5.1.

```
procedure MAS
inicializar_parametros()
while not condicion_parada()
      generacion=generacion + 1
      repeat para cada hormiga k
            construir_solucion()        // según ecuación (4.9)
            actualizar_conjunto_pareto()
            actualizar_feromonas()      // según ecuación (4.2)
      end repeat
      if (no_cambio(CP,K')) //sin cambios en K' generaciones
            reiniciar_feromonas()
end while
print ConjuntoPareto
end
```

Figura 5.1. Pseudo-código del MAS

En el siguiente capítulo se explican algunas técnicas de paralelización de algoritmos ACO. Estas técnicas buscan mejorar el desempeño de estos algoritmos ACO basados en la disponibilidad y utilización de una mayor cantidad de recursos computacionales.

Capítulo 6

Estrategias de Paralelización de Algoritmos ACO

Dada la naturaleza inherentemente independiente de las hormigas individuales, es razonable pensar que estas hormigas (o colonias de hormigas) puedan ser simuladas en paralelo o inclusive en una forma totalmente distribuida.

Uno de los enfoques a considerar podría ser el de paralelizar la construcción de las soluciones mismas. También puede considerarse la posibilidad de contar con varias colonias resolviendo un mismo problema en diversos procesadores. Los enfoques mencionados pueden ser implementados de manera síncrona o asíncrona. También, en otros trabajos, se propusieron versiones parcialmente asíncronas de algoritmos ACO paralelos [Bullnheimer1998].

La computación paralela es considerada como un método efectivo en costo para tratar complejos problemas computacionales. Los computadores paralelos y el surgimiento de los *Cluster of Workstations* (Conjunto de estaciones de trabajo) favorecieron la aplicabilidad de los métodos paralelos, además de la existencia de librerías computacionales que facilitan la paralelización de algoritmos.

Resulta interesante introducir algunos conceptos de la teoría de la computación paralela como la granularidad, el *speed-up*, y los tiempos de sincronización y comunicación entre procesos, ya que en las siguientes secciones son utilizados en la descripción de distintas estrategias de paralelización de algoritmos ACO. Lectores interesados en profundizar estos conceptos pueden referirse al libro de Kumar [Kumar2003].

El número y tamaño de las tareas en las cuales se descompone un problema determina la granularidad de la descomposición. Una descomposición en grandes cantidades de tareas simples se denomina descomposición de grano fino. La descomposición en una menor cantidad de tareas complejas es denominada descomposición de grano grueso. En este capítulo se presentarán estrategias de paralelización de distintas granularidades, es por eso que resulta interesante conocer el concepto de granularidad en el contexto paralelo.

Otro concepto de interés, es la aceleración (*speed-up*) que mide el beneficio en el tiempo de ejecución logrado por una paralelización de una aplicación sobre una versión secuencial de la misma. La aceleración se define como la relación entre el tiempo secuencial necesario para resolver el problema particular, y el tiempo de procesamiento necesario en *NPr* unidades de procesamiento. Al paralelizar un algoritmo buscamos conseguir la mayor aceleración posible.

En cuanto a tiempo de una computación paralela, podemos decir que un programa paralelo utiliza su tiempo de procesamiento en resolver el problema dado, en la comunicación con los demás procesadores, e inclusive algún tiempo permanece ocioso debido a estrategias síncronas, si fuera el caso.

En general, una granularidad más fina implica una comunicación entre procesadores más constante, mientras que una granularidad de grano grueso implica una comunicación menos frecuente. No obstante, considerando el *speed-up*, resulta claro que una granularidad más fina implica ejecutar mayor cantidad de instrucciones en paralelo lo que llevaría a obtener un mayor *speed-up*, si se suponen que los costos de comunicación entre procesadores no son muy elevados. Estas características normalmente se dan en el caso de los computadores multiprocesadores.

En muchos casos, un computador con múltiples procesadores no está disponible, y generalmente se cuenta con múltiples computadores personales. En tal caso, se busca un punto óptimo de granularidad (entre fino y grueso) de manera a ejecutar la mayor cantidad de instrucciones en paralelo sin deteriorar el *speed-up* obtenido a causa de un gran volumen de comunicación entre procesos.

En la actualidad, una plataforma utilizada para la paralelización es denominada *Cluster of Workstations* (como ya se adelantó anteriormente), que representa un conjunto de computadores personales conectados mediante una red de computadoras. En este tipo de plataformas normalmente

se implementan sistemas distribuidos, en los cuales se distribuye un algoritmo (o copias del mismo) en distintos computadores para resolver un problema específico.

En este capítulo se mencionarán algunas posibilidades de paralelización de los algoritmos ACO para tal ambiente.

6.1 Paralelización a nivel de Hormigas

La idea básica de la paralelización a nivel de hormigas es asignar una hormiga a cada unidad de procesamiento (*UP*) cuando se cuentan con *NPr* unidades de procesamiento, de modo a poder construir las soluciones concurrentemente. Esta noción es atractiva dado que las hormigas son independientes entre sí y se comunican de manera indirecta mediante las feromonas.

Una de las primeras versiones paralelas de un algoritmo ACO fue propuesta por Bolondi y Bondanza [Bolondi1993]. Realizaron la implementación del "*Ant System*" paralelo para el TSP. Desafortunadamente, los resultados experimentales mostraron que los costos de comunicación pueden ser un problema considerable en un entorno distribuido debido a la granularidad fina del enfoque, ya que las hormigas utilizan la mayor parte del tiempo comunicando a las demás hormigas las modificaciones que hicieron a los rastros de feromonas. Como resultado, el algoritmo no obtuvo el rendimiento esperado y su escalabilidad era pobre al aumentar el tamaño del problema, como se menciona en [Dorigo1999].

También Bullnheimer *et al*. [Bullnheimer1997] propusieron un algoritmo donde se asigna una hormiga a una unidad de procesamiento. En su trabajo, concluyeron que es más eficiente utilizar un enfoque distinto, que se describe en la siguiente sección.

Cabe destacar también que el enfoque descrito aquí asume que se cuenta con una cantidad de Unidades de Procesamiento igual a la cantidad de hormigas que se piensan utilizar. Generalmente, esta condición no se cumple en la realidad, por lo que un enfoque en el cual se asignan varias hormigas a una Unidad de Procesamiento es considerado a continuación.

6.2 Paralelización a nivel de Colonias

Otro enfoque para la paralelización de los algoritmos ACO es ejecutar no sólo el código correspondiente a la construcción de una solución, sino el algoritmo correspondiente a una colonia completa en forma concurrente.

Bullnheimer *et al*. [Bullnheimer1997] propusieron dos versiones paralelas de grano relativamente grueso del "*Ant System*" (AS) llamadas Implementación Paralela Síncrona (IPS) e Implementación Paralela Parcialmente Asíncrona (IPPA).

Con IPS básicamente se divide la colonia en *NPr* subcolonias. Cada subcolonia actúa como una colonia completa e implementa de este modo un algoritmo AS estándar. Cada vez que una subcolonia ejecuta una iteración del algoritmo, envía las soluciones encontradas en la iteración a un proceso *Maestro*, que se encarga de realizar la actualización global de los niveles de feromonas. Luego, este proceso *Maestro* envía a todas las subcolonias la información de feromonas de manera a ejecutar otra iteración.

En IPPA el envío de soluciones e información de feromonas entre una subcolonia y el proceso *Maestro* es realizado cada vez que la subcolonia ejecuta un número fijo de iteraciones, definido *a priori*. Los dos algoritmos fueron evaluados mediante simulaciones. Los resultados mostraron que IPPA tuvo un mejor rendimiento que IPS, donde las métricas fueron tiempo de ejecución y *speed-up* (aceleración del tiempo total de ejecución). Esto se debe probablemente a la reducida cantidad de comunicación necesaria por IPPA, ocasionado por el intercambio menos frecuente de feromonas entre subcolonias.

Un aspecto interesante de cualquier implementación paralela que intercambia información entre hormigas concurrentes es el tipo de información de feromonas que deberían ser intercambiado entre ellas y cómo esta información debería ser utilizada para actualizar la información de feromonas de las hormigas. En el trabajo de Dorigo [Dorigo1999] se consideraron: (i) el intercambio de la mejor solución global, donde cada conjunto de hormigas usa la mejor solución global para determinar en que aristas incrementar las feromonas; (ii) el intercambio de mejores soluciones locales, en la que cada conjunto de hormigas recibe la mejor solución local de todos los otros conjuntos de hormigas y las utiliza para actualizar los rastros de feromonas; y (iii) el intercambio de toda la información de feromonas, donde cada colonia computa el promedio de la información de feromonas de todas las colonias. Por ejemplo, si $\tau^{UP}=[\tau_{ij}^{UP}]$ es la información de feromonas de la colonia *UP*, $1 \le UP \le NPr$, donde *NPr* representa la cantidad de procesadores disponibles, entonces cada colonia *UP* envía τ^{UP} a las demás colonias y luego calcula cada componente como:

$$\tau_{ij}^{UP} = \frac{1}{NPr} \cdot \left[\sum_{h=1}^{NPr} \tau_{ij}^{h} \right], 1 \le i, j \le n \qquad (6.1)$$

Resultados experimentales, como menciona el trabajo de Dorigo [Dorigo1999], indicaron que los métodos (i) y (ii) son más rápidos y encuentran mejores soluciones que el método (iii).

6.3 Corridas Paralelas Independientes

Una de las formas más sencillas de obtener un algoritmo ACO paralelo es ejecutándolo en varios procesadores simultáneamente de manera independiente, y luego recolectar las soluciones obtenidas por cada procesador y seleccionar la solución que obtenga una mejor evaluación. Una implementación de este tipo fue realizada por Stützle [Stützle1998] y la denominó *Corridas Paralelas Independientes*. En esta implementación se considera un contexto mono-objetivo, pero para el caso multi-objetivo se podría utilizar la noción de un conjunto de mejores soluciones en vez de una única mejor solución.

Se menciona en el trabajo de Stützle [Stützle1998] que éste enfoque presenta escalabilidad con respecto a la calidad de las soluciones y el tiempo total de ejecución, a pesar de la sencillez de su implementación. Puede notarse que esta implementación paralela requiere un esfuerzo de implementación casi nulo, contando con las implementaciones secuenciales del algoritmo a paralelizar, como menciona el trabajo de Stützle [Stützle1998]. Esto se debe a que simplemente se ejecutan concurrentemente los algoritmos ACO, sin contar con un esquema de intercambio de información o con algún proceso centralizado que oriente a los distintos algoritmos. Solo se necesita recolectar las mejores soluciones una vez que todos los procesadores terminaron su ejecución.

También se demuestra en el trabajo de Stützle [Stützle1998] que la probabilidad de encontrar soluciones óptimas mejora al aumentar el número de procesadores en los que se ejecuta una instancia del algoritmo, obteniendo de este modo una considerable aceleración y escalabilidad.

Una particularidad de este enfoque es la posibilidad de ejecutar un mismo algoritmo con distintas configuraciones de parámetros en los distintos procesadores disponibles. Se espera de este modo encontrar una mayor diversidad de soluciones, lo que supone una mayor exploración del espacio de soluciones del problema.

Una restricción al respecto de este tipo de implementaciones es que solo son aplicables con algoritmos aleatorios, es decir en donde el comportamiento se basa en decisiones aleatorias. Este es el caso de los algoritmos basados en colonias de hormigas, donde la construcción de soluciones es esencialmente un proceso aleatorio.

Comentario acerca de implementaciones síncronas y asíncronas

En los trabajos de Dorigo [Dorigo1999] y Bullnheimer [Bullnheimer 1998] puede notarse que las implementaciones asíncronas presentan rendimientos superiores a las implementaciones síncronas. Inclusive un esquema parcialmente asíncrono como se describió en la sección 6.2, experimentalmente presentó un mejor comportamiento, según describe el trabajo de Bullnheimer [Bullnheimer 1998].

Con el asincronismo se obtienen importantes aceleraciones en el tiempo de ejecución, ya que no se necesitan tiempos de sincronización, y se reduce el tiempo ocioso de un procesador al no necesitar tiempos de sincronización en los cuales algunos procesadores deberán esperar a otros.

Para el caso de los *Clusters of Workstations*, que representa la realidad en muchos países donde no es fácil conseguir una supercomputadora con múltiples procesadores, las implementaciones asíncronas resultan más naturalmente aplicables. Esto es debido a que el *Cluster* es generalmente un sistema distribuido de computadores personales, altamente asíncrono, donde cada computador posee su propia velocidad y capacidad de procesamiento.

Capítulo 7

Equipo Distribuido de Algoritmos

(*Team Algorithm*)

La idea principal de los *Team Algorithms* (TA) es dividir un problema complejo en sub-problemas menores que pueden ser resueltos con varios métodos disponibles, combinando los resultados parciales de cada sub-problema de modo a encontrar una buena solución global. Por otro lado, se pretende asegurar la convergencia global al complementar diferentes algoritmos para la resolución de un mismo sub-problema de gran complejidad [Barán2002].

En general, es altamente improbable que exista un único algoritmo que pueda resolver todos los sub-problemas igualmente bien. Es mejor entonces escoger para cada sub-problema el mejor algoritmo conocido para el mismo, buscando al mismo tiempo asegurar la convergencia a buenas soluciones del problema [Barán2002, Talukdar1997]. Se busca que los distintos métodos trabajen en equipo de manera a lograr en conjunto lo que probablemente no podrían lograr trabajando de manera independiente. El resultado es una forma de organización híbrida, denominado *Team Algorithm*, que combina las características de los diferentes métodos utilizados en la resolución de un problema.

Uno de los principales problemas en un TA es la partición del problema en sub-problemas. No es una tarea sencilla determinar la partición óptima del problema y existen métodos que buscan descomponer problemas de manera óptima, por ejemplo Benítez *et al.* [Benítez1996] realizaron un

trabajo donde se estudia la partición de sistemas de ecuaciones lineales para su resolución en un ambiente computacional distribuido.

Los *Team Algorithms* pueden ser naturalmente implementados en un entorno paralelo, asignando diferentes sub-problemas a cada procesador de un sistema distribuido, por ejemplo una red de computadoras. Un estudio formal, con un análisis de convergencia puede encontrarse en [Barán1996]. En el capítulo anterior se mencionaron alternativas de paralelización de algoritmos basados en colonias de hormigas. Este trabajo propone un equipo distribuido de algoritmos basados en colonias de hormigas.

El "*Team Algorithm*" propuesto se asemeja a la estrategia de corridas paralelas independientes, mencionado en el capítulo anterior, con algunas adaptaciones. Primero, el equipo utiliza inicialmente algoritmos distintos en cada procesador del sistema distribuido. Además, el equipo propuesto utilizará algoritmos multi-objetivos, con lo cual mantiene un conjunto de mejores soluciones, a diferencia de lo explicado en el capítulo 6.

Por ende, se espera que el equipo de algoritmos presente las mismas ventajas mencionadas para el método de las corridas paralelas independientes, además de otras que serán mencionadas en este capítulo.

7.1. Equipos Asíncronos (*A-Teams*)

Las implementaciones de algoritmos paralelos en sistemas distribuidos asíncronos se incrementan conforme las redes de computadoras convergen a una única gran red, considerando también que las arquitecturas denominadas "*Clusters of Workstations"* (COW) [Fernández2005] son relativamente más accesibles que aquellas arquitecturas de multiprocesadores con memoria compartida. Por este motivo, es de interés analizar los equipos de algoritmos asíncronos.

Como cada procesador trabaja a su velocidad y eventualmente, con un algoritmo asignado exclusivamente a él, una implementación asíncrona elimina los tiempos ociosos (*idle*) de sincronización acelerando el proceso de búsqueda [Barán2002].

La combinación paralela asíncrona descrita se conoce como *Asynchronous Team* (*A-Team*) y se puede observar un creciente interés en la utilización de los mismos para diversas aplicaciones [Barán2002, Talukdar1997]. En efecto, existen trabajos que explotan la posibilidad de utilizar varias estrategias o técnicas para resolver un mismo problema. Por ejemplo, combinando métodos

numéricos iterativos con Algoritmos Genéticos aplicados a la resolución de sistemas algebraicos de ecuaciones no lineales [Talukdar1983] o la utilización de diferentes algoritmos evolutivos en la resolución de diferentes problemas multi-objetivos [Fernández2005].

De manera a especificar detalladamente un equipo de algoritmos, se presenta una nomenclatura formal utilizada posteriormente. Dado un problema $F(X)=[\zeta_1(X),\zeta_2(X),\ldots,\zeta_h(X)]^T=0$ a resolver, dividido en h sub-problemas $\zeta_i(x)=0 \forall\, i\in\{1,2,\ldots,h\}$, y $X=[x_1, x_2, \ldots, x_h]^T$ que representa el vector de variables independientes, dividido en h particiones $x_i \forall\, i\in\{1,2,\ldots,h\}$, deseamos encontrar X tal que $F(X)=0$, por medio de un *A-Team*.

Denotaremos como $G_i \forall\, i\in\{1,2,\ldots,k\}$, al *i*-ésimo algoritmo utilizado, donde *k* es el número de algoritmos distintos utilizados. Existen varias estrategias posibles de interacción entre los procesadores componentes del *Team*, algunas de las cuales se mencionan a continuación.

7.1.1. Solución por bloques

A cada procesador *i* se le asigna un sub-problema local $\zeta_i(x_i)=0$. Este sub-problema es resuelto en el procesador *i* utilizando el algoritmo G_i. Luego de obtener un nuevo valor para x_i, lo transmite a los demás procesadores y recibe de ellos el valor más actualizado de $x_j \;\forall\, j\neq i$, véase la figura 7.1. El proceso continúa mientras no se cumpla con un criterio de parada.

Como se aprecia en la figura 7.1, en el modelo de solución por bloques se pretende dividir el problema en sub-problemas de menor tamaño y asignar éstos sub-problemas a distintos procesadores. De esta forma el problema es resuelto en paralelo y posiblemente con distintos métodos.

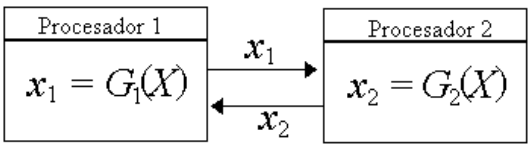

Figura 7.1. Esquema de resolución empleando la estrategia por Bloques en dos procesadores

7.1.2. Solución con Solapamiento (*Overlapping*)

En este caso, cada procesador resuelve todo el problema $F(X)=0$ con el algoritmo asignado al mismo, enviando sus resultados parciales a un *Administrador*, quien combina estos resultados en

una forma pre-establecida de modo a obtener una única solución global que es enviada a cada procesador para la siguiente iteración, véase la Figura 7.2.

Para el ejemplo la Figura 7.2, los valores de c, w_1 y w_2 representan pesos asignados a los procesadores y por ende a los algoritmos que éstos ejecuten. Al escoger estos pesos se debe asegurar que las soluciones ya encontradas no sean descartadas [Barán2002]. Generalmente $c, w_1, w_2 \in (0,1) \wedge c+w_1+w_2=1$, en [Barán2002] pueden encontrarse definiciones formales sobre los valores asignados a estos pesos.

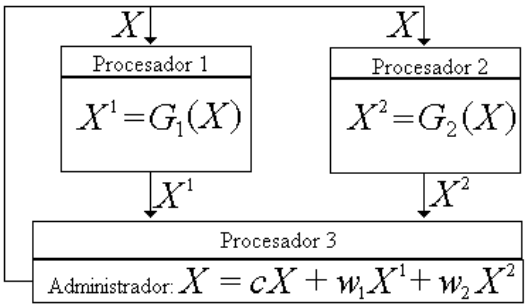

Figura 7.2. Esquema de resolución empleando *Overlapping*

7.1.3. *A-Team* Generalizado

Un *A-Team* generalizado (ver Figura 7.3) combina soluciones por bloques y solapamiento. En este caso, también el problema es divido en sub-problemas de menor tamaño, y un sub-problema puede ser asignado a único procesador o puede solaparse en distintos procesadores, donde cada procesador ejecuta algún algoritmo específico. En el caso de existir sub-problemas (variables) solapados, se debe especificar un *Administrador* que agregue las soluciones generadas por los distintos procesadores y distribuya el valor de la solución a los demás procesadores para los siguientes cálculos.

Cabe destacar que también es posible implementar un *A-Team* similar al ilustrado en la Figura 7.3 utilizando sólo dos procesadores, donde el proceso Administrador, debido a su menor complejidad, puede ejecutarse en alguno de los procesadores que resuelve un sub-problema particular.

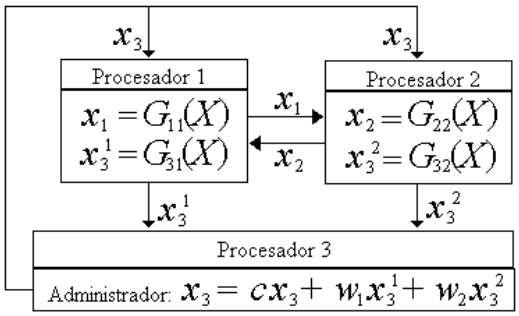

Figura 7.3. Esquema de resolución empleando un *A-Team Generalizado*. Se ilustra un ejemplo con tres procesadores. G_{ij} representa el algoritmo utilizado para resolver el sub-problema $\zeta_i(x_i)=0$ en el procesador *j*.

7.2. *Team Algorithm* de algoritmos ACO multi-objetivos (TA)

Con el objetivo de mejorar el desempeño de los algoritmos ACO multi-objetivos se propone en esta sección un equipo de algoritmos ACO multi-objetivos basado en la teoría de la computación paralela. Por medio de esta técnica se pretende utilizar todos los recursos disponibles y mejorar el tiempo de respuesta del mismo [Barán2002].

Utilizando un *Team Algorithm* [Barán2002] se combinan múltiples algoritmos para la resolución del mismo problema buscando mejorar el tiempo de procesamiento, e inclusive la calidad de las soluciones obtenidas, ya que distintos métodos son ejecutados en paralelo y combinados entre sí. De esta manera, se explora el espacio de soluciones del problema con varios métodos, esperando que un equipo de algoritmos logre una mayor exploración y por ende, mejores resultados.

En diversos trabajos se demostró empíricamente la eficiencia de los *Team Algorithms* [Barán1995, Barán1996, Barán1998, Fernández2005]. Este trabajo propone la combinación de 9 algoritmos ACO multi-objetivos que constituyen en la actualidad el estado del arte en la optimización multi-objetivo basada en colonias de hormigas, que son mencionados en la tabla 7.1. Se incluye en el equipo al MAS, que es una propuesta del presente trabajo.

El *Team Algorithm* consiste en 9 procesos esclavos que ejecutan algún algoritmo ACO multi-objetivo en particular (ver tabla 7.1) y un proceso Administrador que se encarga de la distribución de los algoritmos a ser ejecutados en cada proceso esclavo así como de la recolección de soluciones no dominadas generadas por dichos algoritmos. De esta manera, combina estas soluciones no

dominadas atendiendo la dominancia Pareto, obteniendo un Conjunto Pareto global que corresponde a las mejores soluciones encontradas por el equipo.

Tabla 7.1. 9 Algoritmos ACO multi-objetivos escogidos para el *Team Algorithm*

Algoritmo	**Año**	**Referencia**
Bicriterion Ant (BIANT)	2001	Iredi *et al.* [Iredi2001]
Bicriterion Multi Colony (BIMC)	2001	Iredi *et al.* [Iredi2001]
COMPETants (COMP)	2003	Doerner *et al.* [Doerner2003]
Multiobjective Ant System (MAS)	2006	*Presente trabajo*
MultiObjective Ant Colony System (MOACS)	2003	Barán *et al.* [Barán2003]
Multiobjective-Max Min Ant System (M3AS)	2005	Pinto *et al.* [Pinto2005]
Multiple Objective Ant Q (MOAQ)	1999	Mariano *et al.* [Mariano1999]
Multiobjective Omicron ACO (MOA)	2005	Gardel *et al.* [Gardel2005]
Pareto-Ant Colony Optimization (PACO)	2002	Doerner *et al.* [Doerner2002]

El equipo se ejecuta hasta cumplir con un criterio de parada global definido a priori, que pueden ser un tiempo máximo y/o un número de iteraciones. En cada iteración, luego de obtener los frentes generados por cada proceso esclavo, se realiza una selección elitista del mejor algoritmo de la iteración atendiendo a la cantidad de soluciones no dominadas aportadas al frente Pareto global. Este algoritmo así seleccionado reemplazará en la siguiente iteración al peor algoritmo determinado con el mismo criterio. En caso de existir igualdades al determinar el mejor o el peor algoritmo, se utiliza una ruleta con probabilidades uniformemente distribuidas para seleccionar el algoritmo correspondiente. Los demás procesos continúan su ejecución manteniendo su algoritmo en la siguiente iteración. En la figura 7.4 puede observarse el pseudo-código del proceso Administrador.

Los procesos esclavos reciben los parámetros del Administrador, como el criterio de parada para cada iteración y el algoritmo a ejecutar. En una iteración, cada proceso esclavo ejecuta su algoritmo hasta cumplir con el criterio de parada establecido por el Administrador. Luego, envía al Administrador su conjunto de soluciones no dominadas encontradas hasta el momento y espera recibir una orden para continuar la siguiente iteración o para finalizar el proceso esclavo. En caso de continuar la ejecución, el Administrador establece si debe continuar con el mismo algoritmo o con

uno nuevo que el Administrador especifica. El pseudo-código de un proceso esclavo se muestra en la figura 7.5.

```
procedure Administrador
inicializar_procesos_esclavos(N_ESCLAVOS)
while not condicion_parada()
      for i=1 to N_ESCLAVOS
            FP_i=recibir_FrentePareto(proceso_i)
            actualizar_FrentePareto_global(FP_i)
      end for
      seleccion_elitista_de_algoritmos()
      continuar_procesos_esclavos()
end while
print ConjuntoPareto
end
```

Figura 7.4. Pseudo-código del proceso Administrador.

En el siguiente capítulo se presentará el ambiente computacional en el cual se implementaron todos los algoritmos descritos en el trabajo, y también la manera en que se realizarán las comparaciones.

```
procedure Esclavo
recibir_parametros_del_Administrador()
repeat
      ejecutar_algoritmo(condicion_parada_local,algoritmo)
      enviar_FrentePareto(Administrador)
      recibir_ordenes_del_Administrador()
until Administrador_termina_ejecucion()
end
```

Figura 7.5. Pseudo-código de un proceso esclavo.

Además se mostrará en el marco experimental los resultados de todas las ejecuciones realizadas. Estas ejecuciones abarcan a todos los algoritmos MOACO presentados y al *Team Algorithm*.

Capítulo 8

Implementación Computacional

Conociendo los algoritmos MOACO existentes y el equipo de algoritmos explicado en el capítulo anterior, en este capítulo presentamos la implementación computacional de los mismos. En las primeras secciones se explican las configuraciones del entorno donde fueron implementados los algoritmos y se mencionan las instancias particulares de cada clase problema resuelto en este trabajo. Además, se presenta la configuración de los parámetros utilizados para los algoritmos MOACO y el equipo de los mismos.

Para las comparaciones se utilizaron métricas especializadas en el contexto multi-objetivo, que serán introducidas más adelante en este capítulo. Finalmente se presentan los resultados experimentales obtenidos, de manera a poder obtener conclusiones empíricas acerca de los algoritmos comparados.

8.1. Configuración del entorno de ejecución

Todos los algoritmos fueron implementados en C++ (*Compilador GNU GCC*) y fueron ejecutados en un entorno Linux, distribución Mandrake 9, en máquinas con procesador AMD 1733 MHz con 512 MB de memoria. Se realizaron diez corridas de 200 segundos para cada algoritmo y para cada problema de prueba. Como problemas de prueba se utilizaron dos instancias de cada tipo de problema planteado en el capítulo 3 (TSP, QAP y VRPTW). En el caso del TSP se utilizaron las instancias bi-objetivas de 100 ciudades KROAB100 y KROAC100, que se encuentran en la distribución del TSPLIB[2]. Para el QAP bi-objetivo[3], se utilizaron las instancias de 75 localidades qapUni.75.0.1 y qapUni.75.p75.1 que pueden ser encontradas en [Sahni1976]. Para el VRPTW bi-

2 http://www-idss.cs.put.poznan.pl/~jaszkiewicz
3 http://www.intellektik.informatik.tu-darmstadt.de/~lpaquete/QAP

objetivo se utilizaron las instancias de 100 clientes c101 y rc101 tomados del conjunto de 56 problemas de prueba propuestos por Solomon [Solomon1987].

Para la implementación del *Team* se utilizó la librería de paso de mensajes *"Parallel Virtual Machine"* (PVM) versión 3.4.5, y las pruebas fueron ejecutadas en las máquinas arriba mencionadas conectadas mediante una red de computadoras. Los parámetros de configuración de los algoritmos se presentan en la tabla 8.1. Los parámetros que son aplicados exclusivamente a un algoritmo poseen entre paréntesis la indicación del algoritmo al cual corresponden.

Tabla 8.1. Parámetros de configuración de los algoritmos

Parámetro	**Valor**
Número de hormigas	10
α	1
β	2
γ (MOAQ)	0.3
ρ	0.1
λ (MOAQ)	0.8
τ_0	1
t_{max} (MOAQ)	0.9
O (MOA)	10
K (MOA)	100
K' (MAS)	500
q_0	0.5

De manera realizar una comparación justa entre los algoritmos, todos fueron ejecutados una cantidad igual de segundos, y por ende, tuvieron el mismo tiempo de ejecución para resolver las 6 instancias consideradas. Además, se utilizaron en todos los casos computadores homogéneos.

8.2. Métricas de desempeño

Se utilizaron 4 métricas de evaluación del desempeño de los algoritmos, de manera a realizar un análisis comparativo entre los mismos. Las métricas denominadas *M1'*, *M2'*, y *M3'*, tomadas de Zitzler *et al.* [Zitzler2000], se refieren respectivamente a la evaluación de la calidad, distribución y extensión del frente Pareto generado por el algoritmo. La cuarta métrica denominada *Error* fue

tomada de Veldhuizen [Veldhuizen1999] y se refiere al porcentaje de soluciones generadas que no pertenecen al frente Pareto.

En todos los casos se utiliza como métrica de distancia, la distancia euclidiana entre dos puntos, representada por $d(p,q)$, y se considera Y_{true} el frente Pareto conocido e Y' el frente Pareto generado por los algoritmos. Las cuatro métricas utilizadas se definen a continuación.

$$M1'(Y') = \frac{1}{|Y'|} \sum_{p \in Y'} min\{ d(p,q) \mid q \in Y_{true} \}$$

donde $|\cdot|$ representa cardinalidad.

Como puede notarse, la métrica *M1'* calcula el promedio de distancias entre cada punto del frente generado por algún algoritmo y el punto más cercano al mismo correspondiente al frente Pareto conocido. Podemos decir entonces, que calcula la distancia promedio del frente generado Y' al frente conocido Y_{true}. En general, será mejor el algoritmo cuyo frente calculado obtenga un menor valor en esta métrica.

La métrica *M2'* utiliza un conjunto $W_p = \{ q \in Y' \mid d(p,q) > \sigma \}$, dado un valor de distancia σ dependiente del problema, y se calcula como:

$$M2'(Y') = \frac{1}{|Y'| - 1} \sum_{p \in Y'} |W_p|$$

Esta métrica *M2'* se utiliza para calcular el número promedio de soluciones del frente generado por un algoritmo que cumplen con la condición de estar a una distancia mayor a σ, definido a priori, de alguna solución del mismo frente *Y'*. Mientras mayor sea el valor de esta métrica, mayor será la cantidad de soluciones distribuidas a una distancia mínima de σ, lo que implica un frente distribuido. Por lo tanto, es deseable un algoritmo cuyo frente generado posea un alto valor en esta métrica.

La métrica *M3'* calcula la extensión del frente Pareto generado según:

$$M3'(Y') = \sqrt{\sum_{i=1}^{b} max\{ d(p_i, q_i) \mid p, q \in Y' \}}$$

M3' se explica fácilmente en el contexto bi-objetivo, calcula la distancia entre las dos soluciones extremas del frente generado por un algoritmo. Se entiende por soluciones extremas, aquellas que

presentan el máximo valor para un objetivo particular con respecto a las demás soluciones del frente. La ecuación presentada para *M3*', es una generalización del concepto explicado para manejar b objetivos. Un valor elevado de esta métrica indica que se encontraron soluciones especializadas en cada objetivo lo que representa una buena exploración del espacio de soluciones del problema. En general, será mejor el algoritmo cuyo frente calculado obtenga una alta evaluación en esta métrica.

La métrica de *Error* se calcula como:

$$E = \frac{\sum_{i=1}^{|Y'|} e_i}{|Y'|}$$

donde e_i toma el valor 0 si la *i*-ésima solución de Y' pertenece al Y_{true}, 1 en caso contrario. Claramente esta métrica calcula el porcentaje de soluciones del frente generado por un algoritmo que no corresponde al frente Pareto conocido. Un valor alto de *Error*, indica una mayor cantidad de soluciones sub-óptimas.

Las métricas *M1'*, *M2'* y *M3'* fueron normalizadas con respecto a un valor máximo de las mismas, dependiendo del problema particular. De esta manera, los resultados obtenidos se representan como porcentajes que se utilizan para comparar los diversos algoritmos en el conjunto de problemas de prueba.

Para la métrica *M2'* se utilizó como valor para σ el diez por ciento de la distancia entre el punto con mejor evaluación en el primer objetivo y el punto con mejor evaluación en el segundo objetivo del frente Y_{true}, así σ representa el diez por ciento de la distancia entre los puntos extremos del Y_{true} para cada problema particular.

8.3. Resultados experimentales

El frente Y_{true} conocido de cada problema fue generado previamente tomando las soluciones no dominadas generadas por todos los algoritmos en todas las corridas [Pinto2005]. A continuación se presentan tablas que muestran los resultados de las evaluaciones de las métricas aplicadas a los frentes Pareto generados por los algoritmos antes descritos para cada problema. Además, se muestran gráficos correspondientes a los frentes Pareto generados por los distintos algoritmos para cada problema de prueba. Para los problemas del tipo VRPTW no se tomaron en cuenta las métricas

M2' y *M3'*, que evalúan la distribución y extensión del frente obtenido, debido a que los frentes Pareto encontrados contaban con pocas soluciones no dominadas.

8.3.1. Resultados para el TSP

En las tablas 8.2 y 8.3 se muestran los valores de las métricas aplicadas a los frentes generados por los algoritmos en ambas instancias bi-objetivas del TSP.

Tabla 8.2. *Ranking* ordenado de las métricas para el TSP KROAB100.

KROAB100							
M1'	**Alg.**	**M2'**	**Alg.**	**M3'**	**Alg.**	**Error**	**Alg.**
2,7	PACO	81,6	MOACS	100,0	COMP	92,7	PACO
2,9	TA	81,4	M3AS	98,3	MOAQ	93,9	MAS
4,6	MAS	81,1	MOA	94,2	MOACS	95,4	TA
5,9	M3AS	80,2	TA	93,5	MOA	99,5	M3AS
6,0	MOACS	77,7	MOAQ	93,0	TA	99,7	MOACS
6,9	MOA	77,2	MAS	93,0	M3AS	99,7	BIANT
13,6	BIMC	71,5	BIANT	88,2	MAS	100,0	MOA
15,4	BIANT	64,7	BIMC	69,2	BIANT	100,0	COMP
17,9	COMP	63,6	COMP	60,6	BIMC	100,0	BIMC
41,0	MOAQ	5,4	PACO	12,9	PACO	100,0	MOAQ

Tabla 8.3. *Ranking* ordenado de las métricas para el TSP KROAC100.

KROAC100							
M1'	**Alg.**	**M2'**	**Alg.**	**M3'**	**Alg.**	**Error**	**Alg.**
2,7	PACO	82,6	MOACS	100,0	COMP	90,8	PACO
3,2	TA	81,9	M3AS	99,0	MOAQ	95,7	TA
4,7	MAS	81,6	MOA	97,0	MOACS	96,2	MAS
6,5	M3AS	81,4	TA	93,9	MOA	98,5	M3AS
6,7	MOACS	78,7	MAS	93,5	TA	99,2	BIMC
7,6	MOA	76,4	MOAQ	89,7	M3AS	99,4	MOACS
13,4	BIMC	73,1	BIANT	88,6	MAS	99,8	BIANT
14,7	BIANT	64,0	BIMC	75,5	BIANT	100,0	MOA
20,7	COMP	63,7	COMP	63,0	BIMC	100,0	COMP
43,5	MOAQ	7,4	PACO	13,4	PACO	100,0	MOAQ

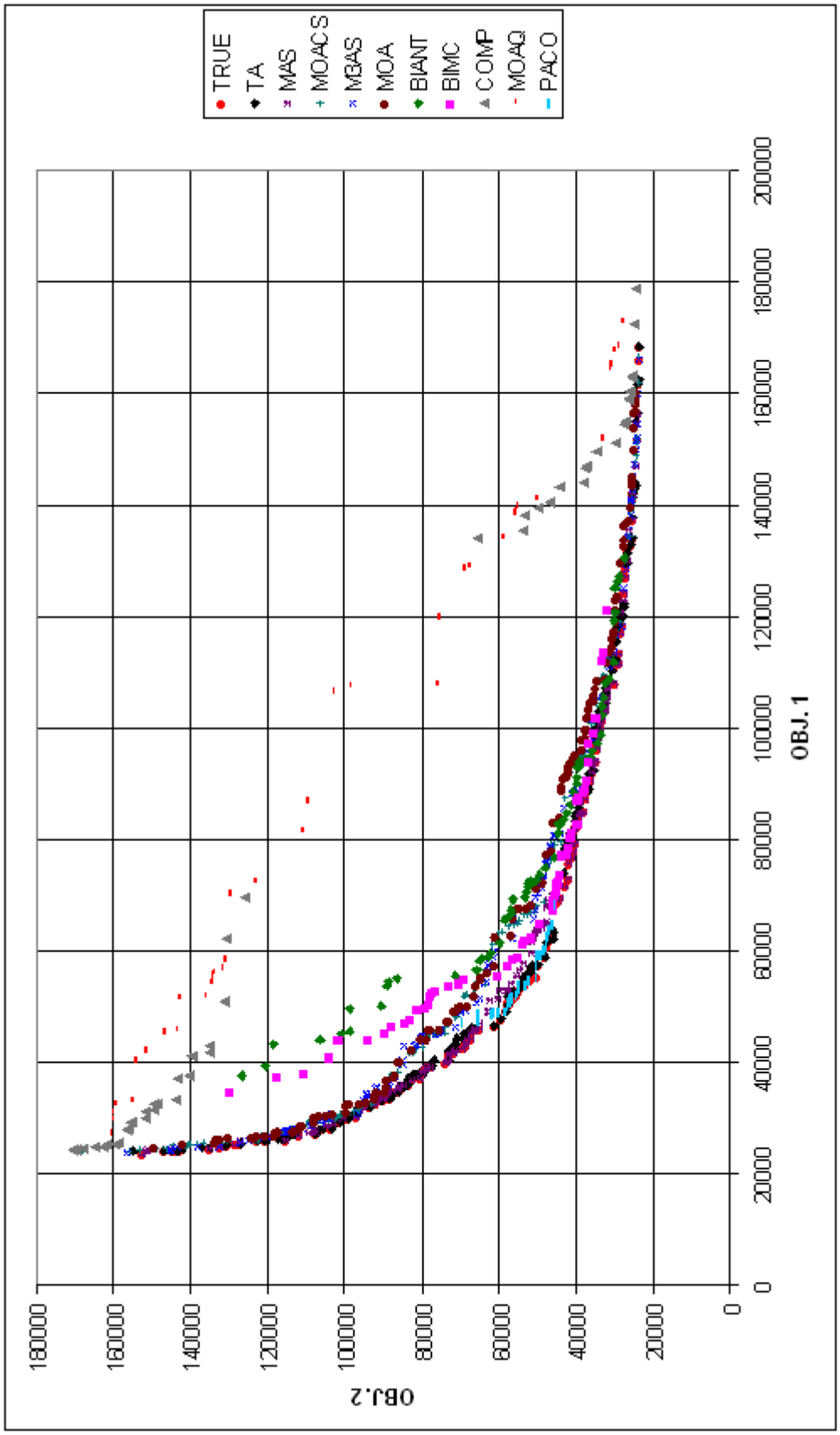

Figura 8.1. Frentes Pareto de los distintos algoritmos para el KROAB100.

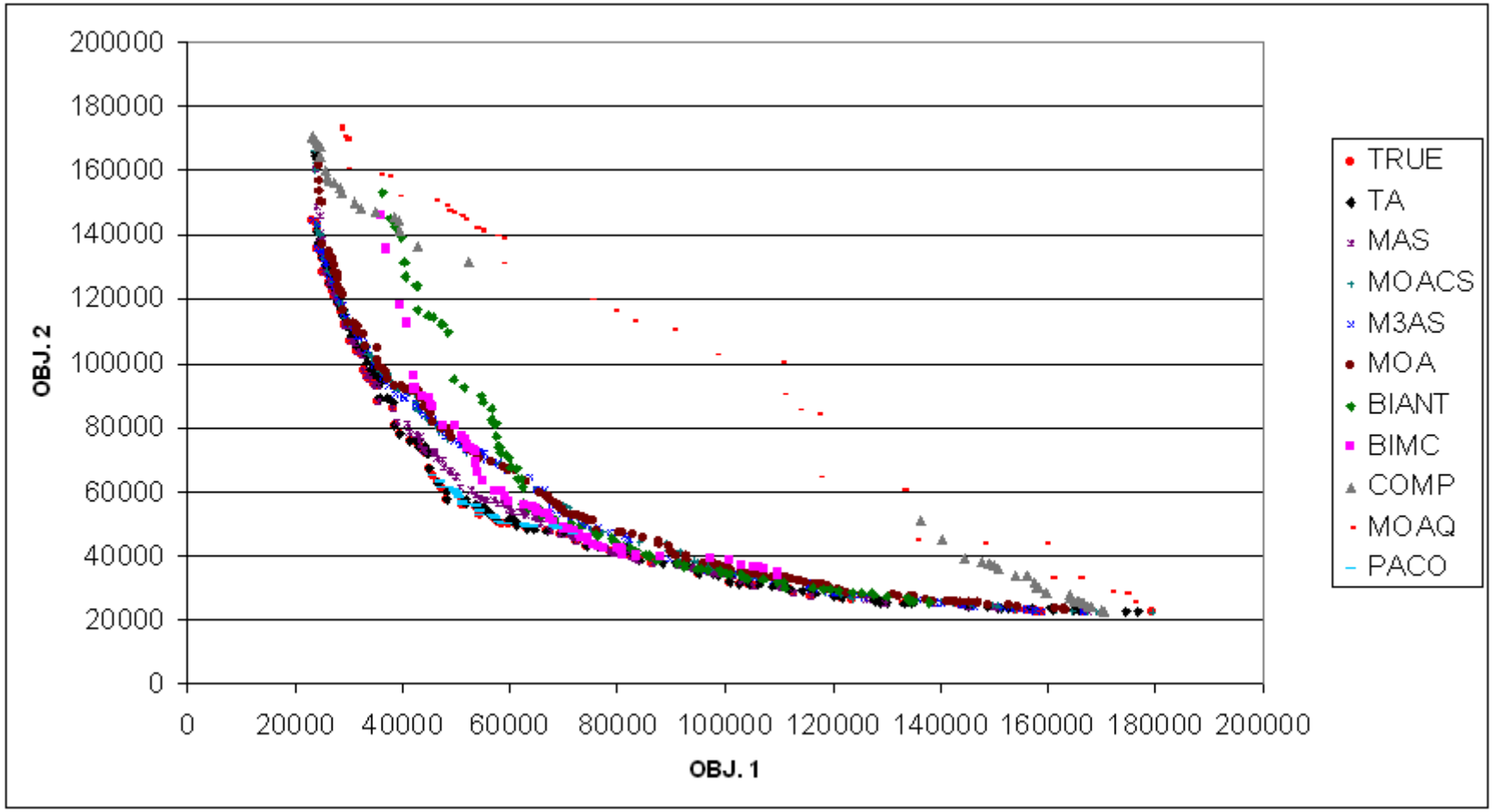

Figura 8.2. Frentes Pareto de los distintos algoritmos para el KROAC100.

Como se muestra en la tablas 8.2 y 8.3, el *Team Algorithm* (TA) presenta el segundo frente más cercano al frente Pareto conocido (*M1'*), solo superado por el PACO en pequeños sectores como muestran las métricas *M2'* y *M3'* donde el TA resultó muy superior al PACO. En las demás métricas el TA presenta valores bastante cercanos al primero en cada caso. En consecuencia, el TA para el TSP presenta un frente bien distribuido (*M2'*), razonablemente extenso (*M3'*) y en gran parte el más cercano al frente conocido, superando en principio a los demás algoritmos. En el caso de la extensión del frente medida por *M3'* todos los algoritmos que superan al TA presentan peores soluciones medidas por *M1'*.

Estos valores de las métricas pueden apreciarse en las figuras 8.1 y 8.2, que muestran los frentes Pareto generados por los algoritmos en ambos problemas TSP. En la figura 8.3 se presentan los frentes particulares del TA y el PACO para el KROAB100.

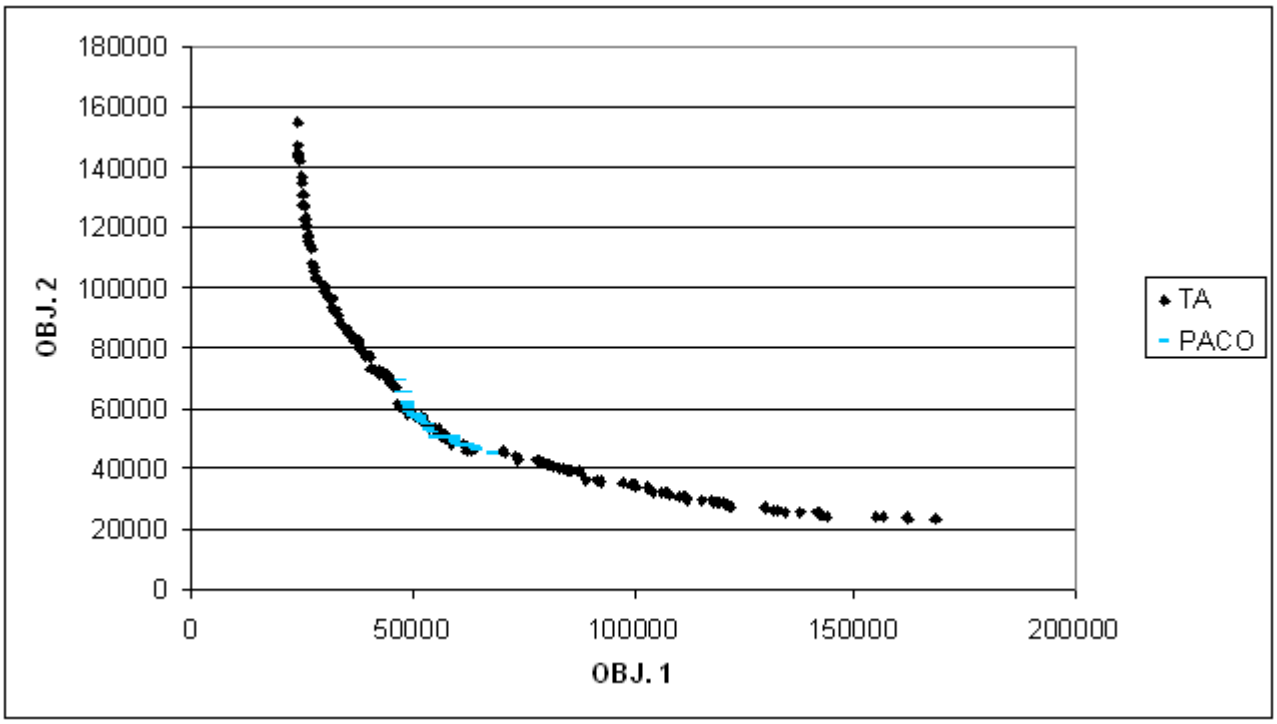

Figura 8.3. Frentes Pareto del TA y el PACO para el KROAB100.

Claramente, según la figura 8.3, el PACO representa solo un pequeño sector central del frente conocido, mientras el TA consigue las mejores soluciones. Resultados análogos se obtienen en el KROAC100 (ver figura 8.2). En la figura 8.4 comparamos el rendimiento del MAS propuesto en este trabajo contra el TA, que presentó el mejor comportamiento.

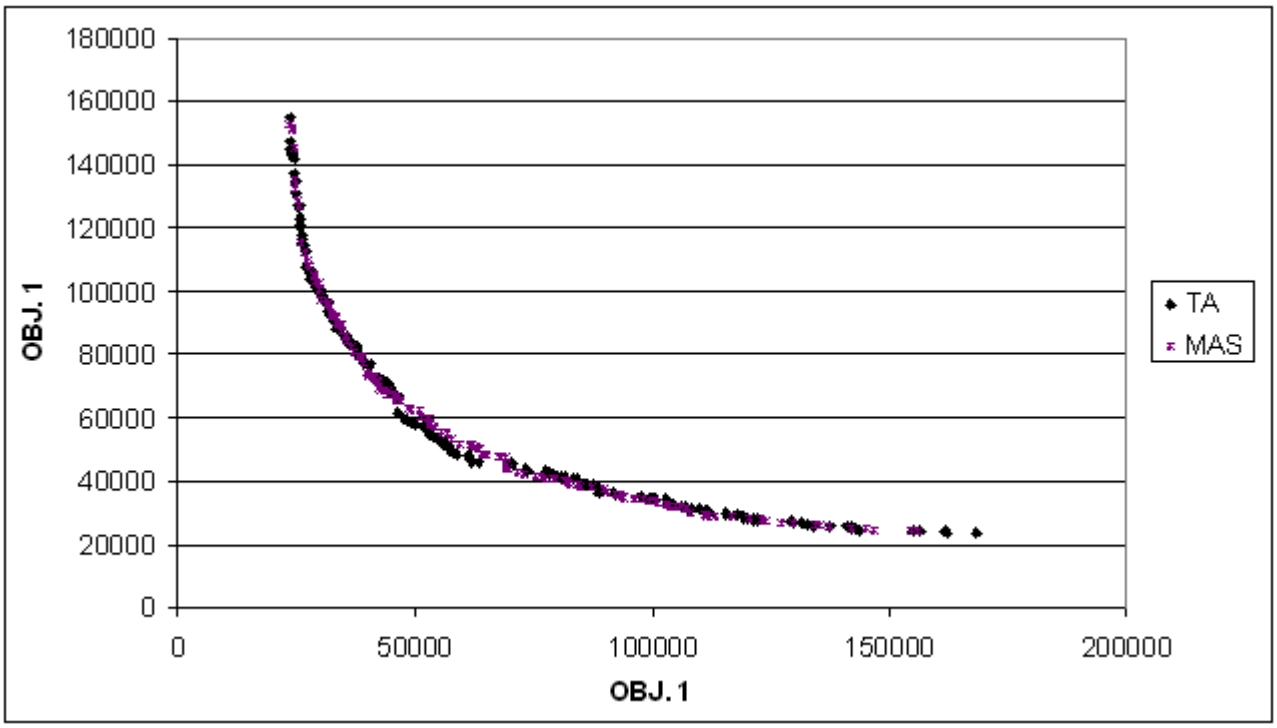

Figura 8.4. Frentes Pareto del TA y el MAS para el KROAB100.

Se puede notar en la figura 8.4 que el MAS presenta un comportamiento similar al TA, posee un frente bastante cercano al frente Pareto conocido y en promedio buena distribución y extensión. Análogamente, el MAS obtiene un buen rendimiento en el KROAC100. El MOACS y el M3AS presentan también buenos resultados en los problemas TSP.

Tabla 8.4. *Ranking* promedio para ambos problemas TSP en 20 corridas de cada algoritmo

RESUMEN TSP							
M1'	**Alg.**	**M2'**	**Alg.**	**M3'**	**Alg.**	**Error**	**Alg.**
2,7	PACO	82,1	MOACS	100,0	COMP	91,8	PACO
3,0	TA	81,6	M3AS	98,7	MOAQ	95,0	MAS
4,6	MAS	81,3	MOA	95,6	MOACS	95,5	TA
6,2	M3AS	80,8	TA	93,7	MOA	99,0	M3AS
6,4	MOACS	78,0	MAS	93,3	TA	99,5	MOACS
7,3	MOA	77,1	MOAQ	91,3	M3AS	99,6	BIMC
13,5	BIMC	72,3	BIANT	88,4	MAS	99,8	BIANT
15,1	BIANT	64,3	BIMC	72,3	BIANT	100,0	MOA
19,3	COMP	63,7	COMP	61,8	BIMC	100,0	COMP
42,3	MOAQ	6,4	PACO	13,1	PACO	100,0	MOAQ

En las figuras 8.1 y 8.2 se observa además que algunos algoritmos como el COMP y el MOAQ presentan frentes extensos y distribuidos pero muy lejanos al frente Pareto conocido, por ende se demuestra la importancia de la métrica *M1'*. Presentamos en la figura 8.5 este comportamiento mencionado del algoritmo COMP, comparado con el TA.

En la figura 8.5 se aprecia claramente que valores buenos para las métricas *M2'* y *M3'* no representan calidad en las soluciones obtenidas. En consecuencia, estas métricas deben ser tenidas en cuenta solo si el algoritmo presenta un buen valor de *M1'*, ya que si no fuera este el caso estaríamos midiendo distribución y extensión de un frente sub-óptimo, como el presentado por el COMP. En ambos problemas TSP, el COMP presenta un comportamiento similar.

Un *ranking* promedio de las métricas para el TSP puede observarse en la tabla 8.4. En el tabla promedio para el TSP se observan los resultados mencionados. De esta forma se aprecia el buen comportamiento del TA e inclusive del MAS (considerando que *M1*' mide la calidad de las soluciones del frente Pareto *Y'*).

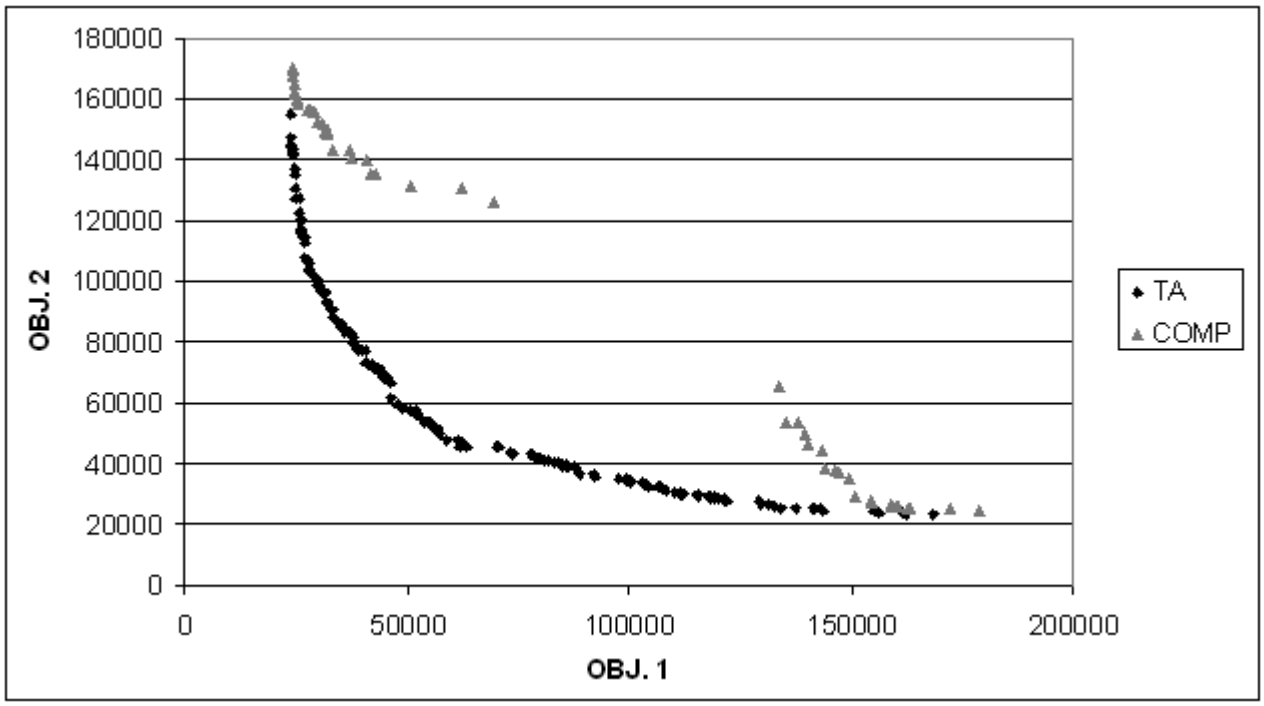

Figura 8.5. Frentes Pareto del TA y el COMP para el KROAB100.

8.3.2. Resultados para el QAP

En las tablas 8.5 y 8.6 se presenta el comportamiento de los algoritmos en ambas instancias del QAP. Además, se muestran los frentes Pareto generados por los algoritmos en las figuras 8.6 y 8.7. El *ranking* promedio de las métricas para el QAP se muestra en la tabla 8.7.

Tabla 8.5. *Ranking* de las métricas aplicadas a la instancia qapUni.75.0.l.

qapUni.75.0.1							
M1'	**Alg.**	**M2'**	**Alg.**	**M3'**	**Alg.**	**Error**	**Alg.**
16,3	TA	89,2	TA	90,6	TA	91,3	TA
17,4	MOACS	86,8	MOAQ	88,2	MOACS	94,3	MOACS
17,6	M3AS	86,6	COMP	81,4	MOA	97,3	M3AS
27,5	PACO	86,3	BIANT	79,8	M3AS	98,7	MOA
27,5	COMP	86,2	M3AS	75,4	COMP	99,0	BIANT
28,3	MOA	86,2	PACO	74,6	PACO	99,2	MAS
29,4	MAS	85,9	MOA	71,8	MOAQ	99,4	PACO
31,9	MOAQ	85,7	MOACS	71,1	BIANT	100,0	BIMC
38,0	BIMC	85,1	BIMC	70,2	BIMC	100,0	COMP
41,9	BIANT	81,7	MAS	67,9	MAS	100,0	MOAQ

Tabla 8.6. *Ranking* de las métricas aplicadas a la instancia qapUni.75.p75.l.

qapUni.75.p75.1							
M1'	**Alg.**	**M2'**	**Alg.**	**M3'**	**Alg.**	**Error**	**Alg.**
25,5	TA	31,6	PACO	11,7	MOAQ	80,0	MOACS
32,1	MOACS	23,3	BIMC	11,0	COMP	100,0	TA
36,0	M3AS	22,2	M3AS	10,9	PACO	100,0	MAS
44,4	MAS	15,2	BIANT	9,5	M3AS	100,0	MOA
45,5	MOA	14,1	COMP	9,3	MOACS	100,0	M3AS
46,2	COMP	13,0	MOAQ	8,2	BIANT	100,0	COMP
51,2	BIMC	5,0	TA	8,0	BIMC	100,0	BIANT
51,3	PACO	4,7	MOACS	7,0	MAS	100,0	BIMC
54,6	MOAQ	3,5	MAS	6,2	MOA	100,0	MOAQ
63,2	BIANT	1,3	MOA	5,8	TA	100,0	PACO

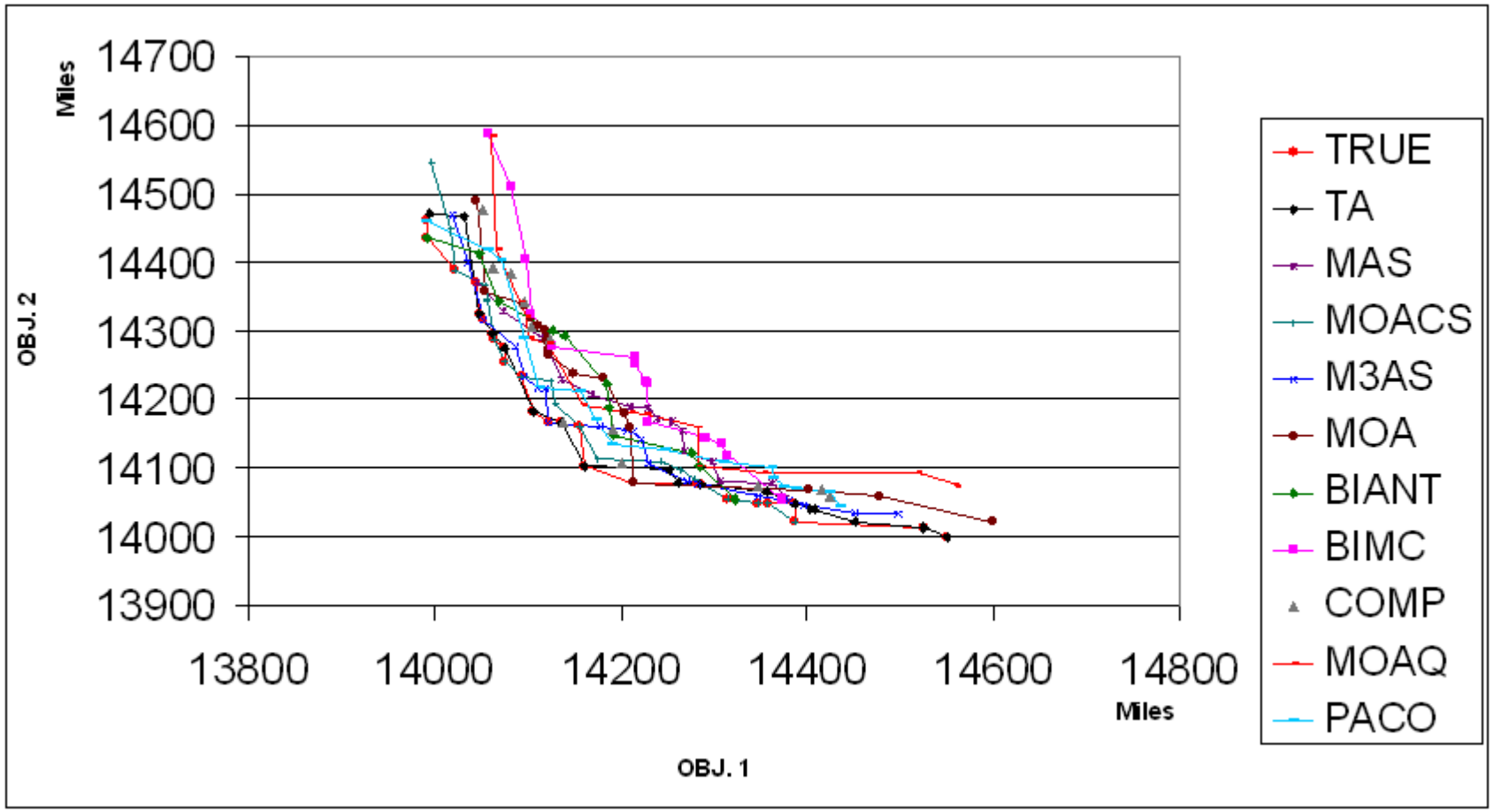

Figura 8.6. Frentes Pareto de los distintos algoritmos para el qap.Uni.75.0.1

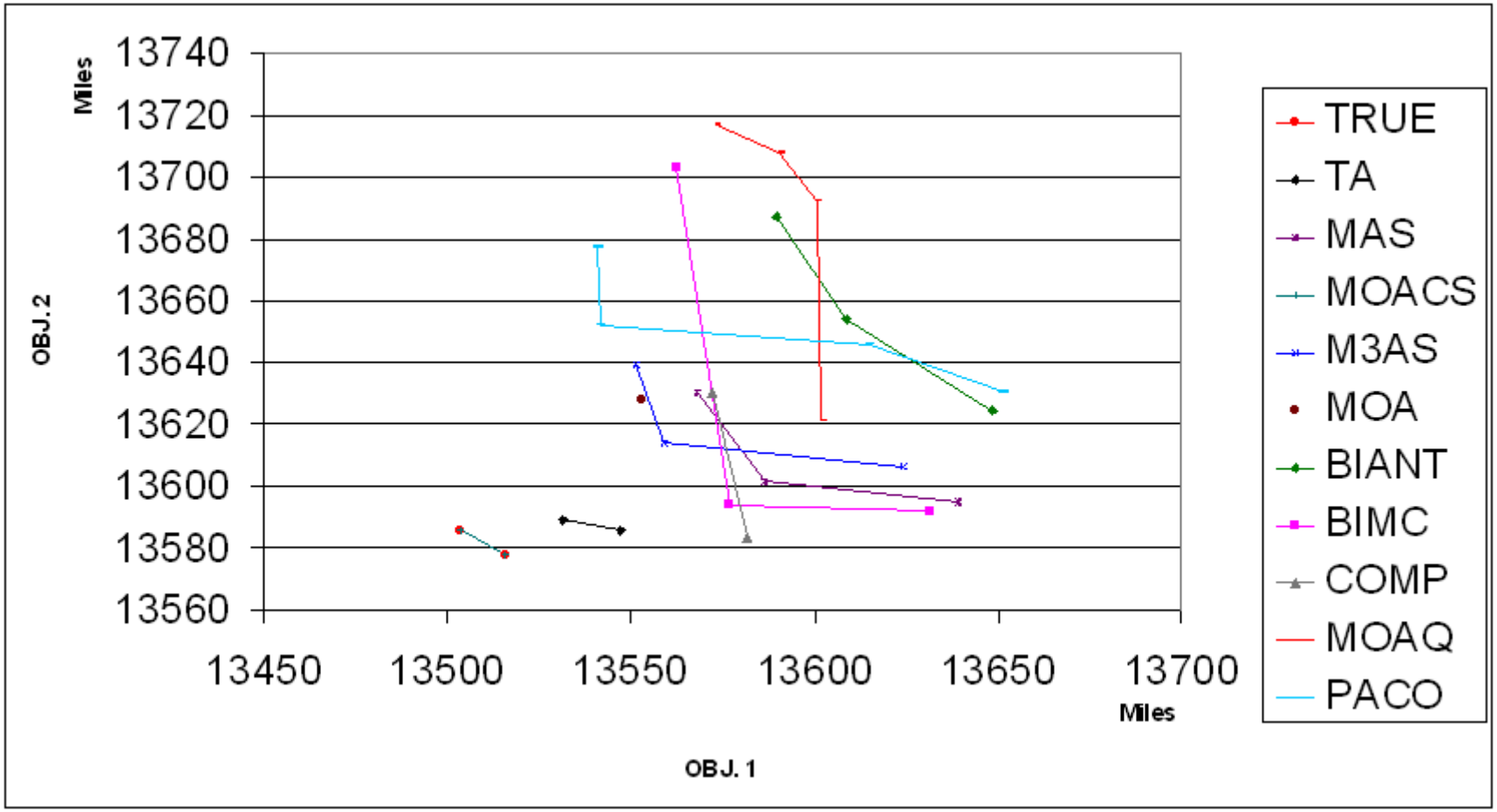

Figura 8.7. Frentes Pareto de los distintos algoritmos para el qap.Uni.75.p75.1

En la instancia qap.Uni.75.0.1 el TA superó a todos los algoritmos en todas las métricas, como se observa en la tabla 8.5. El MAS presentó un frente promedio cercano en distancia al frente Pareto conocido pero no obtuvo un buen comportamiento en extensión y distribución. El MOACS y el M3AS volvieron a presentar buenos valores promedios para las métricas.

En la figura 8.8 presentamos el frente generado por el TA para el qap.Uni.75.p75.1, comparado con el frente del MOACS, que representa el mejor encontrado.

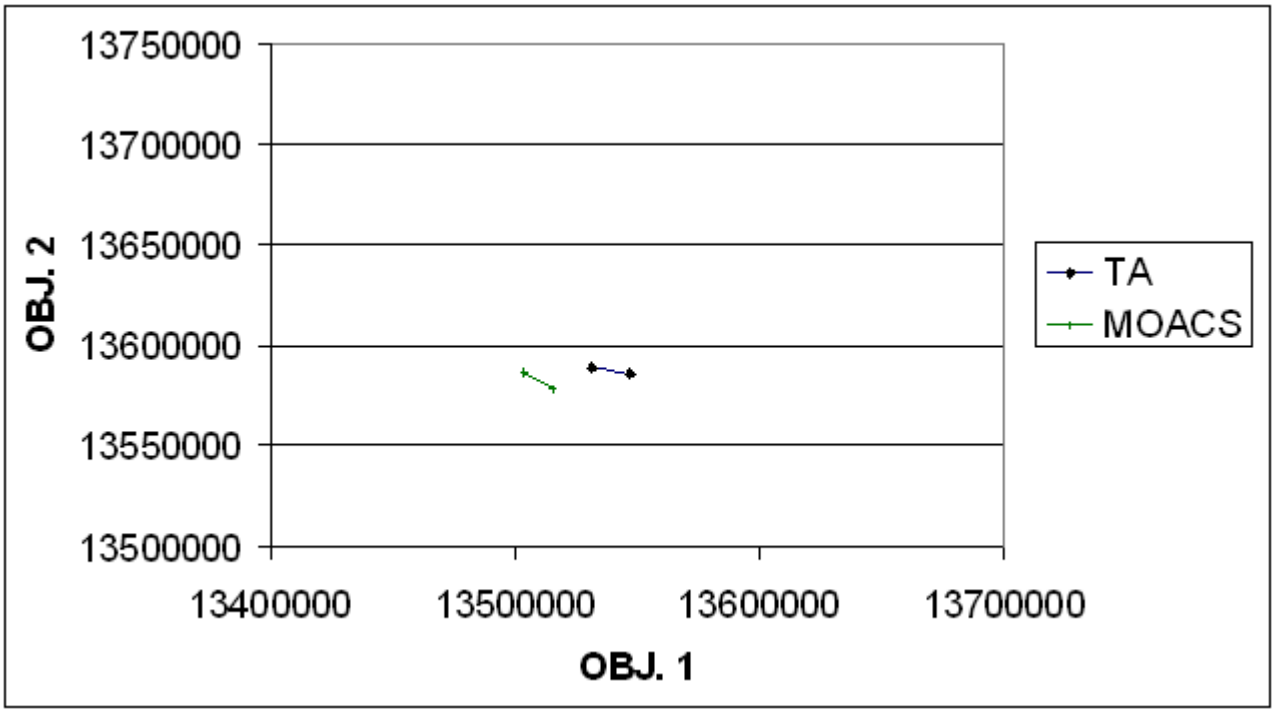

Figura 8.8. Frentes Pareto del TA y el MOACS para el qap.Uni.75.p75.1

Se puede notar en la figura 8.8, que el TA presenta un frente bastante cercano al del MOACS, y sus relativamente bajos valores en *M2'* y *M3'* se deben en realidad a la poca cantidad de soluciones encontradas, no obstante supera muchos algoritmos en la calidad de las soluciones encontradas.

En los problemas QAP (ver la tabla 8.7) el TA presenta un buen rendimiento general, posee la menor distancia al frente conocido y una buena distribución de soluciones (*M1' y M2'*). En las demás métricas, sus resultados son bastantes cercanos al mejor en cada caso. Para el QAP, el MOACS obtiene el comportamiento general más completo atendiendo a todas las métricas, pero al considerar la calidad de las soluciones, es superado por el TA.

Tabla 8.7. *Ranking* promedio de las métricas para el QAP en 20 corridas de cada algoritmo.

RESUMEN QAP							
M1'	**Alg.**	**M2'**	**Alg.**	**M3'**	**Alg.**	**Error**	**Alg.**
20,9	TA	69,5	MOAQ	48,8	MOACS	87,1	MOACS
24,8	MOACS	67,7	COMP	48,2	TA	95,6	TA
26,8	M3AS	65,8	PACO	44,7	M3AS	98,6	M3AS
36,9	MOA	63,4	BIMC	43,8	MOA	99,3	MOA
36,9	COMP	62,9	M3AS	43,2	COMP	99,5	BIANT
36,9	MAS	58,4	BIANT	42,8	PACO	99,6	MAS
39,4	PACO	57,4	MOACS	41,7	MOAQ	99,7	PACO
43,3	MOAQ	53,3	TA	39,6	BIANT	100,0	BIMC
44,6	BIMC	48,5	MOA	39,1	BIMC	100,0	COMP
52,5	BIANT	43,8	MAS	37,4	MAS	100,0	MOAQ

8.3.3. Resultados para el VRPTW

Para el caso del VRPTW, se presentan las tablas 8.8 y 8.9 con los valores de las métricas aplicadas a ambas instancias C101 y RC101. Los frentes Pareto generados se muestran en las figuras 8.9 y 8.10 y el promedio de las métricas para el VRPTW se presenta en la tabla 8.10. Considerando que el frente Pareto conocido presenta poca cantidad de soluciones, no serán presentadas las métricas de distribución (*M2'*) y extensión (*M3'*).

En la instancia C101, claramente el MAS y el TA obtienen los mejores resultados en ambas métricas consideradas. Además el BIANT obtiene buenas soluciones (tabla 8.8).

En la tabla 8.9 se observa nuevamente un buen rendimiento del BIANT y el TA para la instancia RC101 del VRPTW. El TA obtiene el frente más cercano al frente Pareto conocido. Además el MAS y el MOACS obtienen buenos resultados en distancia al frente conocido. Estos resultados también se observan las figuras 8.9 y 8.10 que presentan los frentes generados.

Puede apreciarse mediante las figuras 8.9 y 8.10 que los frentes Pareto generados por cada uno de los algoritmos cuentan, en efecto, con una escasa cantidad de soluciones. Debido a esto se

decidió no aplicar a las instancias de este problema las métricas *M2′* y *M3′* que representan distribución y extensión del frente Pareto generado, respectivamente.

Tabla 8.8. *Ranking* de las métricas para la instancia C101 del VRPTW.

C101			
M1'	**Algoritmo**	**Error**	**Algoritmo**
1,4	MAS	90,0	MAS
1,7	TA	90,0	TA
4,4	BIANT	90,0	BIANT
9,9	BIMC	100,0	MOACS
10,5	COMP	100,0	MOA
11,3	MOACS	100,0	M3AS
15,5	PACO	100,0	BIMC
25,7	M3AS	100,0	COMP
39,0	MOA	100,0	MOAQ
92,0	MOAQ	100,0	PACO

Tabla 8.9. *Ranking* de las métricas para la instancia RC101 del VRPTW.

RC101			
M1'	**Algoritmo**	**Error**	**Algoritmo**
10,2	TA	80,0	BIANT
10,4	BIANT	100,0	TA
11,4	MOACS	100,0	MAS
13,9	MAS	100,0	M3AS
21,0	BIMC	100,0	MOACS
22,0	COMP	100,0	MOA
29,4	PACO	100,0	COMP
48,3	M3AS	100,0	MOAQ
69,7	MOA	100,0	PACO
96,2	MOAQ	100,0	BIMC

En la tabla 8.10 de promedios para el VRPTW se mantienen los resultados ya citados, el TA, el BIANT y el MAS presentan los mejores resultados generales. En particular, se observa que el TA obtiene la menor distancia al frente Pareto conocido, mientras que el BIANT y el MAS presentan resultados similares y bastante cercanos al TA.

Podemos afirmar empíricamente, entonces, que ambos algoritmos propuestos en este trabajo, el TA y el MAS, presentan un buen rendimiento en estos problemas del tipo VRPTW. Considerando la importancia de este tipo de problemas en ingeniería y especialmente en logística, resulta interesante haber obtenido los mejores resultados mediante algoritmos originales propuestos en este trabajo.

Tabla 8.10. *Ranking* promedio de las métricas para el VRPTW en 20 corridas de cada algoritmo.

RESUMEN VRPTW			
M1'	**Algoritmo**	**Error**	**Algoritmo**
6,0	TA	85,0	BIANT
7,4	BIANT	95,0	TA
7,7	MAS	95,0	MAS
11,4	MOACS	100,0	M3AS
15,5	BIMC	100,0	MOACS
16,3	COMP	100,0	MOA
22,5	PACO	100,0	BIMC
37,0	M3AS	100,0	COMP
54,4	MOA	100,0	MOAQ
94,1	MOAQ	100,0	PACO

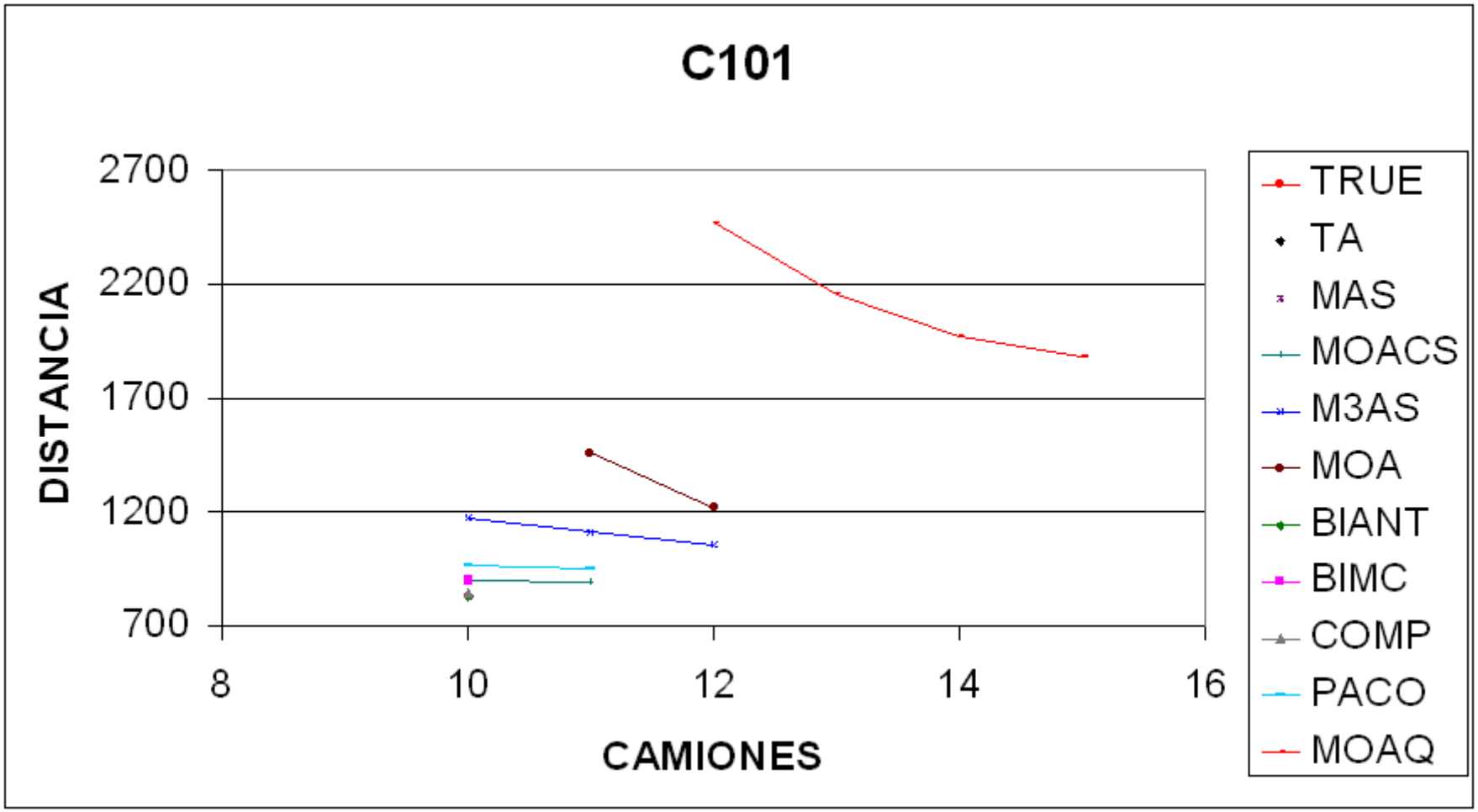

Figura 8.9. Frentes generados por los distintos algoritmos para la instancia C101.

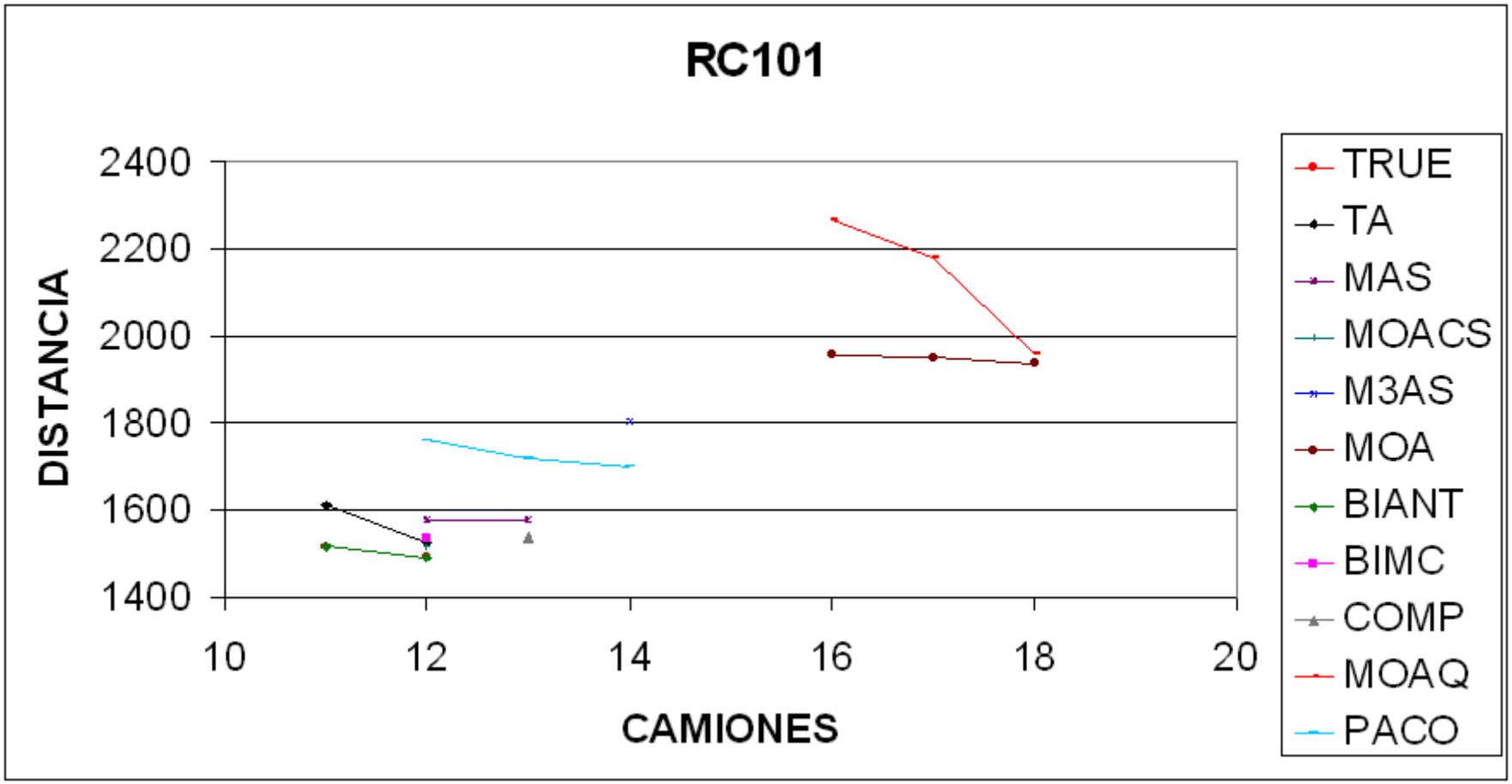

Figura 8.10. Frentes generados para la instancia RC101.

8.3.4. Resultados Generales

Con el objetivo de determinar el algoritmo con el mejor comportamiento general en todas las instancias de todos los problemas de prueba considerados, se presenta en la tabla 8.11 un promedio general atendiendo a las 60 corridas de todos los algoritmos en todos los problemas.

Claramente, se observa que el TA presenta la menor distancia al frente conocido (*M1'*) atendiendo todos los problemas considerados. En las demás métricas, el TA presenta muy buenos resultados considerando que son bastante cercanos al primero en cada caso. También puede notarse que el MOACS presenta un buen promedio general. El MAS, propuesto en este trabajo, obtiene el tercer lugar en distancia al frente conocido, lo cual es significativo considerando que supera a 7 algoritmos del estado del arte en la optimización multi-objetivo basada en colonia de hormigas al observar la calidad (*M1'*) de las soluciones obtenidas.

Considerando que el TA produce frentes extensos, distribuidos y además obtuvo las mejores soluciones (medidas por *M1'*), se demuestra empíricamente un buen comportamiento del mismo al compararlo con cada uno de los algoritmos implementados. Además, el TA presenta robustez ante distintos problemas manteniendo un buen rendimiento general. Esto puede explicarse por medio del efecto cooperativo del equipo, en el cual se aprovecha al mejor algoritmo para cada caso, manteniendo su rendimiento.

Tabla 8.11. Ranking promedio general en 60 corridas de cada algoritmo

RESUMEN GENERAL							
M1'	**Alg.**	**M2'**	**Alg.**	**M3'**	**Alg.**	**Error**	**Alg.**
10,0	TA	67,9	M3AS	72,2	MOACS	94,7	BIANT
14,2	MOACS	63,9	TA	71,6	COMP	95,4	TA
16,4	MAS	63,6	MOACS	70,7	TA	95,5	MOACS
21,5	PACO	63,5	MOAQ	70,2	MOAQ	96,5	MAS
23,3	M3AS	62,5	MOA	68,7	MOA	97,1	PACO
24,1	COMP	61,5	BIANT	68,0	M3AS	99,2	M3AS
24,5	BIMC	60,3	MAS	62,9	MAS	99,7	MOA
25,0	BIANT	59,3	BIMC	56,0	BIANT	99,8	BIMC
32,8	MOA	57,0	COMP	50,5	BIMC	100,0	COMP
59,9	MOAQ	32,6	PACO	27,9	PACO	100,0	MOAQ

La tabla 8.12 presenta la posición (*ranking*) promedio que obtuvo cada algoritmo considerando las 4 métricas de la tabla 8.11. Cabe resaltar que en la tabla 8.11 las columnas correspondientes a métricas con altos valores deseables (*M2' y M3'*) están ordenadas descendentemente, y aquellas columnas correspondientes a métricas con valores bajos deseables (*M1' y Error*) se encuentran ordenadas ascendentemente. La robustez y el buen comportamiento general se expresan en la tabla 8.12, en la cual el TA supera a todos los algoritmos comparados considerando el *ranking* final de promedios de todas las métricas utilizadas en este trabajo.

Tabla 8.12. Ranking final de promedios

RANKING FINAL	
2,00	**TA**
2,25	MOACS
4,75	M3AS
5,75	*MAS*
5,75	BIANT
6,50	COMP
6,75	MOA
7,00	PACO
7,00	MOAQ
7,25	BIMC

Empíricamente se demuestra que el MOACS produjo en promedio frentes Pareto bastante cercanos al frente conocido que los demás algoritmos, solo superado por el TA. En las demás métricas del promedio general, el MOACS obtuvo resultados considerablemente cercanos al primero en cada métrica y en extensión obtuvo el mejor resultado. Teniendo en cuenta su promedio en la métrica *M1',* se sugiere al MOACS como el mejor algoritmo en caso de no contar con los recursos computacionales para implementar el TA.

El algoritmo también propuesto en este trabajo, el MAS, obtuvo buenos resultados en los problemas específicos del tipo VRPTW y TSP. Esto puede observarse en los frentes generados por el mismo en cada caso. Atendiendo el *ranking* final de promedios supera a 6 algoritmos ACO multi-objetivos.

Capítulo 9

Conclusiones y Trabajos Futuros

Este trabajo presentó por primera vez en la literatura un equipo distribuido de algoritmos ACO multi-objetivos que aprovecha los recursos computacionales disponibles y distintas técnicas computacionales para la resolución de complejos problemas multi-objetivos.

Empíricamente se observó un buen comportamiento general del "*Team Algorithm"*, y se pudo demostrar su robustez en los diferentes problemas utilizados, manteniendo un buen rendimiento constante. En el caso de contar con los recursos computacionales, esta técnica puede ser considerada como una buena alternativa, y se puede aprovechar su efecto cooperativo al utilizar múltiples algoritmos y combinar sus soluciones. La tabla 8.12 muestra al TA como el mejor algoritmo general atendiendo los problemas considerados. Puede apreciarse en la tabla 8.11 que teniendo en cuenta la calidad, distribución y extensión del frente Pareto generado el TA presenta siempre buenas evaluaciones.

En el caso de tratar con problemas reales en los cuales el mejor algoritmo no es conocido, también se puede utilizar el equipo de algoritmos, ya que la cooperación entre los mismos combina soluciones y en base al criterio elitista de selección de algoritmos luego de un tiempo, termina encontrando no sólo buenas soluciones sino también al mejor algoritmo, que termina ejecutándose en todos los procesadores.

Atendiendo a los resultados generales en promedio, se puede notar que al considerar la distancia al frente óptimo (métrica *M1'*) los dos mejores algoritmos componentes del equipo implementan la estrategia de utilizar una única tabla de feromonas y varias visibilidades dependientes de cada objetivo a optimizar. Esto empíricamente demuestra que la estrategia de usar una única matriz de

feromonas con varias visibilidades obtiene un buen comportamiento con respecto a otras posibles implementaciones en la optimización multi-objetivo.

Se pudo observar que el MOACS tuvo un buen rendimiento general atendiendo a los problemas considerados, por lo que puede considerarse como una buena opción en caso de no disponerse de los recursos computacionales necesarios para utilizar un equipo distribuido de algoritmos.

Puede observarse también que el MAS propuesto en este trabajo presentó resultados aceptables, superando inclusive a varios algoritmos del estado del arte en la optimización multi-objetivo basada en colonias de hormigas.

Como trabajo futuro podría considerarse la extensión del equipo de algoritmos, de modo a incluir algoritmos basados en otras meta-heurísticas o técnicas tales como algoritmos basados en "*Particle Swarm Optimization"* [Kennedy1995], algoritmos evolutivos [Coello2002] o alguna otra técnica de interés. También se puede considerar la utilización de otros criterios de selección del mejor y peor algoritmo de cada iteración y verificar su comportamiento empírico. Además, se puede definir una estrategia en la cual los procesos esclavos reciban información del equipo de manera a compartir el conocimiento global adquirido.

Un interesante trabajo futuro es formular matemáticamente como un TSP, QAP o VRPTW los problemas reales mencionados en los apartados de aplicaciones prácticas del capítulo 3 y resolverlos utilizando datos reales. De esta forma se demostraría la aplicación práctica de los algoritmos propuestos en este trabajo. Además, se puede aumentar el conjunto de problemas de prueba considerando otros problemas multi-objetivos.

Dado que el MAS es una modificación sencilla del "*Ant System"* para resolver problemas multi-objetivos, también es interesante mejorar su implementación de modo a volverlo mas competitivo. Podrían agregarse estrategias específicas de optimización como optimizadores locales o mejorar la técnica de control de convergencia implementando un mecanismo más refinado que la simple reinicialización de feromonas que representan los conocimientos aprendidos. Considerando los resultados obtenidos en este trabajo, el MAS propuesto obtuvo un buen comportamiento general, por lo cual puede ser interesante seguir investigando su comportamiento.

Referencias

[Bailey2000] Bailey, C., McLain, T., Beard, R. (2000), Fuel Saving Strategies for Separated Spacecraft Interferometry, AIAA Guidance, Navigation, and Control Conference, Denver, CO, Paper no. AIAA-2000-4441:AA0-37143.

[Barán1995] Barán, B., Cáceres, N., Chaparro, E. (1995), Reducción del Tiempo de Búsqueda utilizando una Combinación de Algoritmos Genéticos y Métodos Numéricos. XV International Conference of the Chilean Computer Science Society. Arica - Chile.

[Barán1996] Barán, B., Kaszkurewicz, E., Bhaya, A. (1996), Parallel Asynchronous Team Algorithms: Convergence and Performance Análisis, IEEE Transactions on Parallel & Distributed Systems, Vol. 7, No. 7, pg. 677-688.

[Barán1998] Barán, B., Chaparro, E., Cáceres, N. (1998), A-Teams en la Optimización del Caudal Turbinado de una Represa Hidroeléctrica. IBERAMIA-98, Lisboa-Portugal.

[Barán2002] Barán, B. (2002), Parallel Asynchronous Team Algorithms, In: Correa, R., Dutra, I., Fiallos, M.: Models for Parallel and Distributed Computation: Theory, Algorithmic Techniques and Applications, Kluwer Academic Publishers.

[Barán2003] Barán, B., Schaerer, M. (2003), A multiobjective Ant Colony System for Vehicle Routing Problems with Time Windows, Proc. Twenty first IASTED International Conference on Applied Informatics, Insbruck, Austria, pg. 97-102.

[Benítez1996] Benítez, D., Ramos, R. (1996), Partición de Sistemas de lineales para su resolución en Sistemas Distribuidos, XXII Conferencia Latino - Americana de Informática (PANEL 96), Bogotá-Colombia.

[Bolondi1993] Bolondi, M., Bondanza, M. (1993), Parallelizzazione di un algoritmo per la risoluzione del problema del commesso viaggiatore, Master's thesis, Dipartimento di Elettronica e Informazione, Politecnico di Milano.

[Bullnheimer1997] Bullnheimer, B., Kotsis, G., Strauss, C. (1997),Parallelization Strategies for the Ant System, SFB Working Paper Nr 8.

[Bullnheimer1999] Bullnheimer, B., Hartl, R., Strauss, C. (1999), A new rank based version of the Ant System. A computational study, Central European Journal for operations Research and Economics, 7:1, 25-38.

[Çela1998] Çela, E. (1998), The Quadratic Assignment Problem. Theory and algorithms, Kluwer Academic Publishers.

[Coello1999] Coello, C. (1999), An updated Survey of Evolutionary Multiobjective Optimization Techniques, state of the art and future trends, In Congress on Evolutionary Computation. Piscataway, N. J., IEEE Service Center. 3–13.

[Coello2002] Coello, C., Mariano Romero, C. E. (2002), Evolutionary Algorithms and Multiple Objective Optimization, En M. Ehrgott y X. Gandibleux (Eds.), Multiple Criteria Optimization: State of the Art Annotated Bibliographic Surveys, pg. 277–331. Kluwer Academic Publishers, Boston.

[Deb1999] Deb, K. (1999), Evolutionary Algorithms for Multi-Criterion Optimization in Engineering Design, In Proceedings of Evolutionary Algorithms in Engineering and Computer Science EUROGEN'99.

[Denenbourg1990] Denenbourg, J., Aron, S., Goss, S., Pasteels, J. (1990), The self-organizing exploratory pattern of the argentine ant. Journal of Insect Behavior, 3, 159–168.

[Doerner2002] Doerner, K., Gutjahr, W., Hartl, R., Strauss, C., (2002), Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection, Proceedings of the 4th. Metaheuristics International Conference. Porto, 243-248.

[Doerner2003] Doerner, K., Hartl, R., Reimann, M., (2003), Are COMPETants more competent for problem solving? – the case of a multiple objective transportation problem, Central European Journal of Operations Research, 11:2, 115-141.

[Dorigo1996] Dorigo, M., Maniezzo, V., Colorni, A., (1996), The Ant System: Optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics - Part B, 26, 1, 29-41.

[Dorigo1997] Dorigo, M., Gambardella, L. (1997), Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation, 1:1, 53-66.

[Dorigo1998] Dorigo, M., Di Caro, G. (1999), Ant Algorithms for discrete optimization, Artificial Life, 5, 2, 137–172.

[Dorigo1999] Dorigo, M., Di Caro, G. (1999), The Ant Colony Optimization Meta-Heuristic, In D. Corne, M. Dorigo, and F. Glover (Eds.), New Ideas in Optimization, McGraw Hill, London, UK, 11-32.

[Fernández2005] Fernández, J., Barán, B. (2005): Equipo Elitista de Algoritmos Evolutivos Multiobjetivo. XXXI Conferencia Latinoamericana de Informática – CLEI'-2005. Cali – Colombia.

[Gambardella1995] Gambardella, L., Dorigo, M. (1995), Ant-Q: A reinforcement Learning approach to the traveling salesman problem, In A. Prieditis and S. Russell (Eds.), Proceedings of the Twelfth International Conference on Machine Learning (ML-95), 252–260. Morgan Kaufmann Publishers, Palo Alto, CA.

[Gambardella1999] Gambardella, L., Taillard, E., Agazzi, G., (1999), MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, In D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw-Hill, 73-76.

[García-Martínez2004] García-Martínez, C., Cordón, O., Herrera, F. (2004), An Empirical Análisis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria TSP. ANTS Workshop 61-72.

[García-Martínez2004a] García-Martínez, C., Cordón, O., Herrera, F. (2004), A Taxonomy and an empirical anlysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria TSP. Technical Report SCI2S-2004-12, Research Group on Soft Computing and Intelligent Information Systems, University of Granada.

[Gardel2005] Gardel, P., Barán, B., Estigarribia, H., Fernández, U., (2005), Aplicación del Ómicron ACO al problema de compensación de potencia reactiva en un contexto multiobjetivo, Congreso Argentino de Ciencias de la Computación - CACIC'2005. Concordia – Argentina.

[Gómez2004] Gómez, O., Barán, B., (2004), Omicron ACO, Proceedings of CLEI'2004. Latin-American Conference on Informatics (CLEI). Arequipa. Perú.

[Gómez2004a] Gómez O., Barán B., (2004), Reasons of ACO's Success in TSP. En M. Dorigo, M. Birattari, C. Blum, L. M. Gambardella, F. Mondada, y T. Stutzle, editores, Proceedings of ANTS 2004 - Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence, 3172 LNCS, Springer-Verlag, Brussels.

[Goss1989] Goss, S., Aron, S., Denenbourg, S, Pasteels, J., (1989), Self-organized shortcuts in the Argentine ant. Naturwissenschaften, 76, 579–581.

[Iredi2001] Iredi, S., Merkle, D., Middendorf, M., (2001), Bi-Criterion Optimization with Multi Colony Ant Algorithms, Proc. First International Conference on Evolutionary Multi-criterion Optimization (EMO'01), Lecture Notes in Computer Science 1993, 359-372.

[Kennedy1995] Kennedy, J., Eberhart, R. C. (1995), Particle Swarm Optimization, In Proceedings of the 1995 IEEE International Conference on Neural Networks, pages 1942–1948, Piscataway, New Jersey.

[Knowles2003] Knowles, J., Corne, D. (2003), Instance generators and test suites for the multiobjective quadratic assignment problem, In: Fonseca, C.M., *et al.* (Eds.) Proc of EMO '03, LNCS 2632 page 295-310, Springer-Verlag.

[Kumar2003] Kumar, V., Grama, A., Gupta, A., Karypis, G., (2003), Introduction to parallel computing, Second Edition, Addison Wesley.

[Lopez-Ibañez2004] Lopez-Ibañez, M., Paquete, L., Stutzle, T., (2004), On the design of ACO for the Biobjective Quadratic Assignment Problem, In: Dorigo, *et al.* (Eds.): Proc. of the Fourth International Workshop on Ant Colony Optimization (ANTS 2004), Lecture Notes in Computer Science, Springer Verlag.

[Maniezzo2004] Maniezzo, V., Gambardella, L., de Luigi, F., (2004), Ant Colony Optimization. In: Onwubolu, G. C., and B. V. Babu (Eds.): New Optimization Techniques in Engineering, Springer-Verlag, Berlin Heidelberg, pg. 101-117.

[Mariano1999] Mariano, C., Morales, E. (1999), A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks, Technical Report HC-9904, Instituto Mexicano de Tecnología del Agua, Mexico, June.

[Pinto2005] Pinto, D., Barán, B.(2005), Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimization approach. LANC'05, Cali, Colombia.

[Sahni1976] Sahni, S., González, T.(1976), P-complete aproximation problems. Journal of the ACM, 23, 555-565.

[Solomon1987] Solomon, M. (1987), Algorithms for the vehicle routing and scheduling problem with time window constraints, Operations Research, 35, 254-365.

[Souza1991] Souza, P., Talukdar, S. (1991), Genetics Algorithms in Asynchronous Teams, ICGA-91.

[Stutzle1998] Stutzle, T (1998), Parallelization strategies for ant colony optimization, In Eiben, A. *et al*. (Eds.) Proceedings of PPSN-V, Fifth International Conference on Parallel Problem Solving from Nature, 722-731, Springer-Verlag.

[Stutzle2000] Stutzle, T., Hoos, H. (2000), Max-Min Ant System, Future Generation Computer Systems, 16:8, 889-914.

[Talukdar1983] Talukdar, S., Pyo, S., Giras, T., (1983), Asynchronous Procedures for Parallel Processing, IEEE Trans. on PAS, Vol. PAS-102, NO 11.

[Talukdar1997] Talukdar, S., Baerentzen, L., Gove, A., de Souza, P., (1997), Asynchronous Teams: Cooperation Schemes for Autonomous Agents, Journal of Heuristics.

[Tarasewich2002] Tarasewich, P., McMullen, P. (2002), Swarm intelligence: power un numbers, Communications of the ACM, vol. 45, No. 8, 62-67.

[Zitzler1998] Zitzler, E., Thiele, L., (1998), An Evolutionary Algorithm for Multiobjective Optimization: The Strength Pareto Approach, TIK-report, No 43.

[Zitzler2000] Zitzler, E., Deb, K., Thiele, L., (2000), Comparison of multiobjective evolutionary algorithms. Empirical result, evolutionary computation. 8, 2, pp 173-195.

[Veldhuizen1999] van Veldhuizen, D. A. (1999). Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations. PhD thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio.