PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Tesis de Maestría
en Informática

Ant Colony Optimization para la resolución del Problema de Steiner Generalizado

Martín Pedemonte

Supervisor: Dr. Ing. Héctor Cancela

Tutor: Dr. Ing. Héctor Cancela

Montevideo, Uruguay
2009

Ant Colony Optimization para la resolución del Problema de Steiner Generalizado
Pedemonte, Martín
ISSN 0797-6410
Tesis de Maestría en Informática
Reporte Técnico RT 09-04
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República.
Montevideo, Uruguay, marzo de 2009

PEDECIBA INFORMÁTICA
Instituto de Computación - Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

TESIS DE MAESTRÍA EN INFORMÁTICA

Ant Colony Optimization para la resolución del Problema de Steiner Generalizado

Ing. Martín Pedemonte

Marzo 2009

Supervisor: Dr. Ing. Héctor Cancela
Tutor: Dr. Ing. Héctor Cancela

Ant Colony Optimization para la resolución del Problema de Steiner Generalizado

RESUMEN

Esta tesis presenta un estudio de la metaheurística Ant Colony Optimization (ACO) y de la aplicación de técnicas de computación de alto desempeño a dicha metaheurística. En particular, se aborda la aplicación de ACO a la resolución del Problema de Steiner Generalizado (GSP). El GSP consiste en el diseño de una subred de costo mínimo que verifique ciertos requerimientos prefijados de conexión entre pares de nodos distinguidos.

En el trabajo se presentan versiones ACO con dos enfoques constructivos de la solución distintos. El primero de los enfoques se basa en incorporar aristas hasta completar un camino, mientras que el segundo determina los K caminos más cortos y realiza una selección entre ellos. También se propone una novedosa formulación de un modelo celular aplicado a la metaheurística ACO y su posible paralelización.

Se incluye los resultados de un estudio experimental exhaustivo de todas las propuestas formuladas en este trabajo, comprendiendo la evaluación de los enfoques basados en aristas y en caminos y el análisis del efecto del tamaño de la población, de la cantidad de caminos y de incorporar operadores de búsqueda local para el enfoque basado en caminos. El estudio permitió comprobar que la utilización de un enfoque basado en caminos con la incorporación del operador de búsqueda local iterado obtiene resultados competitivos con las mejores técnicas disponibles en la actualidad. Asimismo, se evaluaron las versiones secuencial y paralela del modelo celular propuesto, constatándose que el desempeño computacional de la implementación paralela es muy promisoria, aunque se producen leves pérdidas en la calidad de las soluciones con relación a estructurar la población en la forma tradicional.

Palabras Clave: Ant Colony Optimization, Computación de alto desempeño, Problema de Steiner Generalizado, Metaheurísticas.

Agradecimientos

A mi tutor y supervisor de maestría Héctor Cancela por su guía académica y su aporte a este trabajo, y porque cuando atravesé una etapa en la que algunos problemas personales me impidieron dedicarme a la maestría, mantuvo la confianza en mí y en que este día llegaría.

A mis compañeros del Grupo CeCal: Sergio Nesmachnow, Pablo Ezzatti, Eduardo Fernández y Gerardo Ares, por aportarme otras ópticas, por los enriquecedores intercambios de ideas, por apoyar mi trabajo y también por los asados hasta altas horas de la noche.

A mis compañeros del Instituto de Computación: Alfredo Olivera, Antonio Mauttone y Pedro Piñeyro, por haber colaborado en la obtención de algunos artículos indispensables para que este trabajo fuera posible.

A mi familia, especialmente a mi madre Graciela y a mi hermano Gerardo por su apoyo continuo y por su cariño, ¡los quiero mucho!

A mis abuelos Abelardo y José porque su pasión por el conocimiento y la lectura siempre me sirven de inspiración.

A todos mis amigos, pero muy especialmente a Andrea, Pablo, Leandro y Lucía porque me ayudaron a navegar en la tormenta y porque sin ellos este día nunca habría llegado. A mis amigos Diego, Federico, Javier y Leonardo, porque hace media vida que estamos juntos, en las buenas y en la malas.

Y finalmente, a Leonella por su cariño y su paciencia, y por haberme ayudado con los últimos escalones. ¡Llegamos!

Índice general

Índice de figuras

Índice de algoritmos

Índice de tablas

Capítulo 1

Introducción

La resolución de problemas de optimización combinatoria del tipo NP-difíciles presenta dificultades en la práctica debido a que no se conocen, y muy probablemente no existan, algoritmos exactos para su resolución en tiempo polinomial. Ante esta dificultad, una alternativa es utilizar técnicas de resolución heurísticas. Este tipo de estrategias no garantizan obtener la solución óptima, pero encuentran soluciones cercanas al óptimo en forma eficiente.

Las metaheurísticas son esquemas generales de heurísticas que permiten abordar un amplio espectro de problemas adaptándose a sus particularidades. En los últimos años la investigación en metaheurísticas ha crecido en forma sustancial, sustentada en los buenos resultados obtenidos debidos a la aplicación de esas técnicas a problemas de optimización, formulándose una gran cantidad de propuestas de nuevas metaheurísticas. Sin embargo, en la práctica solamente un grupo pequeño de esas propuestas ha logrado consolidarse, demostrando una amplia aplicabilidad sobre problemas de diversas características y adquiriendo la madurez necesaria para ser una alternativa real al momento de resolver un problema de optimización.

Ant Colony Optimization (ACO) es una metaheurística cuya utilización para la resolución de problemas de optimización se ha consolidado en los últimos años. ACO es una metaheurística basada en población que unifica, bajo un esquema general, varias técnicas de resolución de problemas de optimización. Esta metaheurística se caracteriza por usar un conjunto de agentes (hormigas artificiales) para construir soluciones en forma incremental. Cada hormiga en forma concurrente, independiente y asíncrona construye incrementalmente una solución mediante la incorporación de componentes sobre una solución parcial. La incorporación de las componentes se realiza utilizando una regla probabilística que considera la experiencia adquirida en la búsqueda e información heurística de las componentes. Para incorporar la experiencia adquirida durante la búsqueda se utiliza una matriz de feromona, a modo de memoria que almacena el rastro depositado por las hormigas en la construcción de soluciones de buena calidad.

El tiempo de cómputo requerido para la resolución de problemas del tipo NP-difícil al utilizar metaheurísticas puede ser elevado, cuando las instancias resueltas tienen alta dimensionalidad. A partir de las crecientes posibilidades brindadas por las arquitecturas de hardware, las técnicas de alto desempeño constituyen una alternativa para la reducción del tiempo de ejecución de las metaheurísticas. Asimismo, las implementaciones paralelas y distribuidas suelen explorar el espacio de búsqueda de forma distinta a las implementaciones secuenciales, resultando en cambios en el comportamiento del algoritmo

que pueden provocar mejoras significativas en la calidad de las soluciones encontradas. En particular, el estudio de la aplicación de paralelismo sobre ACO es reciente y no ha completado su maduración. La investigación en esta temática ha crecido notablemente en los últimos cinco años, aunque las primeras propuestas de paralelización se remontan a los orígenes de la propia metaheurística.

El Problema de Steiner Generalizado (GSP, por sus siglas en inglés) modela el diseño de redes de comunicaciones confiables en las cuales se exigen requisitos de conexión tratando de garantizar con alta probabilidad la comunicación entre nodos distinguidos, llamados terminales. El GSP consiste en el diseño de una subred de mínimo costo que verifique una cierta cantidad de requerimientos prefijados de conexión entre pares de nodos distinguidos denominados terminales. Al utilizarse el GSP para el diseño de redes de comunicaciones se garantiza un cierto nivel de confiabilidad debido a la existencia de caminos alternativos entre nodos terminales. El funcionamiento de la red resultante es más robusto, siendo más resistente a fallas en sus componentes.

El GSP es un problema NP-difícil que ha sido poco abordado por la comunidad científica. En el Instituto de Computación (InCo) de la Facultad de Ingeniería de la Universidad de la Repúbica, se han aplicado exitosamente técnicas evolutivas para resolver el GSP con un enfoque en el cual se eliminan paulatinamente enlaces de la red original. Por otro lado, se han realizado propuestas caracterizadas por trabajar en forma constructiva, es decir agregando enlaces sobre una red originalmente vacía, que no han logrado obtener resultados de calidad similar.

Relacionando los conceptos anteriormente presentados, el trabajo que se presenta en esta Tesis de Maestría tuvo como objetivos el estudio de la metaheurística ACO y sus variantes, el estudio de las aplicaciones existentes de paralelismo a ACO y la propuesta e implementación de versiones secuenciales y paralelas de ACO aplicadas al caso concreto de la resolución del GSP. En particular, se buscaba desarrollar un algoritmo competitivo con los mejores algoritmos conocidos, pero que se caracterizara por trabajar en forma constructiva.

La estructura del resto del documento se describe a continuación.

El capítulo 2 es introductorio y sirve de sustento para el resto del trabajo, presentando nociones elementales vinculadas a los problemas de optimización combinatoria, la teoría de NP-completitud y las metaheurísticas.

Las principales características de la metaheurística ACO se comentan en el capítulo 3, incluyendo la descripción de su esquema general. Se presenta un amplio relevamiento de las variantes propuestas que instancian el esquema general, buscando explicar las diferentes características de cada una las versiones. El relevamiento tiene como eje central el estudio de propuestas formuladas para abordar problemas estáticos de optimización combinatoria.

El capítulo 4 introduce conceptos básicos de las técnicas de alto desempeño. Luego se comentan las taxonomías de estrategias para la aplicación de paralelismo a ACO, incluyendo una propuesta de ampliación sobre las taxonomías existentes. Finalmente, se presenta un relevamiento de la aplicación de paralelismo a ACO resumiendo las principales propuestas existentes en la literatura.

En el capítulo 5 se describe el problema considerado para la aplicación de la técnica ACO, presentándose el GSP, su modelo matemático y variantes del problema. También se incluye un relevamiento de los trabajos relacionados a la aplicación de técnicas metaheurísticas a la resolución del GSP. Finalmente, se presenta un resumen de trabajos

de ACO aplicado a problemas con características similares al GSP y que se consideran relevantes para contribuir al diseño del algoritmo para la resolución del GSP.

Las propuestas de aplicación de ACO a la resolución del GSP formuladas en este trabajo se presentan en el capítulo 6. Todas las propuestas formuladas se caracterizan porque cada hormiga construye una solución completa procesando iterativamente los requerimientos de conectividad entre los terminales. Dos enfoques se proponen para la construcción de caminos entre pares de terminales: incorporar aristas hasta completar el camino o determinar los K caminos más cortos para el par de terminales considerado y realizar la elección entre ellos; formulándose varias versiones para cada uno de los enfoques. Finalmente, se propone una novedosa formulación de un modelo celular para la metaheurística ACO y en particular su aplicación al GSP.

El capítulo 7 presenta la evaluación experimental de las propuestas de aplicación de ACO a la resolución del GSP presentadas en el capítulo 6, incluyendo la calibración y la evaluación de los enfoques basados en aristas y en caminos. La evaluación también considera el estudio del efecto del tamaño de la población, de la cantidad de caminos y de incorporar operadores de búsqueda local para el enfoque basado en caminos. Finalmente, se evalúa el nuevo modelo celular propuesto.

Por último, las conclusiones de esta tesis se presentan en el capítulo 8, conjuntamente con las líneas de trabajo futuro que se han identificado como continuadoras de este trabajo.

Capítulo 2

Metaheurísticas para problemas de optimización combinatoria

"Todo lo fácil fue difícil, todo lo chico fue gigante"
Árbol, *El campo sin fin* del disco *Hormigas*

Los problemas de optimización combinatoria consisten en la obtención de un elemento de un conjunto finito o infinito contable, que sea óptimo para la minimización o maximización de un cierto criterio. Desde el siglo pasado, ese tipo de problemas han resultado de interés para la comunidad científica por su alta aplicación a problemas reales y la simpleza de su formulación. El desarrollo de la computación ha contribuido considerablemente al progreso en la resolución de problemas de optimización combinatoria, al permitir implementar computacionalmente algoritmos para su solución.

Desde el auge de la computación, los avances concretados en el área de optimización combinatoria han sido significativos, imponiendo nuevos y mayores desafíos. Actualmente, una de las temáticas que despierta gran interés en la comunidad científica es el estudio de los problemas de optimización combinatoria del tipo NP-difíciles. Para los problemas NP-difíciles no se conocen, y muy probablemente no existan, algoritmos exactos para su resolución en tiempo polinomial. Ante esta dificultad, una alternativa es utilizar técnicas de solución heurísticas que, aunque no garantizan obtener la solución óptima de un problema, encuentran soluciones cercanas al óptimo en forma eficiente. Las metaheurísticas son esquemas generales de heurísticas que permiten abordar un amplio espectro de problemas adaptándose a las particularidades de cada problema. En los últimos años, la investigación en metaheurísticas ha crecido en forma sustancial sustentada en los buenos resultados obtenidos como resultado de la aplicación de esas técnicas a problemas de optimización.

En este capítulo se presentan las nociones elementales de optimización combinatoria, teoría de NP-completitud y metaheurísticas que sirven de sustento para el resto de este trabajo. La estructura del capítulo es la siguiente. En la sección 2.1 se presenta la definición de los problemas de optimización combinatoria. Posteriormente, en la sección 2.2 se introduce la formalización de los problemas NP-difíciles. La definición de metaheurística y algunas taxonomías propuestas para metaheurísticas se comentan en la sección 2.3. Finalmente, las conclusiones de este capítulo se presentan en la sección 2.4.

2.1. Optimización combinatoria

Los problemas de optimización consisten en la obtención de un elemento óptimo de un conjunto de soluciones posibles. El elemento se considera óptimo en el sentido que satisface un cierto criterio u objetivo. Para poder medir la satisfacción del objetivo se introduce la noción de función objetivo a ser minimizada o maximizada.

Formalmente, los problemas de optimización se corresponden con la tupla $(\mathcal{S},f,\Omega)$ siendo $\mathcal{S}$ el espacio de soluciones, f la función objetivo que asigna valores $f(s)$ a $s \in \mathcal{S}$ y Ω un conjunto de restricciones. El conjunto de las soluciones factibles $\mathcal{S}_\Omega$ está formado por aquellas soluciones que cumplen las restricciones Ω. Los problemas de optimización pueden ser de minimización o maximización de la función objetivo. Los problemas de optimización que consisten en una maximización pueden ser transformados fácilmente en una minimización negando la función objetivo. Resolver un problema de optimización, en el caso de una minimización[1], corresponde a encontrar una solución factible $s^* \in \mathcal{S}_\Omega$ tal que: $f(s^*) \leq f(s), \forall s \in \mathcal{S}_\Omega$.

Es posible que el problema de optimización no tenga una única función objetivo sino varias, resultando así los problemas de optimización multiobjetivo. Ese tipo de problemas suelen implicar el balance entre intereses contrapuestos.

Las variables de decisión asociadas a las soluciones pueden tomar valores reales o discretos. Cuando los valores son reales resultan los problemas de optimización de dominio continuo. Cuando los valores son discretos resultan los problemas de optimización de dominio discreto. Los problemas de optimización combinatoria están comprendidos dentro de los de dominio discreto y corresponden a encontrar un objeto dentro de un conjunto finito o infinito contable [21].

La formalización del problema de optimización combinatoria $P = (\mathcal{S},f,\Omega)$, siguiendo la notación usada por Blum y Dorigo [20], es la siguiente:

- X_i es un conjunto de variables discretas con los valores $x_i \in D_i = \{d_1^i, \ldots, d_{|D_i|}^i\}$, $i = 1, \ldots, n$;
- Ω es un conjunto de restricciones entre las variables;
- la función objetivo del problema a minimizar es $f : D_1 \times \ldots \times D_n \to \mathbb{R}$.

El espacio de búsqueda $\mathcal{S}_\Omega$ se define por $\{s = \{(X_1, x_1), \ldots, (X_n, x_n)\} | x_i \in D_i, s$ satisface las restricciones del problema $(\Omega)\}$. $\mathcal{S}_\Omega$ es el conjunto de todas las posibles asignaciones de valores a variables que cumplen con las restricciones de factibilidad y s es una solución al problema de optimización combinatoria. Resolver el problema de optimización combinatoria consiste en encontrar una solución factible $s^* \in \mathcal{S}_\Omega$ tal que: $f(s^*) \leq f(s), \forall s \in \mathcal{S}_\Omega$. Se define a s^* como una solución óptima global del problema P.

Otros conceptos vinculados a la resolución de problemas de optimización son vecindad y mínimo local cuya definición es la siguiente:

- Vecindad: es una función $\mathcal{N} : \mathcal{S} \to 2^{\mathcal{S}}$ que le asigna a cada $s \in \mathcal{S}$ un conjunto de soluciones vecinas ($\mathcal{N}(s) \subseteq \mathcal{S}$).
- Mínimo local: $\hat{s}$ es un mínimo local si $f(\hat{s}) \leq f(s), \forall s \in \mathcal{N}(\hat{s})$.

[1] De aquí en más, al referirse a un problema de optimización se asume el caso de una minimización.

Los problemas de optimización pueden ser estáticos o dinámicos. Los estáticos se caracterizan por estar completamente definidos a priori, conociéndose el dominio de las variables de decisión, el conjunto de restricciones y la función objetivo antes de su resolución y manteniéndose inalteradas durante la resolución del problema. Por el contrario, los problemas dinámicos presentan características que se modifican mientras que está siendo resuelto. Las modificaciones pueden incluir cambios en los dominios de las variables de decisión, en el conjunto de restricciones y/o en la función objetivo.

2.2. Problemas NP-difíciles

Para evaluar la complejidad computacional de un algoritmo se utiliza el número de instrucciones o de operaciones en función del tamaño de la entrada. De esa forma, el análisis es independiente de la arquitectura del computador y del lenguaje de programación utilizados. Se suele analizar la complejidad para el peor caso, de forma de obtener una cota superior asintótica. Formalmente si $f, g : \mathbb{N}^+ \rightarrow \mathbb{R}^+$ se dice que la función $f(x)$ tiene orden $O(g(x))$, si existen x_0 y $c > 0$ tales que $0 \leq f(x) \leq c.g(x), \forall x \geq x_0$. Los algoritmos que tienen tiempo polinomial son aquellos cuya complejidad en el peor caso tiene $O(g(x))$ para alguna función g polinomial.

La teoría de NP-completitud [83] distingue dos clases: la clase P, que contiene los problemas que pueden ser resueltos en tiempo polinomial en una computadora determinística y la clase NP, que contiene los problemas que pueden ser resueltos en tiempo polinomial en una computadora no determinística. En forma equivalente, la clase NP puede ser conceptualizada como verificable en tiempo polinomial, es decir que se puede comprobar para todas las instancias si son solución al problema o no en tiempo polinomial. A partir de las definiciones es posible concluir que P $\subseteq$ NP. Sin embargo, la pregunta ¿es P = NP? no ha podido ser respondida en más de 30 años. Continúa siendo una pregunta abierta aunque se cree que P $\neq$ NP, constituyendo la principal conjetura de la teoría de la computación en la actualidad [45].

Se define a un problema como NP-difícil, si todo problema en NP puede ser transformado en él mediante un algoritmo de tiempo polinomial. A partir de la definición resulta natural que los problemas NP-difíciles son por lo menos tan difíciles como cualquier problema en NP. Sin embargo, es conveniente remarcar que los problemas NP-difíciles no tienen porque pertenecer a NP. Para los problemas que son NP-difíciles y pertenecen a NP se utiliza el término NP-completo. Evidentemente, si se pudiera encontrar un algoritmo de tiempo polinomial para resolver un problema NP-completo, todos los problemas NP-completos podrían ser resueltos en tiempo polinomial a través de las transformaciones. En ese caso, se podría concluir que P = NP. En la figura 2.1 se muestra la relación entre las clases de problemas P, NP y NP-completos si P $\neq$ NP.

La pertenencia de un problema a la clase P, es decir la existencia de un algoritmo de tiempo polinomial para su resolución, se asocia a la posibilidad de ser resuelto en la práctica. En contraposición, la pertenencia de un problema a la clase NP-difícil y por tanto la inexistencia de un algoritmo que lo resuelva en tiempo polinomial, se asocia a la imposibilidad de ser resuelto en tiempo razonable al considerarse instancias de tamaño creciente.

Un compendio actualizado de problemas NP-difíciles se encuentra disponible en la página web [45].

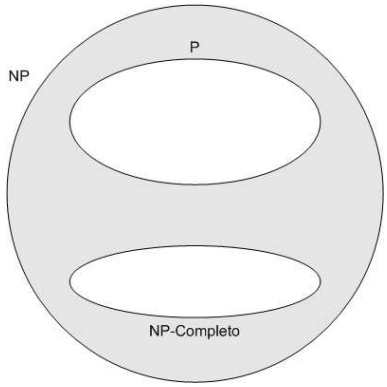

Figura 2.1: Relación entre las clases de problemas P, NP y NP-completo (si P $\neq$ NP).

2.3. Metaheurísticas

Afrontar un problema de optimización combinatoria con una técnica exacta garantiza que la solución obtenida sea la óptima. Sin embargo, en caso de tratarse de un problema NP-difícil, el algoritmo tendrá un tiempo no polinomial, lo cual puede volver prohibitiva su utilización al considerarse instancias de tamaño creciente del problema. En contraposición, las técnicas de solución heurísticas no garantizan la obtención del óptimo, pero pueden encontrar soluciones casi-óptimas en forma eficiente. Esta característica vuelve a las técnicas heurísticas muy atractivas al momento de afrontar un problema de optimización combinatoria del tipo NP-difícil.

Es posible diseñar heurísticas específicas para cada problema o esquemas generales que permitan abordar múltiples problemas. En general, la especificidad permite obtener mejores resultados sacrificando la simpleza y la adaptabilidad. No existe en la comunidad científica un consenso sobre la definición exacta de la palabra metaheurística. El término fue acuñado por Glover en 1986 para referirse a "heurísticas con un nivel más alto de abstracción"[84]. La definición más satisfactoria y comúnmente usada es considerar a las metaheurísticas como un conjunto de patrones empleados para definir métodos heurísticos que pueden ser aplicados a una amplia gama de problemas. Es posible interpretar a las metaheurísticas como un patrón general de algoritmos que pueden ser aplicados a diferentes problemas de optimización con relativamente pocas modificaciones para su adaptación a las particularidades del problema considerado [2].

Las metaheurísticas se caracterizan por:

- ser estrategias generales que guían el proceso de búsqueda, pudiendo incorporar información específica del problema e inclusive incorporar heurísticas subordinadas.
- ser de uso genérico, debiendo ser instanciadas para cada problema particular.
- permitir una descripción a nivel abstracto de sus componentes básicos.
- explorar eficientemente el espacio de búsqueda con el objetivo de encontrar soluciones factibles de buena calidad (con valores de la función objetivo cercanos al óptimo).
- recorrer el espacio de búsqueda sin quedar atrapados en regiones particulares, especialmente en cercanías de óptimos locales.

- ser algoritmos aproximados y en general no determinísticos (estocásticos).

Para la resolución de un problema mediante la aplicación de una técnica metaheurística resulta fundamental un balance entre la diversificación y la intensificación, entendiéndose por diversificación la exploración del espacio de búsqueda y por intensificación la explotación de la experiencia acumulada a través del proceso de búsqueda [21].

Debido al aumento en el interés en la aplicación de metaheurísticas para la resolución de problemas de optimización combinatoria del tipo NP-difícil, se han propuesto una gran cantidad de nuevas metaheurísticas. Sin embargo, solamente un grupo pequeño de esas propuestas ha logrado consolidarse en la práctica, demostrando una amplia aplicabilidad sobre problemas de muy diversas características y adquiriendo la madurez necesaria como técnica para ser una alternativa real al momento de resolver un problema de optimización.

Algunas de las técnicas más populares son: Iterated Local Search (ILS) [108], Guided Local Search (GLS) [165], Simulated Annealing (SA) [104], Tabu Search (TS) [84], Variable Neighbourhood Search (VNS) [122], Greedy Randomized Adaptive Search Procedure (GRASP) [74], la familia de técnicas Evolutionary Computation (EC) [77, 86, 142] y Ant Colony Optimization (ACO) [64].

2.3.1. Clasificaciones

La comunidad académica ha planteado varias clasificaciones que agrupan las técnicas metaheurísticas de acuerdo a ciertas características consideradas [21, 118].

Uno de los criterios de clasificación propuestos consiste en si la fuente de inspiración de la técnica proviene de la naturaleza o no. Es un criterio intuitivo aunque pueden existir dificultades para clasificar algunas técnicas. SA, EC y ACO son técnicas inspiradas en la naturaleza.

Otro posible criterio distingue entre las técnicas aleatorias o determinísticas. Originalmente, varias técnicas eran determinísticas (por ejemplo: TS y GLS) pero paulatinamente han surgido propuestas que han ido incorporando elementos aleatorios sobre esas técnicas, volviendo al criterio poco útil en la práctica.

La utilización o no de memoria es otro posible criterio de clasificación. Las técnicas pueden incorporar mecanismos de memoria a corto plazo o a largo plazo. La memoria a corto plazo suele utilizarse para reconocer soluciones recientemente visitadas. En contraposición, la memoria a largo plazo suele acumular la experiencia global adquirida durante la búsqueda. Técnicas que utilizan memoria son TS y ACO.

Otros posibles criterios de clasificación distinguen entre técnicas basadas en trayectoria o poblacionales, y técnicas de búsqueda basadas en instancias o en modelos. Ambos criterios se comentan a continuación.

Métodos basados en trayectoria o en población

Esta taxonomía para metaheurísticas es la que más comúnmente se utiliza y distingue entre las técnicas basadas en trayectoria y las poblacionales o basadas en población [21, 118].

La primera clase de técnicas se caracteriza por trabajar con una única solución en cada paso de la búsqueda y suele estar fuertemente vinculada a la utilización de operadores de búsqueda local. Las técnicas incluidas en la primera clase utilizan un

mecanismo de exploración caracterizado por recorrer una única trayectoria en el espacio de búsqueda. ILS, GLS, SA, TS, VNS y GRASP son técnicas basadas en trayectoria.

La segunda clase de técnicas se caracteriza por utilizar un conjunto de soluciones (población) en cada paso de la búsqueda en lugar de una única solución. Las técnicas incluidas en la segunda clase utilizan un mecanismo de exploración caracterizado por recorrer múltiples trayectorias en el espacio de búsqueda. EC y ACO son técnicas basadas en población.

Búsqueda basada en instancia o en modelo

Esta taxonomía es reciente y tiene una fuerte vinculación con el área de aprendizaje automático [174, 175]. La característica que se considera para esta clasificación es el tipo de información utilizada para explorar el espacio de búsqueda, distinguiendo entre la búsqueda basada en instancia y la búsqueda basada en modelo.

Las búsquedas basadas en instancia solamente utilizan la solución actual o la población actual de soluciones para generar nuevas soluciones. La información adquirida sobre el espacio de búsqueda durante el proceso de búsqueda no se maneja en forma explícita, sino que está contenida en forma implícita en la solución o las soluciones que utiliza la técnica. ILS, GLS, SA, TS, VNS, GRASP y la mayoría de los métodos de EC son técnicas de búsqueda basadas en instancia.

Las búsquedas basadas en modelo utilizan un modelo probabilístico parametrizado que pretende reflejar las mejores regiones del espacio de búsqueda para generar nuevas soluciones. La información adquirida sobre el espacio de búsqueda se maneja en forma explícita en ese modelo. A partir de soluciones obtenidas en etapas anteriores de la búsqueda, se realizan actualizaciones sobre el modelo para que la búsqueda se concentre en regiones que contengan soluciones de buena calidad. Las técnicas dentro de esta clase pueden utilizar información adicional al modelo, como por ejemplo una memoria auxiliar de soluciones para ser consideradas al momento de realizar la actualización del modelo. ACO y los métodos del tipo Estimation of Distribution Algorithms (EDA[2]) [124] son técnicas de búsqueda basadas en modelo.

Dos de las técnicas del tipo EDA, Population-Based Incremental Learning (PBIL) [12] y Compact Genetic Algorithm (cGA) [95], presentan interesantes puntos de contacto con algunas variantes de ACO, en lo que respecta a las reglas de actualización del modelo que utilizan [21, 118].

2.4. Conclusiones

En el presente capítulo se han introducido nociones elementales de optimización combinatoria, teoría de NP-completitud y metaheurísticas que sirven de marco de refencia para el resto de este trabajo. Asimismo, se han presentado algunas taxonomías para metaheurísticas que agrupan técnicas que tienen características similares.

[2]Los EDA pertenecen a la familia de los algoritmos evolutivos y por tanto están comprendidos dentro de EC.

Capítulo 3

Ant Colony Optimization

"Algo le llamó la atención desde el suelo. Algo vivo se movía sobre su mano: era un ser quitinoso que le causó admiración; lo recordaba por haberlo visto en otros tiempos. La hormiga sujetaba una pequeña partícula blanca entre sus mandíbulas, y la miró alejarse. Las hormigas no eran muy inteligentes como especie, pero al menos habían logrado sobrevivir."
Philip K. Dick, *La penúltima verdad*

En los últimos años, la comunidad científica ha realizado una gran cantidad de propuestas de nuevas metaheurísticas que prometían resolver un amplio espectro de problemas de optimización del tipo NP-difícil. Sin embargo, en la práctica solamente un grupo pequeño de esas propuestas ha logrado consolidarse, demostrando una amplia aplicabilidad sobre problemas de muy diversas características y adquiriendo la madurez necesaria para ser una alternativa real al momento de resolver un problema de optimización.

Ant Colony Optimization (ACO) es una metaheurística sobre la que se ha trabajado ampliamente en los últimos quince años, demostrando su potencial al haberse aplicado con éxito sobre varios problemas estándares de optimización. Este capítulo introduce las principales características de la metaheurística basada en población ACO, presentándose un amplio relevamiento de las principales variantes propuestas de ACO en los últimos quince años. El eje central de este relevamiento es el estudio de las variantes propuestas para abordar problemas estáticos de optimación combinatoria.

El resto del capítulo se estructura de la siguiente forma. En la sección 3.1 se presenta el esquema general de la metaheurística ACO. Las secciones subsiguientes presentan una amplia reseña de las variantes propuestas que instancian ese esquema general. El orden de presentación de las propuestas en el capítulo intenta ser fiel a su orden cronológico de aparición, de modo que resulte más sencillo explicar las mejoras que incorpora cada una de las variantes. En cada una de las secciones, además de describirse las principales características de la variante propuesta, se comentan los parámetros que utiliza y se realiza una evaluación de la variante a partir de la bibliografía relevada. Ant System fue la primera propuesta formulada que se enmarca dentro del esquema general de ACO. En la sección 3.2 se presentan el algoritmo Ant System y las variantes estrechamente vinculadas al mismo. Posteriormente, en las secciones 3.3 a 3.7 se describen los algoritmos ACO que han tenido más amplia difusión en los últimos años. En las secciones 3.8 y 3.9 se agrupan las variantes de ACO que utilizan una población auxiliar y las que comienzan la construcción de las soluciones desde una solución parcial respectivamente. Posteriormente en la sección 3.10 se describen algoritmos que están fuertemente vinculados a la metaheurística ACO aunque presentan características que se apartan de su

esquema general. Finalmente, la sección 3.11 presenta un resumen de la evaluación de las variantes de ACO y las conclusiones de este relevamiento.

3.1. Ant Colony Optimization (ACO)

El comportamiento social de las colonias de hormigas ha servido de fuente de inspiración para el diseño de múltiples algoritmos. A pesar de las limitadas capacidades de cada hormiga, la colonia logra en forma colectiva realizar tareas de gran complejidad a través de la coordinación entre los individuos. Una de las maneras de comunicación que utilizan las hormigas para lograr tal coordinación de tareas es en forma indirecta y consiste en realizar modificaciones en su entorno[1]. Como las hormigas tienen una visión escasa, la comunicación se realiza mediante el marcado en el suelo de rastros que sirven de referencia a otras hormigas, a través de la liberación de una sustancia química llamada feromona.

ACO es una metaheurística basada en población formulada por Dorigo [63, 64, 68] que unifica, bajo un esquema general común, varias propuestas de técnicas de resolución de problemas de optimización caracterizadas por la utilización de hormigas artificiales. El mecanismo de funcionamiento de ACO se caracteriza por construir soluciones en forma incremental a partir del agregado de componentes. En la construcción de las soluciones se incorpora la experiencia adquirida durante la búsqueda, utilizando una memoria que almacena el rastro depositado por las hormigas en la construcción de soluciones de buena calidad.

Las características fundamentales de las técnicas ACO son:

- Se utilizan agentes (las hormigas artificiales) para construir soluciones en forma incremental. Cada una de las hormigas construye en forma independiente una solución mediante la incorporación de componentes sobre una solución parcial.

- Para la incorporación de componentes se realiza una elección mediante una regla probabilística que tiene en cuenta la experiencia adquirida en etapas anteriores de la búsqueda e información heurística del problema considerado.

- Para incorporar la experiencia adquirida en la construcción de soluciones se utiliza una matriz de feromona, a modo de memoria que almacena el rastro depositado por las hormigas en la construcción de soluciones de buena calidad.

En el algoritmo 1 se presenta la estructura general de un algoritmo ACO aplicado a un problema de optimización combinatoria estático. Se han propuesto variantes que adaptan el esquema general de ACO para la resolución de otros tipos de problemas de optimización, por ejemplo: problemas de dominio continuo [16, 69, 158, 163], problemas multiobjetivo [14, 56, 60, 82, 89] y problemas dinámicos [51, 52, 53, 54, 55]. Las diferencias existentes en la estructura del algoritmo ACO aplicado sobre otros tipos de problemas no serán comentadas por estar fuera del alcance de este trabajo.

En primer lugar se realiza una etapa de inicialización donde se establecen los valores de partida del algoritmo para los rastros de feromona ($initializePheromoneTrails$). Luego, cada una de las iteraciones se divide en tres etapas:

[1] A este concepto se le llama *stigmergy* en la literatura científica y se suele traducir al castellano como estigmergia.

Algoritmo 1 ACO aplicado a un problema estático

```
T = initializePheromoneTrails()
s_best = s | f(s)= +∞
while not stopCriteria() do
   pop = constructAntsSolutions(T)
   pop'= applyLocalSearch(pop)   % opcional
   T = updatePheromones(T,pop')
   s = selectBestOfPopulation(pop')
   if f(s) < f(s_best) then
      s_best = s
   end if
end while
return s_best
```

- En la primera etapa, en el procedimiento *constructAntsSolutions*, cada hormiga de la colonia en forma concurrente, independiente y asíncrona construye incrementalmente una solución mediante la incorporación de componentes sobre una solución parcial. Para la incorporación de los componentes se realiza una elección mediante una regla probabilística que considera la experiencia adquirida en la búsqueda a través del rastro de feromona depositada e información heurística de los componentes considerados. La incorporación de la información heurística se realiza con el objetivo de garantizar la diversificación de la búsqueda, es decir la exploración de nuevas regiones del espacio de búsqueda.

- La segunda etapa es opcional y consiste en la utilización de una búsqueda local que permita mejorar las soluciones encontradas mediante el procedimiento *applyLocalSearch*, que aplica una técnica específica para el problema considerado. En este capítulo el interés está centrado en la mecánica de funcionamiento de los algoritmos y no en las particularidades de la aplicación de las técnicas ACO sobre un problema en concreto, por lo que no se entrará en mayores detalles al respecto.

- La última etapa consiste en la actualización del rastro de feromona depositado en las componentes a partir de la calidad de las soluciones construidas mediante el procedimiento *updatePheromones*. El valor de los rastros almacenados en la matriz puede decrementarse por vía de la evaporación o incrementarse por el depósito en las componentes utilizadas por las hormigas para construir soluciones. La evaporación de feromona tiene como objetivo evitar la convergencia prematura del algoritmo a regiones no óptimas, permitiendo continuar con la exploración del espacio de búsqueda, a partir de la utilización de la información heurística. Por el contrario, el depósito de feromona en una componente utilizada en la construcción de una buena solución la vuelve más atractiva en el futuro, aumentando su probabilidad de ser seleccionada en la etapa de construcción, intensificando la búsqueda en regiones próximas a soluciones de buena calidad.

La comunidad científica ha propuesto múltiples variantes que instancian el esquema general descrito. En las próximas secciones se presentan las variantes más relevantes. La notación utilizada por Blum y Dorigo [20] se utilizará en las secciones subsiguientes,

ya que contribuye a la claridad al ser independiente del problema considerado. En particular, se notará: c_{i,x_i} a la componente de una solución s que se corresponde con la combinación de una variable X_i con uno de los valores de su dominio $x_i \in D_i$, y τ_{i,x_i} al valor correspondiente al rastro de feromona en la componente c_{i,x_i}.

3.2. Ant System (AS) y variantes similares

La primera propuesta que se enmarca dentro del esquema general de ACO fue realizada por Dorigo [38, 62, 67] en el contexto de su tesis de doctorado y estableció las características fundamentales de este tipo de técnicas. La propuesta consiste en tres variantes diferentes: *ant-density*, *ant-quantity* y *ant-cycle*. Las dos primeras variantes presentaron en la práctica peores resultados que la tercera, y por tanto cayeron rápidamente en desuso, utilizándose en la actualidad el nombre Ant System (AS) para referirse a la variante *ant-cycle*. A continuación se presenta la descripción de AS, indicándose donde corresponda las diferencias con las variantes en desuso.

En el algoritmo 2 se presenta el AS, siendo A el conjunto de hormigas (colonia), T la matriz de feromona y s_a la solución construida por una hormiga $a \in A$ a partir de T y un criterio heurístico H. Para cada paso del bucle principal, se construyen soluciones para cada una de las hormigas que posteriormente serán utilizadas para actualizar el rastro de feromona.

Algoritmo 2 Ant System

```
T = initializePheromoneValues()
s_best = s | f(s)= +∞
while not stopCriteria() do
   for all a ∈ A do
      s_a = constructAntSolution(T,H)
      T = applyOnlineDelayedPheromoneUpdate(T,s_a|a ∈ A)
      if f(s_a) < f(s_best) then
         s_best = s_a
      end if
   end for
end while
return s_best
```

El procedimiento *initializePheromoneValues* inicializa los valores del rastro de feromona (τ_{init}) para cada una de las componentes que pueden formar parte de una solución. En caso de utilizarse valores muy grandes para la inicialización serán necesarias varias iteraciones antes de que la evaporación logre reducir suficientemente su efecto, permitiendo que sea realmente la feromona depositada por las hormigas quien guíe la búsqueda. Por el contrario, si los valores utilizados para la inicialización son muy pequeños, la búsqueda estará fuertemente orientada por las primeras soluciones generadas. La experiencia indica que conviene inicializar el rastro en una cantidad similar a la que será depositada en una iteración por la colonia de hormigas. Naturalmente, esto implica la necesidad de conocer el costo de una solución de buena calidad al problema considerado. Típicamente se utiliza el costo asociado a una solución construida mediante alguna heurística auxiliar sencilla o simplemente la estimación del costo de

una buena solución. En general, se recomienda inicializar el rastro de feromona de la siguiente forma $\tau_{i,x_i} = \frac{m}{Costo(SolucionAuxiliar)}, \forall \tau_{i,x_i} \in T$, siendo m la cantidad de hormigas y $Costo(SolucionAuxiliar)$ el costo de una solución construida utilizando alguna heurística auxiliar [68].

Posteriormente, cada hormiga construye incrementalmente una solución mediante el agregado de componentes sobre una solución parcial s^p (inicialmente vacía) a través del procedimiento *constructAntSolution*. La elección de la próxima componente a agregar se realiza utilizando la regla de transición de estados conocida como regla proporcional aleatoria (random proportional rule) que se presenta en la ecuación 3.1.

$$p(c_{i,x_i}|s^p) = \begin{cases} \frac{[\tau_{i,x_i}]^\alpha [\eta_{i,x_i}]^\beta}{\sum_{c_{j,x_j} \in J(s^p)} [\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta} & \text{si } c_{i,x_i} \in J(s^p) \\ 0 & \text{en otro caso} \end{cases} \tag{3.1}$$

En la ecuación 3.1 los parámetros α y β son utilizadas para ajustar la influencia relativa entre los valores de feromona (τ_{i,x_i}) y los valores asociados a la información heurística (η_{i,x_i}), y $J(s^p)$ es el conjunto de componentes que pueden ser agregados a la solución parcial s^p. Se conoce como visibilidad al término de la información heurística. Típicamente se utiliza $\eta_{i,x_i} = \frac{1}{Costo(c_{i,x_i})}$. En la regla de transición de estados si se anula el parámetro α, considerando únicamente la visibilidad, se puede interpretar el algoritmo resultante como un algoritmo greedy aleatorio con múltiples puntos de arranque. En cambio, si se anula el parámetro β, considerando únicamente la información histórica, se observa la aparición del fenómeno conocido como estancamiento, que consiste en la situación en la cual todas las hormigas siguen exactamente el mismo camino y construyen exactamente la misma solución. También se suele observar estancamiento cuando se trabaja con valores de α mayores a 1 [67].

Cuando todas las hormigas han construido sus soluciones, se realiza la actualización de la feromona mediante la aplicación de la regla de actualización conocida como *online delayed pheromone update rule*. El objetivo de la regla es aumentar la cantidad de feromona en los componentes de la solución que han sido utilizados por las soluciones de calidad alta. La regla de actualización se presenta en las ecuaciones 3.2 y 3.3.

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \sum_{a \in A} \Delta\tau_{i,x_i}^{s_a} \quad \forall \tau_{i,x_i} \in T \tag{3.2}$$

$$\text{siendo} \quad \Delta\tau_{i,x_i}^{s_a} = \begin{cases} F(s_a) & \text{si } c_{i,x_i} \in s_a \\ 0 & \text{en otro caso} \end{cases} \tag{3.3}$$

En la ecuación 3.2, ρ es la tasa de evaporación de la feromona ($0 \leq \rho \leq 1$). En la ecuación 3.3, $F : S \mapsto \mathbb{R}^+$ es la función de calidad que indica la calidad de la solución considerada. Típicamente se utiliza $F(s_a) = \frac{1}{Costo(s_a)}$ [68]. Originalmente se utilizaba $F(s_a) = \frac{Q}{Costo(s_a)}$, siendo Q una constante, pero se han realizado estudios empíricos que muestran que el valor de Q no tiene mayor impacto en la calidad de las soluciones obtenidas [67], por lo cual se suele considerar como 1 para eliminar un parámetro.

La principal diferencia entre AS y las variantes que cayeron en desuso es la forma en que se realiza la actualización del rastro de feromona. *Ant-density* y *ant-quantity* se caracterizan porque cada una de las hormigas deposita la feromona cuando utiliza la componente, en lugar de realizarlo cuando se ha completado la construcción de la

solución. La variante *ant-density* utiliza la regla de la ecuación 3.4, mientras que la variante *ant-quantity* utiliza la regla de la ecuación 3.5.

$$\Delta\tau^{a}_{i,x_i} = \begin{cases} Q & \text{si la hormiga } a \text{ agrega } c_{i,x_i} \text{ en ese paso} \\ 0 & \text{en otro caso} \end{cases} \tag{3.4}$$

$$\Delta\tau^{a}_{i,x_i} = \begin{cases} \frac{Q}{Costo(c_{i,x_i})} & \text{si la hormiga } a \text{ agrega } c_{i,x_i} \text{ en ese paso} \\ 0 & \text{en otro caso} \end{cases} \tag{3.5}$$

Los parámetros que utiliza AS son el tamaño de la población de hormigas m, la cantidad inicial de feromona en las componentes de las soluciones τ_{init}, la influencia relativa de la componente del rastro de feromona α, la influencia relativa del componente heurístico β y la tasa de evaporación de la feromona ρ.

Existen dos propuestas posteriores que intentan mejorar los resultados obtenidos por AS. Elitist Ant System (EAS) y Rank-Based Ant System (AS_{rank}) presentan grandes similitudes con AS, por lo que serán presentadas en esta misma sección.

3.2.1. Elitist Ant System (EAS)

La idea central de EAS es darle mayor énfasis al mejor camino encontrado, ponderándolo de forma particular mediante un refuerzo adicional al momento de realizar la actualización de los rastros de feromona [67]. Esta idea puede interpretarse como si la mejor solución construida fuera recorrida por un cierto número de hormigas, las que se denominan hormigas elitistas. Con respecto a AS, el único cambio se provoca en la regla de actualización del rastro de feromona. La regla de actualización de EAS se presenta en la ecuación 3.6.

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \sum_{a \in A} \Delta\tau^{s_a}_{i,x_i} + e * \Delta\tau^{s_{bs}}_{i,x_i} \quad \forall \tau_{i,x_i} \in T \tag{3.6}$$

En la ecuación 3.6, s_{bs} es la mejor solución encontrada hasta el momento (best-so-far) y e la cantidad de hormigas elitistas utilizadas. La regla para calcular la cantidad de feromona a depositar se mantiene incambiada (presentada en la ecuación 3.3).

EAS utiliza los mismos parámetros que AS e incorpora el parámetro cantidad de hormigas elitistas e.

Posteriormente, se ha propuesto utilizar para el refuerzo la mejor solución generada por cada una de las hormigas hasta el momento en lugar de la mejor solución hasta el momento [101, 168]. Esa propuesta se conoce como Ant System Local Best Tour (AS-LBT). Los autores de AS-LBT señalan como una virtud que puede funcionar en forma completamente distribuida.

También existen propuestas sobre EAS para aumentar la diversidad, que pueden ser aplicadas sobre otras variantes. Una propuesta sencilla en esa línea consiste en depositar un refuerzo adicional de feromona sobre las componentes que no se hayan utilizado por ninguna hormiga para construir soluciones en la iteración [75]. La idea de la propuesta es volver atractivas componentes que no forman parte ni de las buenas ni de las malas soluciones, ya que en caso contrario el efecto de la evaporación volverá a esas componentes menos atractivas.

3.2.2. Rank-Based Ant System (AS_{rank})

La incorporación de elitismo sobre AS podría otorgar un énfasis excesivo a la intensificación, privilegiando en demasía las regiones del espacio de búsqueda cercanas a la mejor solución encontrada. Otro aspecto negativo de la incorporación de elitismo es que la distribución de la feromona es realizada por todas las hormigas que construyen soluciones, permitiendo reforzar el rastro de feromona de soluciones de muy baja calidad. Intentado paliar estos inconvenientes Bullnheimer et al. propusieron en 1997 [26, 27, 28] la variante AS_{rank}[2].

La idea de AS_{rank} consiste en ordenar las soluciones construidas por las hormigas de acuerdo a su costo y ponderar su contribución en la actualización del rastro de acuerdo a la posición μ que ocupe cada solución en el ranking. Adicionalmente, únicamente las mejores $\omega - 1$ soluciones son tomadas en cuenta en la actualización. Del mismo modo que en EAS se utilizan hormigas elitistas, asignándoles el peso máximo ω. Se considera como el factor de ponderación mínimo el 1, por lo cual se deduce que el peso para la μ-ésima hormiga es $\omega - \mu$, resultando la regla de actualización que se presenta en la ecuación 3.7. Para el cálculo de la cantidad de feromona a depositar se utiliza la misma regla que en la variante AS (presentada en la ecuación 3.3).

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \sum_{\mu=1}^{\omega-1} (\omega - \mu) \Delta\tau_{i,x_i}^{s_\mu} + \omega * \Delta\tau_{i,x_i}^{s_{bs}} \quad \forall \tau_{i,x_i} \in T \tag{3.7}$$

La variante AS_{rank} utiliza los mismos parámetros que AS, e incorpora el parámetro ω para la cantidad de hormigas que depositan feromona.

También existen propuestas sobre AS_{rank} para aumentar la diversidad que pueden ser aplicadas sobre otras variantes. Uno de los mecanismos propuestos consiste en seleccionar la siguiente componente aleatoriamente de forma equiprobable [125, 126]. El mecanismo descrito incorpora un parámetro r a partir del cual y en forma aleatoria se determina si para la próxima componente, se debe utilizar la regla de transición de estados (presentada en la ecuación 3.1) o la nueva regla de selección aleatoria.

3.2.3. Evaluación

La variante AS ha sido utilizada para resolver el Travelling Salesman Problem (TSP), el Asymmetric Travelling Salesman Problem (ATSP), el Quadratic Assignment Problem (QAP) y el Job-Shop Scheduling Problem (JSSP) entre otros problemas, mostrando la versatilidad de ACO como metaheurística.

La variante AS presenta en la práctica excelentes resultados sobre el TSP para instancias pequeñas, inclusive aventajando a metaheurísticas ampliamente usadas como SA, TS y Genetic Algorithms (GA). Al considerarse instancias de tamaño creciente, resulta necesaria la inclusión de mecanismos como el ranking o la utilización de operadores de búsqueda local para mantener su potencia.

El TSP es el problema que generalmente se ha utilizado para la validación de propuestas. En particular, los resultados obtenidos por AS_{rank} superan a los de EAS, que a su vez superan a los resultados obtenidos por AS [68].

El código fuente de las variantes AS, EAS y AS_{rank} está disponible en *The Ant Colony Optimization Home Page* [61].

[2]El nombre original de la variante fue Ant system with elitist strategy and ranking.

3.3. Ant Colony System (ACS)

Explorando alternativas para el problema de la degradación de la calidad de las soluciones al aumentar el tamaño de las instancias consideradas, Gambardella y Dorigo propusieron en 1995 [79] la incorporación de ideas presentes en técnicas de aprendizaje automático . En particular se propuso el algoritmo ANT-Q, que incorporaba mecanismos similares a los de la técnica de aprendizaje por refuerzo conocida como Q-Learning [167]. Posteriormente, la propuesta fue reformulada en algunos de sus detalles y es conocida actualmente como Ant Colony System (ACS) [65, 66].

ACS se diferencia de AS fundamentalmente en los siguientes aspectos:

1. **Regla de transición de estados:** en ACS se modifica la regla de transición de estados incluyendo un mecanismo directo para balancear entre la explotación del conocimiento acumulado del problema y la exploración controlada de nuevas componentes, mediante la incorporación del parámetro $q_0 \in [0, 1]$. Cuando una hormiga debe incorporar una nueva componente, genera un número aleatorio $q \in [0, 1]$ y aplica la regla de transición de estados conocida como regla proporcional pseudoaleatoria (pseudorandom proportional rule) que se muestra en las expresiones de la ecuación 3.8.

$$\begin{array}{l} \text{Si} \quad q \leq q_0\text{:} \\ p(c_{i,x_i}|s^p) = \left\{ \begin{array}{ll} 1 & \text{si } c_{i,x_i} = argmax([\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta) \text{ con } c_{j,x_j} \in J(s^p) \\ 0 & \text{en otro caso} \end{array} \right. \\ \\ \text{Si} \quad q > q_0\text{:} \\ p(c_{i,x_i}|s^p) = \left\{ \begin{array}{ll} \frac{[\tau_{i,x_i}]^\alpha [\eta_{i,x_i}]^\beta}{\sum_{c_{j,x_j} \in J(s^p)} [\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta} & \text{si } c_{i,x_i} \in J(s^p) \\ 0 & \text{en otro caso} \end{array} \right. \end{array} \tag{3.8}$$

   Se suele eliminar el parámetro α, considerándolo como 1, aunque en ninguno de los trabajos considerados en esta reseña aparece una justificación directa para su eliminación. Sin embargo, conviene notar que en el algoritmo AS original α y β ponderan a los términos que reflejan la información histórica (explotación) y heurística (exploración) respectivamente. Mientras que en ACS existe un parámetro (q_0) que explícitamente realiza el balance entre la explotación (si $q \leq q_0$) y una exploración controlada (si $q > q_0$), con lo cual se pierde el sentido que originalmente tenían ambos parámetros, fundamentalmente α.

2. **Actualización del rastro de feromona:** En ACS solamente una hormiga por iteración actualiza el rastro de feromona, pero el depósito y la evaporación de feromona se realiza únicamente sobre las componentes que están presentes en la solución de la hormiga considerada ($\tau_{i,x_i} \in T^a$). Existen dos opciones sobre qué solución considerar para la actualización: la mejor hasta el momento o la mejor de la iteración. Típicamente se utiliza la mejor solución hasta el momento porque es la que ha presentado los mejores resultados en la práctica [65]. La regla de actualización global del rastro de feromona se presenta en la ecuación 3.9.

$$\tau_{i,x_i} \leftarrow (1 - \rho) * \tau_{i,x_i} + \rho * \Delta\tau_{i,x_i}^{s_{bs}} \quad \forall \tau_{i,x_i} \in T^{bs} \tag{3.9}$$

La regla para el cálculo de la cantidad de feromona a depositar utilizada por ACS es la misma que la de AS, presentada en la ecuación 3.3. Solamente se realizan cambios en las componentes que forman parte de la mejor solución hasta el momento, mientras que para el resto de las componentes no se realiza actualización. La feromona depositada es multiplicada por el factor ρ, con lo cual la nueva cantidad de feromona es en realidad un promedio ponderado entre la cantidad anterior y la cantidad depositada.

3. **Actualización local del rastro de feromona:** ACS utiliza un mecanismo de actualización local del rastro de feromona. Al incorporar una componente a su solución parcial, cada hormiga actualiza el rastro de feromona de acuerdo a la expresión de la ecuación 3.10.

$$\tau_{i,x_i} \leftarrow (1 - \xi) * \tau_{i,x_i} + \xi * \tau_{init} \tag{3.10}$$

En la ecuación 3.10, ξ es la tasa local de evaporación de feromona ($0 < \xi < 1$) y τ_{init} es la cantidad de feromona depositada localmente. El algoritmo ANT-Q utiliza como valor para la cantidad de feromona depositada localmente $\gamma * max(\tau_{i,x_i})$ con $c_{i,x_i} \in J(s^p)$, es decir el valor máximo de feromona para el próximo componente a introducir en la solución multiplicado por el parámetro γ [79].

La idea detrás de la actualización local es provocar pequeñas reducciones en el valor de feromona de las componentes en la misma iteración a medida que las hormigas van utilizando las componentes para construir las soluciones. De esta forma se vuelve menos atractiva la inclusión por parte del resto de las hormigas de las componentes utilizadas en la misma iteración, permitiendo la exploración de componentes no utilizadas. A raíz de la actualización local, el algoritmo no presenta el mismo resultado si las hormigas construyen sus soluciones en forma secuencial o concurrente. En general se ha optado por permitir la construcción en paralelo, pero no existen estudios que comparen ambas opciones [68].

ACS también cuenta con el mérito de haber sido la primera variante que incorporó la utilización de una lista de candidatos para obtener mejoras en el desempeño cuando se resuelven instancias grandes de los problemas [68]. Al utilizar una lista de candidatos se evita considerar todas las posibles opciones cuando se debe agregar un componente a la solución parcial, concentrándose solamente en un subconjunto de las opciones posibles. En caso de que todas las componentes disponibles en la lista de candidatos ya formaran parte de la solución, se procede a considerar el resto de las opciones posibles.

La variante ACS utiliza los mismos parámetros que la variante AS (excepto α) e incorpora los parámetros q_0 para el balance entre las reglas de transición de estados y ξ para la tasa local de evaporación de feromona.

3.3.1. Evaluación

ACS presenta en la práctica excelentes resultados sobre el TSP para instancias pequeñas, aventajando a otras metaheurísticas como SA, GA y EP, mejorando inclusive los resultados obtenidos por AS. Sin embargo, requiere la incorporación de un operador de búsqueda local al considerarse instancias grandes [65].

ANT-Q es una variante predecesora de ACS y con la que está estrechamente vinculada. Sin embargo, ANT-Q ha caído en desuso debido a que la forma de cálculo de la actualización local de feromona es más costosa computacionalmente que la de ACS aunque ambas variantes presentan resultados de calidad similar sobre el TSP [65].

El código fuente de ACS está disponible en *The Ant Colony Optimization Home Page* [61].

3.4. $\mathcal{MAX} - \mathcal{MIN}$ Ant System ($\mathcal{MM}$AS)

Como se señaló en la sección 3.2, los buenos resultados obtenidos por AS sobre instancias pequeñas del TSP no se mantenían al incrementar las dimensiones de las instancias consideradas. En 1996, Stützle y Hoos formularon la variante $\mathcal{MAX}-\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) [149, 150, 151, 152] para resolver algunas de las carencias detectadas en AS. La idea fundamental de la variante $\mathcal{MM}$AS es lograr una mayor explotación de las mejores soluciones encontradas aunque se incorporan mecanismos para garantizar que no se alcance un estado de estancamiento en etapas tempranas de la búsqueda.

Las principales características de la propuesta son las siguientes:

1. **Actualización del rastro de feromona:** en $\mathcal{MM}$AS en cada iteración solamente se deposita feromona sobre las componentes de la solución asociada a una hormiga. Si bien la idea de utilizar solamente una hormiga para actualizar el rastro presenta algún punto de contacto con ACS, el mecanismo de esta variante presenta mayores similitudes con AS, ya que se realiza la evaporación sobre todas las componentes. Existen dos opciones sobre qué solución considerar para realizar el depósito de feromona: la mejor hasta el momento o la mejor de la iteración. La actualización de la feromona se realiza de acuerdo a la expresión de la ecuación 3.11.

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \Delta\tau_{i,x_i}^{s_{best}} \quad \forall\tau_{i,x_i} \in T \tag{3.11}$$

   En la ecuación 3.11, s_{best} es la mejor solución de la iteración o la mejor solución hasta el momento. Stützle y Hoos recomiendan intercalar en algunas iteraciones la utilización de la mejor solución de la iteración y la mejor solución hasta el momento [149]. La regla para el cálculo de la cantidad de feromona a depositar se mantiene incambiada respecto a la utilizada en la variante AS.

2. **Límites en el rastro de feromona:** en $\mathcal{MM}$AS se incorporan los límites explícitos τ_{min} y τ_{max} para la cantidad de feromona que puede haber en una componente, cumpliéndose $\tau_{min} \leq \tau_{i,x_i} \leq \tau_{max}$. Después de cada iteración se debe asegurar que el rastro de la feromona quede comprendido en el umbral, por lo cual si $\tau_{i,x_i} > \tau_{max}$ se ajusta $\tau_{i,x_i} = \tau_{max}$ y si $\tau_{i,x_i} < \tau_{min}$ se ajusta $\tau_{i,x_i} = \tau_{min}$. Imponiendo las condiciones naturales $\tau_{min} > 0$ y $\eta_{i,x_i} < \infty$, la probabilidad de elegir un componente específico siempre es positiva.

   Para estimar el límite τ_{max} se toma como referencia el mayor valor que teóricamente podría tener el rastro de feromona, $\tau_{max}^{teo} = \frac{1}{\rho * Costo(s_{opt})}$. Evidentemente no se dispone del costo óptimo $Costo(s_{opt})$, por lo cual se utiliza el costo correspondiente a la mejor solución encontrada hasta el momento $Costo(s_{bs})$. Por lo tanto, en cada iteración en la que se obtenga una mejor solución se debe ajustar el valor de τ_{max}.

La estimación de τ_{min} es bastante más compleja. A continuación se presenta un breve resumen de los conceptos esenciales presentados por Stützle y Hoos para la comprensión de la expresión resultante que se presenta en la ecuación 3.12 [152]. El problema utilizado para desarrollar la estimación de τ_{min} es el TSP. Se define la convergencia de $\mathcal{MM}$AS como el estado en el que la distribución de rastro de feromona en cada punto de selección de componentes cumple que una de las posibles componentes tiene τ_{max} y el resto tiene τ_{min}. Cuando se ha alcanzado la convergencia, la probabilidad de construir la mejor solución hasta el momento es $p_{best} > 0$. Una hormiga construirá la mejor solución hasta el momento si en cada selección de componente realiza la elección correcta, es decir si selecciona la componente que tiene asociado como rastro de feromona τ_{max}. La hormiga deberá realizar n (la cantidad de ciudades) decisiones correctas, es decir una por cada componente que selecciona. Finalmente, se asume que la cantidad de opciones entre las que se debe elegir en cada paso es constante (avg) y que la probabilidad de tomar la decisión correcta en cada paso también es constante. Bajo estos supuestos se determina la estimación de τ_{min} que se presenta en la ecuación 3.12.

$$\tau_{min} = \frac{\tau_{max} * (1 - \sqrt[n]{p_{best}})}{(avg - 1) * \sqrt[n]{p_{best}}} \tag{3.12}$$

3. **Inicialización de los rastros de feromona:** en $\mathcal{MM}$AS los rastros de feromona son inicializados con el máximo valor posible para todas las componentes (τ_{max}). Antes de la primera iteración no se conoce una solución al problema, por lo cual se utiliza un valor alto, para que tras la primera iteración todas las componentes tengan asociado el valor τ_{max}. Se suelen utilizar tasas de evaporación lenta, logrando aumentar la exploración que se realiza en etapas tempranas del algoritmo.

Adicionalmente, se propone el mecanismo de suavizado de los rastros de feromona (Pheromone Trail Smoothing, PTS). La idea del mecanismo consiste en fomentar la exploración incrementando la probabilidad de seleccionar componentes con una baja cantidad de feromona cuando el $\mathcal{MM}$AS ha convergido o está próximo a la convergencia. Existen mecanismos que permiten determinar cuando se está en esas condiciones, por ejemplo el Average Branching Factor, presentado originalmente como el Mean λ-Branching Factor [79]. El rastro se incrementa en forma proporcional a la diferencia al máximo rastro posible mediante la expresión de la ecuación 3.13 [152].

$$\tau^*_{i,x_i} \leftarrow \tau_{i,x_i} + \delta * (\tau_{max} - \tau_{i,x_i}) \quad \text{con} \quad 0 < \delta < 1 \quad \forall \tau_{i,x_i} \in T \tag{3.13}$$

$\mathcal{MM}$AS incorpora la utilización de lista de candidatos descrita en la sección 3.3.

La variante $\mathcal{MM}$AS utiliza los mismos parámetros que AS e incorpora los parámetros τ_{min} para la cantidad mínima de feromona en las componentes de las soluciones, τ_{max} para la cantidad máxima de feromona en las componentes de las soluciones, p_{best} para la probabilidad de que una hormiga construya la mejor solución, avg para el promedio de la cantidad de componentes entre los que debe elegir una hormiga en cada paso para armar una solución completa y δ para la tasa del PTS. Aunque la cantidad de parámetros es alta, varios están determinados analíticamente y simplemente se debe realizar su cálculo como τ_{min}, τ_{max} y τ_{init}, mientras que avg queda determinado directamente por el problema considerado. Los parámetros que requieren calibración son los mismos que en AS (excepto τ_{init}), p_{best} y δ (si se incorpora el mecanismo PTS).

3.4.1. Evaluación

$\mathcal{MM}$AS es una variante que presenta muy buenos resultados y que ha sido ampliamente utilizada, dejando en segundo plano a las variantes más simples como AS, EAS y AS_{rank}. En particular, Dorigo y Stützle realizaron un amplio estudio sobre las variantes AS, EAS, AS_{rank}, $\mathcal{MM}$AS y ACS aplicadas al TSP, concluyendo que $\mathcal{MM}$AS obtiene los mejores resultados [68]. Si bien los resultados obtenidos por ACS son levemente inferiores en calidad a los de $\mathcal{MM}$AS, Dorigo y Stützle constataron que la evolución en la calidad de las soluciones obtenidas por $\mathcal{MM}$AS es más lenta que la de ACS [68]. Es posible que este comportamiento se deba a que ACS realiza una mayor explotación de las mejores soluciones encontradas y por tanto no logra explorar correctamente el espacio de búsqueda.

$\mathcal{MM}$AS ha demostrado su versatilidad al haber sido utilizada exitosamente para resolver el TSP, ATSP y QAP, aunque al incrementarse el tamaño de los problemas considerados también requiere la utilización de operadores de búsqueda local para alcanzar resultados competitivos con otras metaheurísticas.

Stützle y Hoos señalan que el límite inferior del rastro de feromona juega un rol más importante que el límite superior para la obtención de soluciones de buena calidad [149]. La fórmula planteada en la ecuación 3.12 para el límite inferior puede presentar dificultades al intentar aplicarla en otros problemas, debido a la dificultad de estimar el valor de algunas variables necesarias (por ejemplo, avg).

El código fuente de $\mathcal{MM}$AS está disponible en *The Ant Colony Optimization Home Page* [61].

3.5. Approximate Nondeterministic Tree Search (ANTS)

Approximate Nondeterministic Tree Search (ANTS) es una propuesta de Maniezzo realizada en 1998 [112, 113, 114, 115] que incorpora conceptos de programación matemática y presenta similitudes con ideas de la técnica Branch & Bound.

ANTS difiere del mecanismo básico del AS en los siguientes aspectos:

1. **Utilización de cotas inferiores:** en ANTS se modifica el mecanismo provisto para considerar la información heurística de una componente. El mecanismo propuesto para esta variante consiste en calcular una cota inferior del costo de completar la solución parcial utilizando una determinada componente, en lugar de considerar únicamente el costo de la componente. Con un menor costo de la cota inferior, más atractiva tiene que resultar la componente, ya que su inclusión en la solución es más prometedora que la inclusión de las otras componentes entre las que se puede optar. El mecanismo descrito permite evitar la construcción de soluciones que superarán el costo de la mejor solución hasta el momento. Si para una solución parcial, cuando se calculan las cotas inferiores para cada una de las componentes disponibles, todas superan el costo de la mejor solución encontrada hasta el momento, la hormiga no debe proseguir con la construcción de la solución.

   La ventaja sustancial que tiene este mecanismo es que aporta una visión global de la solución en la información heurística considerada, al tener en cuenta cómo repercute la inclusión de la componente en la construcción completa de la solución y en el costo de la solución construida. La propuesta contrasta con el enfoque seguido por AS que tiene una visión sumamente local, al considerar exclusivamente

el costo de la componente. Como contrapartida el mecanismo puede ser costoso computacionalmente, porque implica que para cada solución parcial, en cada paso en el cual se deba incorporar una componente a la solución, se debe realizar el cálculo para obtener la cota inferior para cada posible componente.

2. **Regla de transición de estados:** en ANTS se modifica la regla de transición de estados, utilizando una expresión que es más simple desde el punto de vista computacional y de los parámetros considerados, ya que evita el cálculo de potencias y utiliza solamente un parámetro (ζ) en lugar de los dos habituales. En la ecuación 3.14 se presenta la regla de transición de estados propuesta por Maniezzo, siendo $0 \leq \zeta \leq 1$.

$$p(c_{i,x_i}|s^p) = \begin{cases} \frac{\zeta\tau_{i,x_i}+(1-\zeta)\eta_{i,x_i}}{\sum_{c_{j,x_j}\in J(s^p)} \zeta\tau_{j,x_j}+(1-\zeta)\eta_{j,x_j}} & \text{si } c_{i,x_i} \in J(s^p) \\ 0 & \text{en otro caso} \end{cases} \tag{3.14}$$

3. **Actualización del rastro de feromona:** se plantea un mecanismo diferente al usual que se caracteriza por no tener una forma explícita para la evaporación de feromona. La fórmula de actualización de la feromona se presenta en las expresiones de las ecuaciones 3.15 y 3.16.

$$\tau_{i,x_i} \leftarrow \tau_{i,x_i} + \sum_{a\in A} \Delta\tau_{i,x_i}^{s_a} \quad \forall\tau_{i,x_i} \in T \tag{3.15}$$

$$\Delta\tau_{i,x_i}^{s_a} = \begin{cases} \vartheta\left(1 - \frac{Costo(s_a)-LB}{Costo(s_{avg})-LB}\right) & \text{si } c_{i,x_i} \in s_a \\ 0 & \text{en otro caso} \end{cases} \tag{3.16}$$

En la ecuación 3.16, $Costo(s_a)$ es el costo de la solución considerada, $Costo(s_{avg})$ es el costo promedio de las últimas l soluciones construidas, LB es el valor de una cota inferior al costo de la solución óptima del problema que debe ser calculada al comienzo de la ejecución del algoritmo ($LB \leq Costo(s_{opt})$), mientras que l y ϑ son parámetros del algoritmo. Si $Costo(s_a) > Costo(s_{avg}) \Rightarrow \Delta\tau_{i,x_i}^{s_a} < 0$, se produce un decremento en la cantidad de feromona de la componente que puede ser visto como una evaporación implícita. En el caso contrario, se produce el aumento en la cantidad de feromona de la componente. El mecanismo planteado por Maniezzo provoca un escalado dinámico de la feromona. En etapas avanzadas de la búsqueda, cuando la calidad de las distintas soluciones sea similar, permite la detección de mejoras leves y el refuerzo de las correspondientes componentes.

Los parámetros que utiliza la variante ANTS son el tamaño de la población de hormigas m, la influencia relativa entre la componente de feromona y la componente heurística ζ, el factor en la fórmula de la cantidad de feromona depositada por una hormiga ϑ y la cantidad de soluciones consideradas para calcular el costo promedio l.

3.5.1. Evaluación

ANTS ha sido aplicada con éxito al QAP [112], FAP [113, 114] y sobre problemas de diseño de Data Warehouse [115]. En el caso del QAP, ha aventajado en calidad a los

resultados obtenidos mediante TS y GRASP, sin requerir mayores tiempos de ejecución [112]. No existen artículos que hagan referencia a su aplicación sobre el TSP. Dorigo y Stützle señalan que se han realizado experimentos limitados sobre el ATSP [68].

Existen aspectos de ANTS que pueden comprometer su éxito al abordar un problema concreto. La variante requiere de un mecanismo que a partir de una solución parcial permita estimar una cota inferior del costo. Es posible que al considerar un problema en particular no se conozca tal mecanismo, lo cual invalidaría la utilización de este algoritmo. Si para el problema considerado existiera ese mecanismo, debería obtener una cota con una precisión aceptable sin comprometer el desempeño debido a un alto costo computacional asociado a su cálculo. La estimación de las cotas inferiores se debe realizar para cada una de las componentes consideradas en cada paso de elección, con lo cual un mecanismo costoso computacionalmente es prohibitivo.

3.6. Hyper-Cube Framework for ACO (HCF-ACO)

Hyper-Cube Framework for ACO (HCF-ACO) es una propuesta realizada por Blum et al. en 2001 [22]. HCF-ACO no es exactamente una variante de ACO, sino una propuesta genérica para el manejo del rastro de feromona que puede ser aplicada sobre cualquier variante. La propuesta consiste en trabajar con un escalado de los valores de los rastros de feromona, de forma que permanezcan en el intervalo [0,1]. La motivación de esta propuesta se vincula a la fuerte relación existente entre la escala del problema y los valores de feromona resultantes en los componentes. Se ha comprobado en la práctica que un mismo algoritmo ACO puede tener resultados distintos al ser aplicado sobre un mismo problema, teniendo como única diferencia que la función objetivo sea multiplicada por una constante [20].

Blum et al. han estudiado el conjunto de los valores de feromona a partir del vector $\overrightarrow{\tau} = (\tau_{i,x_i}), \quad \forall \tau_{i,x_i} \in T$. Si se considera el problema de optimización combinatoria subyacente como un problema de decisión (0-1) o como un problema de programación entera y se relaja permitiendo valores en el intervalo $[0, 1]$ para las variables de decisión, $\overrightarrow{\tau}$ es una solución al nuevo problema propuesto. De este modo, los cambios en $\overrightarrow{\tau}$ implican desplazamientos en el hiperespacio de las soluciones del problema relajado.

Blum et al. han calculado fórmulas para realizar la actualización de la feromona de forma de cumplir con la premisa que los valores pertenezcan al intervalo [0,1] para las variantes AS y $\mathcal{MM}$AS [22]. En las expresiones de las ecuaciones 3.17 y 3.18 se presentan las fórmulas de actualización para el AS y en las expresiones de las ecuaciones 3.19 y 3.20 las de $\mathcal{MM}$AS.

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \rho \sum_{a \in A} \Delta\tau_{i,x_i}^{s_a} \quad \forall \tau_{i,x_i} \in T \tag{3.17}$$

$$\Delta\tau_{i,x_i}^{s_a} = \begin{cases} \frac{\frac{1}{F(s_a)}}{\sum_{l \in A} \frac{1}{F(s_l)}} & \text{si } c_{i,x_i} \in s_a \\ 0 & \text{en otro caso} \end{cases} \tag{3.18}$$

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \rho \Delta\tau_{i,x_i}^{s_{best}} \quad \forall \tau_{i,x_i} \in T \tag{3.19}$$

$$\Delta\tau_{i,x_i}^{s_{best}} = \begin{cases} 1 & \text{si } c_{i,x_i} \in s_{best} \\ 0 & \text{en otro caso} \end{cases} \tag{3.20}$$

Una ventaja de aplicar este propuesta sobre la variante $\mathcal{MM}$AS es que no requiere recalcular τ_{min} y τ_{max} al encontrarse una mejor solución, ya que sus valores se conocen a priori (son 0 y 1, respectivamente). Se recomienda utilizar para τ_{init} el valor 0.5, que da la misma importancia a incluir o no incluir la componente [20].

Adicionalmente Blum y Dorigo presentan un mecanismo para detectar la convergencia que consiste en contar cuántas componentes han alcanzado el valor de τ_{min} o τ_{max}. Cuando un porcentaje alto de las componentes han alcanzado uno de esos valores se produce la reinicialización de los rastros al valor τ_{init} [20]).

3.6.1. Evaluación

La principal dificultad que presenta la utilización de HCF-ACO es hallar las expresiones que permitan actualizar los rastros de feromona para una variante, de forma que los valores se mantengan en el intervalo [0,1]. Blum et al. han obtenido las expresiones que permiten realizar las actualizaciones de los rastros de feromona para las variantes AS, ACS y $\mathcal{MM}$AS [20]. La propuesta presenta como ventaja que al trabajar con valores normalizados se vuelve más explícita la relación existente entre la cantidad de feromona depositada sobre una componente y su utilización para la construcción de soluciones.

Blum y Dorigo reportan resultados de la comparación entre AS y AS con HCF-ACO sobre el Unconstrained Binary Quadratic Programming (UBQP), mostrando que los resultados de la nueva variante son independientes del rango de los valores de las funciones objetivos de la instancia del problema considerado [20]. Los autores también realizan comparaciones sobre el mismo problema entre $\mathcal{MM}$AS con la actualización de HCF-ACO, TS y SA, obteniendo la variante de ACO los mejores resultados.

El código fuente de HCF-ACO está disponible *The Ant Colony Optimization Home Page* [61].

3.7. Best-Worst Ant System (BWAS)

El algoritmo Population-Based Incremental Learning (PBIL), de la familia de los algoritmos evolutivos, presenta fuertes similitudes con los algoritmos del esquema general ACO, como se señaló en la sección 2.3.1. Ambos algoritmos utilizan un modelo probabilístico que es adaptado con el paso de las iteraciones, a partir del cual se generan las soluciones. En ambos casos la adaptación del modelo se produce considerando como retroalimentación la calidad de las soluciones generadas. Por otro lado, la incorporación de ideas presentes en algoritmos evolutivos sobre los algoritmos ACO ha probado ser positiva, como por ejemplo en EAS y AS_{rank}. La variante Best-Worst Ant System (BWAS) fue formulada por Cordón et al. en 2000 [41] y propone la incorporación en ACO de mecanismos presentes en PBIL.

Las principales diferencias entre BWAS y AS son las siguientes:

1. **Actualización de la feromona:** en BWAS en cada iteración solamente se deposita feromona sobre las componentes de la solución asociada a una hormiga. La solución considerada para el depósito de feromona es la mejor hasta el momento. La evaporación de feromona se realiza sobre todas las componentes aunque reciben una penalización extra las componentes que están presentes en la peor solución de la iteración y que no están en la mejor solución hasta el momento. El mecanismo

de actualización de la feromona resultante se presenta en las expresiones de las ecuaciones 3.21 y 3.22, siendo s_{wi} la peor solución de la iteración.

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \Delta\tau_{i,x_i}^{s_{bs}} \quad \forall \tau_{i,x_i} \in T \tag{3.21}$$

$$\tau_{i,x_i} \leftarrow (1-\rho)\tau_{i,x_i} \quad \forall \tau_{i,x_i} \in T | c_{i,x_i} \in s_{wi} \wedge c_{i,x_i} \notin s_{bs} \tag{3.22}$$

La regla para el cálculo de la cantidad de feromona a depositar se mantiene incambiado de la variante AS presentada en la ecuación 3.3.

2. **Incorporación de la mutación del rastro de feromona:** se introduce un mecanismo para incorporar diversidad, que no está presente en ninguna otra variante, la mutación de los rastros de feromona. El mecanismo está diseñado de forma de provocar pequeñas variaciones en las primeras etapas de la búsqueda, y grandes variaciones en las últimas etapas. En etapas avanzadas, cuando el algoritmo ha convergido, se provocan saltos en el espacio de búsqueda para encontrar zonas no visitadas, fomentando así la exploración.

   Cada componente de la matriz de feromona es mutada con probabilidad P_m de acuerdo a la expresión de la ecuación 3.23, siendo a un valor aleatorio en $\{0, 1\}$, it el número de la iteración que se está ejecutando y $\tau_{threshold}$ el promedio del rastro de las componentes que forman parte de la mejor solución presentada en la ecuación 3.24. Al restarse un valor al rastro actual de feromona podría ocurrir que el valor resultante sea negativo, debiéndose en tal caso corregir el valor a 0.

$$\tau_j = \begin{cases} \tau_j + mut(it, \tau_{threshold}), & \text{si } a = 0 \\ \tau_j - mut(it, \tau_{threshold}), & \text{si } a = 1 \end{cases} \tag{3.23}$$

$$\tau_{threshold} = \frac{\sum\limits_{c_{i,x_i} \in s_{bs}} \tau_{i,x_i}}{|s_{bs}|} \tag{3.24}$$

   La función mut se define en la expresión de la ecuación 3.25, siendo Nit el máximo número de iteraciones del algoritmo, it_r la última iteración en la que se produjo una reinicialización del rastro de feromona y σ un parámetro que permite ajustar el peso máximo de la mutación. Dicha función cumple con la propiedad de volver a su valor inicial cada vez que se produzca un reinicio.

$$mut(it, \tau_{threshold}) = \frac{it - it_r}{Nit - it_r} * \sigma * \tau_{threshold} \tag{3.25}$$

3. **Reinicialización del rastro de feromona:** en BWAS se reinicializa el rastro de feromona a τ_{init} cuando se detecta el estado de estancamiento. El criterio propuesto originalmente para detectar el estancamiento es que el porcentaje de componentes diferentes entre la mejor y la peor solución generada en la iteración sea menor que un porcentaje específico [41]. En la práctica este criterio no producía la reinicialización al utilizar instancias con gran cantidad de componentes ya que no se alcanzaba el umbral. Posteriormente el criterio fue modificado por el de no existir mejoras en la mejor solución hasta el momento durante una cantidad específica de iteraciones [40].

La variante BWAS utiliza los mismos parámetros que la variante AS e incorpora los parámetros P_m para la probabilidad de mutación del rastro de feromona y σ para el coeficiente de ajuste del peso máximo del operador de mutación.

Existe una propuesta de hibridización con el algoritmo ACS [39].

3.7.1. Evaluación

La variante BWAS ha sido aplicada con éxito sobre el QAP [39] y el TSP [40, 41] mostrando resultados competitivos contra las variantes AS y ACS para instancias de variados tamaños.

Se ha realizado un estudio sobre la incidencia en la calidad de los resultados obtenidos de cada una de las tres características del algoritmo propuesto. Los resultados demuestran que existe un balance entre las características, y que la eliminación de cualquiera de ellas empeora la calidad de los resultados obtenidos [40].

El código fuente de la variante BWAS está disponible en *The Ant Colony Optimization Home Page* [61].

3.8. Utilización de una población auxiliar

Las variantes presentadas en esta sección incorporan una población auxiliar que permite almacenar soluciones de iteraciones anteriores, adicionalmente a la población habitual que utiliza la técnica para realizar múltiples búsquedas en cada iteración. La población auxiliar se utiliza cuando se realiza la actualización del rastro de feromona, por lo cual estas variantes no requieren almacenar explícitamente la matriz de rastros de feromona, que puede ser calculada en cada una de las iteraciones a partir de la población auxiliar. La principal motivación detrás de este cambio es permitir una mejor adaptación sobre problemas dinámicos, permitiendo que se reflejen los ajustes en la matriz de feromona en forma más rápida que con el manejo tradicional.

3.8.1. Population Based ACO (P-ACO)

La primera variante que utiliza una población auxiliar fue propuesta en 2002 por Guntsch y Middendorf, y se llama Population Based ACO (P-ACO) [92]. La idea principal de P-ACO es mantener una población auxiliar de soluciones P de tamaño k (más pequeña que la población de hormigas) que almacene las mejores soluciones generadas en las iteraciones pasadas. Inicialmente P está vacía, y en cada iteración, cuando todas las hormigas han construido su respectiva solución, se realiza la actualización de P mediante la incorporación de la mejor solución de la iteración. Si como resultado de la actualización P contiene $k+1$ soluciones, la solución más antigua de P es borrada. Posteriormente Guntsch y Middendorf evaluaron otras opciones para seleccionar la solución a retirar de la población [93]. Los criterios evaluados fueron: edad (la más antigua) y calidad (la peor); y calidad y probabilística (a partir de la calidad de las soluciones) considerando también como posible solución a retirar, la que se está insertando en la población. Sin embargo, solamente se estudió el efecto de los criterios sobre problemas de optimización combinatoria dinámicos.

P-ACO, como los otros algoritmos ACO, utiliza una matriz de feromona a partir de la cual las hormigas construyen las soluciones. La diferencia fundamental consiste en que en P-ACO dicha matriz se calcula en cada iteración a partir de P. En P-ACO se

inicializa el valor de los rastros de feromona con el valor $\tau_{init} > 0$ y se incrementa el valor de los rastros correspondientes a las componentes que están presentes en las soluciones de la población P. La expresión para calcular el rastro de feromona se presenta en la ecuación 3.26, siendo ζ_{i,x_i} el número de soluciones en P que contienen la componente c_{i,x_i}.

$$\tau_{i,x_i} = \tau_{init} + \zeta_{i,x_i} * \Delta \quad \forall \tau_{i,x_i} \in T \tag{3.26}$$

El manejo del rastro de feromona en P-ACO discretiza la cantidad de feromona de una forma conocida a priori, e independiente de la calidad de cada una de las soluciones consideradas. Los valores posibles del rastro de feromona son $\tau_{init}, \tau_{init} + \Delta, \ldots, \tau_{init} + k * \Delta$. Debido a la forma en que se maneja el rastro de feromona no es necesario un mecanismo explícito de evaporación. La cantidad a depositar Δ se calcula en función del valor máximo de feromona para cada una de las componentes τ_{max}, de acuerdo a la siguiente expresión $\Delta = \frac{\tau_{max} - \tau_{init}}{k}$.

Una de las premisas seguidas al momento de diseñar la variante P-ACO fue reducir el tiempo de ejecución asociado al calculo de la actualización de la matriz de feromona [100]. En el resto de las variantes, la cantidad de feromona a depositar se determina a partir de la calidad de cada una de las soluciones consideradas, siendo necesario realizar operaciones para cada solución. En P-ACO en cada iteración solamente una solución puede entrar y otra puede salir de la población auxiliar, por tanto para evitar recalcular en cada iteración toda la matriz de feromona, se almacena la matriz de feromona aunque estén almacenadas las soluciones que permiten su cálculo directo. La actualización del rastro de feromona si se almacena la matriz requiere solamente $2n$ operaciones, ya que se debe sumar Δ sobre los componentes de la solución que ingresa a la población y restar Δ sobre los componentes de la solución que se elimina de la población.

P-ACO utiliza la misma regla de transición de estados de ACS que realiza un balance explícito entre la explotación determinística y la exploración probabilística incorporando el parámetro q_0, presentada en la ecuación 3.8.

La variante P-ACO utiliza los mismos parámetros que la variante AS (excepto ρ) e incorpora los parámetros q_0 para la regla de transición de estados, k para el tamaño de la población auxiliar de hormigas y τ_{max} para la cantidad máxima de feromona en las componentes.

3.8.2. Omicron ACO (OA)

Omicron ACO (OA) fue propuesta en 2004 por Gómez y Barán estando fuertemente inspirada en P-ACO [85]. Del mismo modo que en P-ACO, se mantiene una población auxiliar P de soluciones de tamaño k con las mejores soluciones generadas en las iteraciones pasadas.

Las principales diferencias que presenta con la variante P-ACO son las siguientes:

- Se elimina el parámetro valor inicial del rastro de feromona, considerándose siempre en 1.
- La actualización del rastro de feromona incorpora el parámetro O (Omicron), que representa la cantidad máxima de feromona que se puede agregar a una componente. Sumándose $\frac{O}{k}$ por cada solución que contenga la componente. Los valores del rastro de feromona posibles son $1, 1+\frac{O}{k}, \ldots, 1+\frac{(k-1)O}{k}, 1+O$. La actualización de la matriz de rastros de feromona se realiza cada K iteraciones.

- La población auxiliar P se compone de las k mejores soluciones de las iteraciones. En cada iteración, se incorpora la mejor solución de la iteración, siempre que no esté presente en P y que sea mejor que el peor elemento de P. La población auxiliar no puede contener elementos repetidos.

- Se utiliza la regla de transición de estados de la variante AS presentada en la ecuación 3.1.

Los parámetros que utiliza la variante OA son el tamaño de la población de hormigas m, el tamaño de la población auxiliar de hormigas k, la cantidad máxima de feromona en las componentes de las soluciones O, la influencia relativa de la componente de feromona α, la influencia relativa del componente heurístico β y la cantidad de iteraciones entre las actualizaciones de la matriz de feromona K. El parámetro K puede ser eliminado y realizar la actualización en todas las iteraciones, aunque esto podría aparejar la degradación del desempeño computacional.

3.8.3. Evaluación

La utilización de una población auxiliar para realizar la actualización del rastro de feromona es una idea relativamente reciente dentro de la comunidad científica y todavía no muy difundida, aunque presenta varios aspectos que hacen atractivo su estudio.

Un primer aspecto de interés consiste en que se visualiza en forma más clara la incidencia real que tienen las soluciones en los valores resultantes en la matriz de feromona, ya que la población auxiliar es una fotografía de qué soluciones se consideraron para su construcción. En las variantes que no utilizan una población auxiliar es más complejo interpretar la incidencia de las soluciones sobre la matriz de feromona resultante. Las variantes que utilizan una población auxiliar presentan parámetros cuya función en el algoritmo resulta fácil de interpretar.

Otro aspecto interesante sobre utilizar una población auxiliar es que presenta algunas similitudes con el enfoque poblacional de GA. En ambos casos, la población resume la experiencia adquirida durante la búsqueda. Esta similitud es una ventaja al momento de realizar implementaciones paralelas, pudiendo adoptarse esquemas de paralelismo similares a los utilizados por GA. La similitud también resulta ventajosa para una posible hibridación con otras técnicas poblacionales, ya que es más natural realizar intercambios con las soluciones de la población auxiliar que hacerlo con las soluciones construidas en la iteración.

P-ACO presenta una mejor eficiencia computacional para la actualización de los valores de los rastros de feromona que las variantes que no utilizan una población auxiliar, ya que requiere solamente $2n$ operaciones. Sin embargo, si se utilizan criterios distintos a la antigüedad como política de reemplazo en la población auxiliar de P-ACO o si se utiliza la variante OA, la eficiencia computacional se degrada, resultando complejo realizar un análisis general.

Un aspecto que podría resultar cuestionable de las variantes que utilizan una población auxiliar es que no utilizan en forma explícita la calidad de las soluciones para calcular el rastro de feromona. Sin embargo, las soluciones que se consideran en la población auxiliar son las últimas k mejores (para la variante P-ACO) o las mejores k (para la variante OA), con lo cual se asegura que la contribución es realizada por una solución de buena calidad. En algunas de las variantes que no utilizan una población auxiliar, la actualización es realizada por todas las hormigas que construyen soluciones,

sean éstas de buena o mala calidad, por lo cual es imprescindible que la cantidad de feromona dependa de la calidad de las soluciones consideradas.

Guntsch y Middendorf realizaron estudios comparativos entre P-ACO, EAS y $\mathcal{MM}$AS sobre los problemas TSP y QAP [92]. Los autores señalaron las dificultades existentes para comparar en forma justa dos variantes que tengan diferentes parámetros. La metodología usada por los autores para realizar la comparación consiste en elaborar un ranking de acuerdo a los resultados obtenidos para todas las configuraciones de cada una de las variantes con un criterio de esfuerzo prefijado y sin utilizar operadores de búsqueda local. Posteriormente, realizaron una clasificación de acuerdo a los parámetros comunes de ambas variantes que se están evaluando, obteniendo la posición promedio del ranking para cada una de las variantes y clasificaciones. A partir de la metodología de evaluación propuesta, Guntsch y Middendorf comprobaron que en general el promedio en el ranking de P-ACO es mejor que el EAS y $\mathcal{MM}$AS, concluyendo que los resultados obtenidos por P-ACO son en promedio por lo menos de similar calidad a los obtenidos por EAS y $\mathcal{MM}$AS.

Posteriormente, Guntsch y Middendorf ampliaron su estudio, considerando problemas de optimización combinatoria dinámicos (TSP y QAP dinámicos) [93]. P-ACO mostró un gran poder de adaptación ante cambios en las condiciones del problema.

Gómez y Barán realizaron estudios comparativos entre OA y $\mathcal{MM}$AS sobre el TSP [85]. Los estudios comparativos fueron realizados siguiendo un criterio de esfuerzo prefijado y sin incorporar operadores de búsqueda local. La evaluación realizada por los autores solamente incluyó un par de instancias del TSP, siendo los resultados obtenidos por OA superiores a los de $\mathcal{MM}$AS.

3.9. Utilización de soluciones parciales

En las variantes presentadas en las secciones anteriores, cuando cada hormiga construye una solución al problema, en el procedimiento *constructAntSolution*, comienza con una solución vacía. En cada paso de la construcción de la solución se incrementa en una componente la solución parcial construida, hasta obtener una solución completa al problema. En general la solución completa es factible, ya que no se suele trabajar en ACO con soluciones no factibles. Las variantes que se presentan en esta sección parten de una premisa diferente, comenzar la construcción de la solución completa a partir de porciones de soluciones generadas anteriormente.

Las variantes presentadas en las secciones anteriores suelen tener una alta explotación de las mejores soluciones encontradas, que puede ocasionar un estancamiento prematuro en la evolución de la calidad de las soluciones, por lo que algunas de las variantes incorporan mecanismos explícitos para evitar ese tipo de inconvenientes. Las variantes que se presentan en esta sección deben considerar especialmente esta posibilidad, ya que utilizan partes de soluciones ya construidas, por lo cual deberán incluir mecanismos que aseguren un correcto balance entre exploración y explotación.

3.9.1. Variantes con memoria externa de Acan

La primera propuesta sobre la incorporación de soluciones parciales generadas previamente en la etapa de construcción de soluciones fue realizada por Acan en 2004 [5]. La idea esencialmente consiste en mantener una memoria externa a la población de hormi-

gas, en la cual se almacenan segmentos provenientes de soluciones de buena calidad, obtenidos en etapas anteriores de la búsqueda.

El mecanismo de funcionamiento puede dividirse en dos etapas, la primera para cargar la memoria externa y la segunda, de funcionamiento en régimen.

En la primera etapa se comienza con una memoria externa de M segmentos de soluciones, que inicialmente está vacía. Se ejecutan sucesivas iteraciones en las cuales se construyen soluciones mediante el procedimiento *constructAntSolution* hasta completar los M segmentos del siguiente modo. En cada una de las iteraciones, se consideran las k mejores soluciones de la iteración y para cada una de ellas se selecciona en forma aleatoria un segmento de largo variable que se incorpora a la memoria. También se almacena en la memoria el largo del segmento y el costo asociado a la solución de la cual provino.

En el algoritmo 3 se presenta el seudocódigo de la segunda etapa, que corresponde al funcionamiento en régimen.

Algoritmo 3 Funcionamiento en régimen de la variante de Acan

```
while not stopCriteria() do
   pop = empty()
   for all a ∈ A do
      s^p = tournamentSelection(M)
      s_a = constructAntSolution(s^p,T,H)
      pop = pop + s_a
   end for
   T = updatePheromone(T,pop)
   M = updateExternalMemory(M,pop)
end while
return s_best
```

En el algoritmo 3, el procedimiento *tournamentSelection* realiza la selección de la solución parcial mediante un torneo entre Q segmentos. El procedimiento *constructAntSolution* incorpora como parámetro la solución parcial a partir de la cual se realiza la construcción, que se hace desde el fin del segmento considerado y podría ser realizada mediante cualquiera de las variantes de ACO. Acan utiliza una variante de $\mathcal{MM}$AS que incorpora una regla de transición de estados del estilo de ACS, con balance explícito entre explotación y exploración controlada [5]. Finalmente, el procedimiento *updateExternalMemory* realiza la actualización de la memoria externa, considerando las k mejores soluciones generadas en la iteración y realizando el corte de segmentos, del mismo modo que en la etapa inicial. En este caso se debe decidir qué segmentos pertenecientes a la memoria deben ser sustituidos por los nuevos. Para cada segmento nuevo se consideran primero aquellos elementos de la memoria que tengan un costo superior, sustituyendo al peor de ellos. Si no existieran segmentos con costo más alto que el nuevo segmento considerado, se concatena el segmento con el elemento de mayor costo para obtener diversidad; si se repiten componentes se procede a su eliminación.

Posteriormente, Acan realizó una segunda propuesta con algunas diferencias con la anteriormente presentada [6]. Los principales aspectos en los que se diferencia con su propuesta original son los siguientes:

1. **Utilización de permutaciones:** la memoria externa utiliza permutaciones, en

lugar de segmentos de soluciones, liberándose la restricción original de que las componentes de las soluciones parciales sean contiguas. Se almacena el costo asociado a la solución de la cual provino y un tiempo de vida de la permutación. Cuando una permutación supera su tiempo de vida es eliminada de la memoria externa.

2. **Selección por torneo:** se realiza a partir de un puntaje que vincula el costo y la edad de la permutación. El puntaje es calculado de forma que ante costos similares privilegia la selección de las permutaciones que fueron incluidas antes en la memoria, ya que están más próximas a ser borradas.

3. **Mecanismo de sustitución de elementos de la memoria:** se modifica el mecanismo de sustitución de elementos de la memoria externa, eliminándose la concatenación de elementos. Cuando no se encuentran en la memoria elementos con peor costo, se realiza la sustitución por el elemento de la memoria con peor costo dentro de los que tienen mayor edad.

La primera propuesta de Acan utiliza los mismos parámetros que la variante AS (excepto ρ) e incorpora los parámetros q_0 para la regla de transición de estados, τ_{max} para la cantidad máxima de feromona en las componentes, M para el tamaño de la memoria externa, Q para la cantidad de elementos participantes del torneo y k para la cantidad de soluciones consideradas para la memoria externa en cada iteración. La segunda propuesta de Acan adicionalmente incorpora el parámetro tiempo de vida de las permutaciones.

3.9.2. Iterated Ants (ia)

La variante Iterated Ants (ia) fue propuesta por Wiesemann y Stützle en 2006 [169] y consiste en incorporar la mecánica utilizada por la metaheurística Iterated Greedy (IG) al proceso de construcción de la solución por parte de una hormiga, cuyo seudocódigo se presenta en el algoritmo 4.

Algoritmo 4 Construcción de una solución para Iterated Ants

```
s^p = destruct(s)
s'= construct(s^p)
s'= applyLocalSearch(s')     % opcional
s = acceptanceCriterion(s,s')
return s
```

En el primer paso del algoritmo 4, el procedimiento *destruct*, a partir de una solución completa s, elimina algunas componentes obteniendo una solución parcial s^p. Existen varias alternativas para determinar el número de componentes a eliminar y para seleccionarlos. La selección de las componentes a eliminar se realiza en forma aleatoria, pudiendo asignarse la probabilidad de que una componente sea seleccionada en forma equiprobable para todas las componentes, proporcional al rastro de feromona presente en cada componente o inversamente proporcional al rastro de feromona presente en cada componente. La cantidad de componentes a eliminar puede ser fija, incorporando el parámetro l para la cantidad de componentes a eliminar, o variable durante la ejecución del algoritmo, aumentando o disminuyendo la cantidad de componentes de acuerdo a si se provocan o no mejoras en las soluciones.

A partir de s^p se realiza la construcción de la solución completa en el procedimiento *constuct*, pudiendo seguir cualquiera de los mecanismos utilizados en las variantes presentadas en las secciones anteriores. En particular, Wiesemann y Stützle utilizan el mecanismo de la variante $\mathcal{MM}$AS.

Es posible incorporar una búsqueda local que permita mejorar la solución generada mediante el procedimiento *applyLocalSearch*. Finalmente, en el procedimiento *acceptanceCriterion* se aplica un criterio de aceptación para decidir cuál es la solución que efectivamente se considera construida por la hormiga.

El manejo de los rastros de feromona de ia podría ser cualquiera de los utilizados por las variantes presentadas en las secciones anteriores. En particular, Wiesemann y Stützle utilizan el manejo de los rastros de feromona de la variante $\mathcal{MM}$AS. La utilización del mecanismo de construcción de soluciones y del manejo de los rastros de feromona de la variante $\mathcal{MM}$AS por parte de la variante ia llevó a los autores a darle el nombre Iterated Ants $\mathcal{MAX} - \mathcal{MIN}$ Ant System (ia$\mathcal{MM}$AS) a la propuesta. Naturalmente, los parámetros que utiliza ia$\mathcal{MM}$AS son los mismos que utiliza la variante $\mathcal{MM}$AS.

3.9.3. Cunning Ants (cAS)

La variante Cunning Ants (cAS) fue propuesta por Tsutsui en 2006 [159] y consiste en que cada hormiga genere la nueva solución usando partes de una solución generada en una iteración previa. La idea incorpora el manejo de dos tipos de hormigas: las astutas (cunning ants, *c-ants*) y las donantes (donor ants, *d-ants*). Las *c-ants* se apropian de parte de las soluciones construidas por las *d-ants*, completando la solución con el mecanismo usual.

La colonia de hormigas se organiza en m unidades que contienen una única solución. En cada iteración y en cada una de las unidades, una *c-ant* genera una nueva solución utilizando a la solución de la unidad como *d-ant*. Posteriormente se compara la nueva solución generada con la ya existente en la unidad, conservándose la mejor como la solución de la unidad. En un sentido amplio puede considerarse como una variante que incorpora una memoria adicional (las soluciones en cada una de las unidades) a la memoria que usualmente utilizan este tipo de algoritmos.

La actualización del rastro de feromona utiliza las soluciones presentes en cada una de las unidades de acuerdo a la expresión de la ecuación 3.27. La regla para el cálculo de la cantidad de feromona a depositar se mantiene incambiada respecto a la de AS, presentada en la ecuación 3.3.

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \sum_{a \in A} \Delta\tau_{i,x_i}^{s_a} \quad \forall \tau_{i,x_i} \in T \tag{3.27}$$

cAS incorpora los límites en el rastro de feromona τ_{min} y τ_{max} propios de $\mathcal{MM}$AS. La fórmula para calcular τ_{min}, presentada en la ecuación 3.12, no sufre modificaciones, pero la de τ_{max} se corrige para tener en cuenta el depósito de feromona por parte de todas las hormigas de la colonia de acuerdo a la expresión de la ecuación 3.28.

$$\tau_{max} = \frac{1}{\rho} * \sum_{a \in A} \frac{1}{Costo(s_a)} \tag{3.28}$$

cAS realiza algunas modificaciones sobre el algoritmo original de AS que se describen a continuación. El primer aspecto que se modifica es que la etapa de inicialización

del rastro de feromona debe incluir la generación de la solución inicial de cada una de las unidades, como se muestra en el seudocódigo presentado en el algoritmo 5. El procedimiento *constructAntSolution* debe incorporar como parámetro la solución parcial a partir de la cual se realiza la construcción. En particular, en el caso de la generación de las soluciones iniciales la solución parcial es vacía, como se muestra en el seudocódigo presentado en el algoritmo 5.

Algoritmo 5 Inicialización del rastro de feromona para cAS

```
T = initializePheromoneTrails()
pop = empty()
for all k ∈ U do
   s'= constructAntSolution(empty(),T,H)
   s''= constructAntSolution(empty(),T,H)
   s_k := best(s',s'')
   pop = pop + s_k
end for
T = updatePheromone(T,pop)
```

Un segundo aspecto que se modifica, es la forma en la cual la colonia de hormigas construye las nuevas soluciones. En el seudocódigo presentado en el algoritmo 6 se refleja el manejo de las *c-ants* y las *d-ants*. Tsutsui presenta y utiliza una función densidad de probabilidad para determinar el número de componentes que se toman de la solución anterior en el procedimiento *borrowSolution*. La función de densidad requiere la incorporación de un parámetro adicional (γ).

Algoritmo 6 constructAntsSolutions para cAS

```
for all k ∈ U do
   d-ant := s_k
   c-ant := borrowSolution(d-ant)
   s'= constructAntSolution(c-ant,T,H)
   s_k := best(s_k, s')
end for
```

Los parámetros que utiliza la variante cAS son los mismos que utiliza la variante $\mathcal{MM}$AS, incorporándose el parámetro γ para la función de densidad para determinar el número de componentes que se toman de la solución anterior.

3.9.4. Evaluación

Las variantes de ACO que incorporan la utilización de soluciones parciales en el proceso de construcción de la solución son muy recientes, y en algunos casos no han completado su maduración. En consecuencia, todavía no es posible realizar una valoración definitiva sobre la eficacia de la propuesta.

Acan realizó estudios comparativos entre sus propuestas y $\mathcal{MM}$AS sobre TSP [5] y QAP [5, 6]. Los resultados reportados por el autor para las variantes con memoria externa de Acan son superiores a los obtenidos por su implementación de $\mathcal{MM}$AS, aunque difieren de los mejores resultados reportados con $\mathcal{MM}$AS por otros autores (en algunos casos incorporando operadores de búsqueda local). Wiesemann y Stützle

señalan, a partir de los resultados presentados por Acan, que la versión implementada por Acan de $\mathcal{MM}$AS contra la que evaluó sus propuestas no era una buena versión [169]. Independientemente de las posibles deficiencias en la implementación de Acan de $\mathcal{MM}$AS, resulta cuestionable la utilización de operadores de búsqueda local cuando se realiza la evaluación entre dos propuestas ya que no se comparan solamente las propuestas. La propuesta de Acan cuenta con el mérito de haber sido la primera en incorporar la construcción de soluciones a partir de soluciones parciales obtenidas en iteraciones anteriores.

Wiesemann y Stützle realizaron estudios comparativos entre ia$\mathcal{MM}$AS y $\mathcal{MM}$AS sobre el QAP, incorporando en ambas variantes operadores de búsqueda local [169]. La evaluación experimental determinó que la utilización de soluciones parciales no pudo igualar la calidad de los resultados obtenidos por $\mathcal{MM}$AS. Sin embargo, los autores señalan que es promisoria la utilización de soluciones parciales para la construcción de soluciones en problemas para los que no se conocen buenos algoritmos de búsqueda local. Finalmente, los autores constataron que para un tiempo de ejecución fijo, ia$\mathcal{MM}$AS logra construir una mayor cantidad de soluciones que $\mathcal{MM}$AS. Este hecho resulta natural debido a que ia$\mathcal{MM}$AS se caracteriza por reutilizar parte de la solución, mientras que $\mathcal{MM}$AS construye completamente cada solución.

Tsutsui realizó estudios comparativos entre cAS, $\mathcal{MM}$AS y ACS sobre el TSP y el ATSP para instancias de variados tamaños, comprobándose que cAS aventajaba a ambas en calidad de resultados [159]. El autor extendió sus pruebas para considerar la utilización de operadores de búsqueda local, constatándose que la calidad de las soluciones obtenidas por cAS es superior al de $\mathcal{MM}$AS y ACS. Recientemente se ha aplicado el cAS sobre el QAP [161, 162], pudiendo comprobarse que se mantienen los buenos resultados obtenidos sobre el TSP. Los excelentes resultados experimentales obtenidos sobre dos de los problemas que mayoritariamente han sido utilizados para la validación de variantes de ACO posicionan a cAS como una de las variantes más promisorias. Un aspecto señalado como positivo de la propuesta es que el mecanismo de construcción de las soluciones evita el estancamiento prematuro al realizar actualizaciones más suaves del rastro de feromona [159]. Este hecho parece contribuir directamente al éxito de cAS, ya que ia$\mathcal{MM}$AS no obtiene tan buenos resultados y las principales diferencias que tienen son la agresividad con la que se realizan las actualizaciones de los rastros de feromona y la forma de seleccionar las componentes para la solución parcial.

3.10. Algoritmos similares a ACO

Los dos últimos algoritmos que se presentan en este capítulo modifican alguno de los aspectos esenciales del funcionamiento del esquema general de ACO, al punto de no considerarse variantes de ACO. Sin embargo, estos algoritmos se caracterizan por utilizar hormigas y por tener bastantes puntos de contacto con el resto de las variantes presentadas como para ameritar su inclusión en este capítulo. El primero de los algoritmos, Hybrid Ant System (HAS), utiliza el rastro de feromona para provocar modificaciones en las soluciones y no para su construcción. El segundo de los algoritmos, Fast Ant System (FANT), utiliza números enteros para el rastro de feromona, en lugar de números reales, y no utiliza ningún mecanismo de evaporación explícita de feromona.

3.10.1. Hybrid Ant System (HAS)

El algoritmo HAS fue propuesto por Gambardella y Dorigo en 1997 [80]. Presenta similitudes con la metaheurística ILS, ya que aplica perturbaciones sobre la solución actual para posteriormente realizar una búsqueda local.

La principal diferencia que presenta HAS con el resto de los algoritmos comentados en este capítulo es que no tiene un procedimiento explícito de construcción de soluciones. Cada hormiga construye inicialmente una solución en forma completamente aleatoria, quedando la solución asociada a la hormiga. Posteriormente, en cada iteración cada una de las hormigas realiza modificaciones en la solución que tiene asociada, a partir del rastro de feromona. Para manejar el número de modificaciones que se puede realizar sobre una solución se introduce el parámetro R.

HAS introduce un nuevo parámetro q_0 similar al de la variante ACS, para realizar un balance explícito entre la explotación determinística y la exploración probabilística. La regla para la transición de estados utilizada por HAS se presenta en las expresiones de la ecuación 3.29. Los parámetros α y β no son considerados en la formulación de HAS.

$$\text{Si} \quad q \leq q_0: \qquad p(c_{i,x_i}|s^p) = \begin{cases} 1 & \text{si } c_{i,x_i} = argmax(\tau_{j,x_j}\eta_{j,x_j}) \text{ con } c_{j,x_j} \in J(s^p) \\ 0 & \text{en otro caso} \end{cases}$$

$$\text{Si} \quad q > q_0: \qquad p(c_{i,x_i}|s^p) = \begin{cases} \frac{\tau_{i,x_i}\eta_{i,x_i}}{\sum_{c_{j,x_j} \in J(s^p)} \tau_{j,x_j}\eta_{j,x_j}} & \text{si } c_{i,x_i} \in J(s^p) \\ 0 & \text{en otro caso} \end{cases} \tag{3.29}$$

La actualización del rastro de feromona es realizada solamente por una hormiga por iteración, de acuerdo a la expresión de la ecuación 3.30. La regla para el cálculo de la cantidad de feromona a depositar se mantiene incambiada respecto a la de la variante AS, presentada en la ecuación 3.3.

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \rho\tau_{i,x_i}^{s_{bs}} \quad \forall \tau_{i,x_i} \in T \tag{3.30}$$

Finalmente, HAS incorpora un mecanismo de diversificación que consiste en la reinicialización de los rastros de feromona cuando se detecta convergencia.

Los parámetros que utiliza HAS son: el tamaño de la población de hormigas m, la cantidad inicial de feromona en las componentes de las soluciones τ_{init}, la tasa global de evaporación de la feromona ρ, la cantidad de modificaciones que se realizan sobre las soluciones R y q_0 para el balance entre las reglas de transición de estados.

3.10.2. Fast Ant System (FANT)

El algoritmo FANT fue propuesto por Taillard y Gambardella en 1997 [154, 155]. Con el objetivo de mejorar el desempeño de los algoritmos de optimización basados en colonia de hormigas, en FANT se simplifica el mecanismo de actualización de los rastros de feromona.

FANT se diferencia de AS fundamentalmente en los siguientes aspectos:

1. **No utilizar una colonia de hormigas:** el algoritmo FANT no utiliza una población de individuos sino que trabaja con un único individuo, aunque Stützle y Linke señalan que no es una característica intrínseca del algoritmo [153].

2. **Actualización del rastro de feromona:** FANT se caracteriza por utilizar números enteros para los rastros de feromona y por no incluir un mecanismo explicito de evaporación de los rastros, simplificando en forma significativa las operaciones que se deben realizar para la actualización de los rastros de feromona.

   El mecanismo para el manejo de los rastros de feromona requiere dos parámetros: r, la cantidad de feromona que se agrega en las componentes de la solución que se genera en la iteración y r^*, la cantidad de feromona que se agrega en las componentes de la mejor solución hallada hasta el momento. El parámetro r^* se mantiene fijo durante la ejecución mientras que r es variable, asignándose al inicio de la ejecución $r = 1$ y $\tau_{i,x_i} = r \quad \forall \tau_{i,x_i} \in T$. Para realizar la actualización del rastro de feromona se utilizan las expresiones de las ecuaciones 3.31, 3.32 y 3.33.

$$\tau_{i,x_i} \leftarrow \tau_{i,x_i} + \Delta\tau_{i,x_i}^{s_{it}} + \Delta\tau_{i,x_i}^{s_{bs}} \quad \forall \tau_{i,x_i} \in T \tag{3.31}$$

$$\Delta\tau_{i,x_i}^{s_{it}} = \begin{cases} r & \text{si } c_{i,x_i} \in s_{it} \\ 0 & \text{en otro caso} \end{cases} \tag{3.32}$$

$$\Delta\tau_{i,x_i}^{s_{bs}} = \begin{cases} r^* & \text{si } c_{i,x_i} \in s_{bs} \\ 0 & \text{en otro caso} \end{cases} \tag{3.33}$$

   Existen dos excepciones en las cuales no se actualiza el rastro de feromona de acuerdo a las ecuaciones presentadas:

   - cuando se mejora s_{bs}, se asigna $r = 1$ y $\tau_{i,x_i} = r, \quad \forall \tau_{i,x_i} \in T$, con el objetivo de intensificar la búsqueda en las cercanías de s_{bs}.
   - cuando la solución construida en la iteración coincide con la mejor solución hasta el momento ($s_{it} = s_{bs}$), se incrementa r en una unidad y se asigna $\tau_{i,x_i} = r, \quad \forall \tau_{i,x_i} \in T$, con el objetivo de diversificar la búsqueda.

3. **Regla de transición de estados:** se modifica la regla de transición de estados eliminando la componente heurística y los parámetros asociados. La regla resultante se presenta en la ecuación 3.34.

$$p(c_{i,x_i}|s^p) = \begin{cases} \dfrac{\tau_{i,x_i}}{\sum_{c_{j,x_j} \in J(s^p)} \tau_{j,x_j}} & \text{si } c_{i,x_i} \in J(s^p) \\ 0 & \text{en otro caso} \end{cases} \tag{3.34}$$

El único parámetro que utiliza FANT es la cantidad de feromona que se agrega en las componentes de la mejor solución hasta el momento r^*.

3.10.3. Evaluación

El algoritmo HAS fue aplicado originalmente sobre el Secuential Ordening Problem (SOP), obteniendo muy buenos resultados en cuanto a calidad y desempeño [80], incluso logrando mejorar las mejores soluciones conocidas sobre algunas instancias. Posteriormente se aplicó sobre el QAP, mostrando buenos resultados comparativos sobre instancias reales, pero siendo superado ampliamente en instancias aleatorias por TS y GA [81]. Años después se intentó con poco éxito utilizar HAS sobre el problema TSP [153], comprobándose que se obtenían mejores resultados realizando perturbaciones aleatorias que a partir de la matriz de feromona.

El algoritmo FANT fue aplicado originalmente sobre el QAP, demostrando buenos resultados para ejecuciones cortas [155]. Sin embargo, al incrementarse el tiempo de ejecución sus resultados son superados por los obtenidos por otras metaheurísticas e inclusive por HAS. Taillard y Gambardella señalan que FANT presenta muy buenos resultados para instancias muy grandes del QAP, al tener una estrategia de búsqueda muy rápida e intensa, inclusive encontrando mejores soluciones que las que se conocían para algunas instancias [155]. Al aplicar FANT sobre el problema TSP no se obtuvieron buenos resultados [153], inclusive utilizando una colonia en lugar de un único individuo como se establece en la propuesta original. El motivo para el mal desempeño en el TSP es que se produce una rápida convergencia a soluciones no óptimas.

3.11. Conclusiones

Ant Colony Optimization es una metaheurística que se ha consolidado en los últimos años, transformándose en una alternativa válida al momento de resolver un problema de optimización combinatoria. Ha demostrado ser competitiva con otras propuestas al ser aplicada sobre varios problemas estándares de optimización combinatoria, como por ejemplo el TSP, el ATSP y el QAP. En los últimos años se ha consolidado el estudio teórico de ACO, existiendo pruebas de convergencia para algunos algoritmos específicos que instancian el esquema general de ACO [68]. Sin embargo, este tipo de pruebas no brindan herramientas concretas que puedan ser utilizadas cuando se aplica ACO para la resolución de un problema específico.

En este capítulo se ha realizado una amplia reseña de las distintas variantes de ACO para la resolución de problemas estáticos de optimización combinatoria. Se incluyeron las propuestas más difundidas, estudiadas y utilizadas, así como nuevas propuestas de reciente formulación. En la tabla 3.1 se presenta un resumen de las características más importantes de cada una de las variantes descritas en este capítulo.

Tabla 3.1: Características de las variantes de ACO

Variante	Características
AS	Utiliza la regla de transición de estados proporcional aleatoria. Todas las hormigas de la colonia depositan feromona proporcionalmente a la calidad de la solución construida. La evaporación se produce sobre todas las componentes.
EAS	Es idéntica a AS con un depósito adicional de feromona en la mejor solución construida en la iteración.

Tabla 3.1: Características de las variantes de ACO (cont.)

Variante	**Características**
AS_{rank}	Utiliza la regla de transición de estados de AS. Se ordenan las soluciones construidas por las hormigas en un ranking, depositando feromona solamente las mejores hormigas. El depósito de feromona se pondera de acuerdo a la posición que ocupa la solución en el ranking de soluciones. Incorpora un depósito adicional de feromona en las componentes de la mejor solución. El depósito de feromona es proporcional a la calidad de la solución construida. La evaporación se produce sobre todas las componentes.
ACS	Utiliza la regla de transición de estados proporcional pseudoaleatoria. Solamente se deposita y evapora feromona en las componentes de la mejor solución. El depósito de feromona es ponderado y proporcional a la calidad de la solución construida. Incorpora una actualización local del rastro de feromona cuando una hormiga utiliza una componente.
$\mathcal{MM}$AS	Utiliza la regla de transición de estados de AS. Solamente se deposita feromona sobre las componentes de la mejor solución, en forma proporcional a su calidad. Incorpora límites superior e inferior para la cantidad de feromona que puede haber en una componente. La evaporación se produce sobre todas las componentes.
ANTS	Utiliza una regla de transición de estados propia en la que el valor asociado a la información heurística se calcula como una cota inferior del costo de completar la solución parcial. Todas las hormigas actualizan el rastro de feromona. El mecanismo de actualización del rastro de feromona es propio de ANTS y se caracteriza por no utilizar una evaporación explícita.
HCF-ACO	Es una propuesta de escalado de los valores de los rastros de feromona que puede ser aplicada sobre cualquier variante.
BWAS	Utiliza la regla de transición de estados proporcional aleatoria de AS. Solamente se deposita feromona sobre las componentes de la mejor solución proporcionalmente a su calidad. La evaporación se produce en todas las componentes. Incorpora una evaporación adicional de feromona en las componentes de la peor solución que no forman parte de la mejor solución. Incorpora un operador de mutación sobre los rastros de feromona.
P-ACO	Utiliza una población auxiliar para almacenar las mejores soluciones, a partir de la que se calcula el rastro de feromona. La cantidad de feromona depositada no depende de la calidad de las soluciones. No utiliza un mecanismo explicito de evaporación. Utiliza la regla de transición de estados de AS.
OA	Utiliza una población auxiliar para almacenar las mejores soluciones, a partir de la que se calcula el rastro de feromona. En la población auxiliar no se permiten soluciones repetidas. La cantidad de feromona depositada no depende de la calidad de las soluciones. No utiliza un mecanismo explicito de evaporación. Utiliza la regla de transición de estados de ACS.

Tabla 3.1: Características de las variantes de ACO (cont.)

Variante	**Características**
Acan	Utiliza una población externa para almacenar partes de soluciones de buena calidad. La construcción de una solución comienza seleccionando una solución parcial de la población auxiliar. Utiliza el manejo de los rastros de $\mathcal{MM}$AS y la regla de transición de estados de ACS.
ia	Cada hormiga almacena una solución. Para construir una solución se parte de la solución almacenada por la hormiga y se le quitan componentes. Utiliza el manejo de los rastros de feromona y la regla de transición de estados de $\mathcal{MM}$AS.
cAS	Cada hormiga almacena una solución. Para construir una solución se seleccionan componentes de la solución almacenada por la hormiga. Todas las hormigas depositan feromona proporcionalmente a la calidad de la solución. Incorpora límites superior e inferior para la cantidad de feromona que puede haber en una componente. La evaporación se produce sobre todas las componentes. Utiliza la regla de transición de estados de AS.
HAS	Cada hormiga almacena una solución. Utiliza una regla de transición de estados propia que se caracteriza por considerar las posibles modificaciones de las soluciones almacenadas. Solamente se deposita feromona en las componentes de la mejor solución en forma proporcional a la calidad de la solución construida. El depósito de feromona es ponderado. La evaporación se produce sobre todas las componentes.
FANT	Utiliza una única hormiga en lugar de una población. El mecanismo de actualización de los rastros de feromona se caracteriza por no utilizar un mecanismo explícito de evaporación. El valor de los rastros de feromona solamente puede tomar valores enteros. Solamente se deposita feromona en las componentes de la mejor solución hasta el momento y de la mejor solución de la iteración. Utiliza una regla de transición de estados propia.

Dentro de las variantes más antiguas, $\mathcal{MM}$AS y ACS se destacan ampliamente por sobre el resto, demostrando una gran versatilidad al haber sido aplicadas exitosamente sobre varios problemas. Las propuestas AS, EAS y AS_{rank} han sido en general superadas en la práctica por $\mathcal{MM}$AS y ACS. Otra de las variantes más antiguas, ANTS presenta como particularidad que requiere la existencia de algún mecanismo que permita obtener cotas inferiores de las soluciones para el problema abordado, con lo cual su aplicabilidad es restringida.

HCF-ACO no es exactamente una variante en si misma, sino una forma de manejar las actualizaciones de los rastros de feromona que puede ser aplicada sobre las otras variantes. El gran mérito de las ideas planteadas en HCF-ACO es que permite obtener resultados con la misma calidad en forma independiente del escalado del problema considerado.

La variante BWAS incorpora conceptos presentes en un tipo de algoritmo evolutivo (PBIL). La gran diferencia conceptual que introduce BWAS es la utilización de un operador de mutación sobre los rastros de feromona para incorporar diversidad en la

búsqueda. Sin embargo, BWAS no ha sido muy utilizada fuera del grupo de investigación que la formuló.

Las variantes que incorporan la utilización de una población auxiliar no han sido lo suficientemente explotadas por la comunidad científica. Las propuestas que utilizan una población auxiliar presentan aspectos interesantes, como ser su claridad en el manejo de los rastros de feromona y las posibilidades que brinda el manejo explícito de la población de soluciones.

La incorporación de soluciones parciales al proceso de construcción de las soluciones es la idea más reciente de las relevadas, por lo cual no se puede realizar una valoración definitiva sobre las virtudes de incorporar este tipo de mecanismos. Sin embargo, *c*AS se perfila como una variante promisoria, como consecuencia de los muy buenos resultados que ha obtenido sobre el TSP, el ATSP y el QAP.

Finalmente, HAS y FANT son propuestas que aunque presentan ideas que pueden ser consideradas interesantes, han sido poco utilizadas ya que su éxito se restringe a algún problema específico.

Capítulo 4

Paralelismo aplicado a ACO

"- Las sustancias químicas tipo Prozac no suprimen las sensaciones
sino que las descomponen en pequeñas 'unidades de sensación',
que el nuevo cerebro en paralelo procesa más deprisa desde un punto de vista informático.
- Me parece que necesito un segundo para digerir esto, Eth ...
- Yo no. El pensamiento lineal está superado. Lo último es el pensamiento en paralelo."
Douglas Coupland, *Microsiervos*

La utilización de metaheurísticas para resolver problemas de optimización combinatoria del tipo NP-difícil permite reducir significativamente el tiempo de ejecución, asegurando la obtención de soluciones cercanas al óptimo. Sin embargo, existen factores que pueden incidir para que la utilización de metaheurísticas resulte impracticable. Dos factores que pueden incidir en un aumento significativo en el tiempo de cómputo de una metaheurística para resolver un problema determinado son: que el problema sea fuertemente restringido y que se utilicen instancias de alta dimensión. Por ese motivo y a partir de las crecientes posibilidades brindadas por las arquitecturas de hardware, la aplicación de paralelismo a las metaheurísticas surge como una alternativa natural para permitir una disminución en el tiempo de ejecución. Además, las implementaciones paralelas suelen explorar el espacio de búsqueda de forma distinta a las implementaciones secuenciales, resultando en cambios en el comportamiento del algoritmo que pueden provocar mejoras significativas en la calidad de las soluciones encontradas.

La aplicación de paralelismo sobre ACO es incipiente y no ha completado su maduración. Si bien las primeras propuestas de paralelismo aplicado a ACO se remontan a los orígenes de la propia metaheurística, la investigación en esta temática ha crecido notablemente en los últimos cinco años, hecho apreciable en la cantidad considerable de artículos publicados sobre esta temática.

Las técnicas de alto desempeño comprenden a la computación paralela (varios procesos cooperan para resolver un problema en común) y la computación distibuida (procesadores independientes con autonomía de procesamiento interconectados cooperan para lograr un objetivo global). Este capítulo se divide conceptualmente en tres componentes. El primero de ellos, es la introducción de conceptos de las técnicas de alto desempeño necesarios para la comprensión del resto del capítulo. El segundo componente es la presentación de las taxonomías de paralelismo aplicado a ACO. El último componente corresponde a un relevamiento de la aplicación de paralelismo a ACO con el objetivo de brindar un resumen de las principales propuestas existentes en la literatura sobre la temática. El relevamiento de propuestas se concentra fundamentalmente en trabajos

realizados sobre clusters de PCs y equipos de memoria compartida, descartándose los trabajos vinculados a implementaciones paralelas por hardware.

El resto del capítulo se estructura de la siguiente forma. En la próxima sección se presenta una descripción de las arquitecturas de computadoras paralelas existentes en la actualidad. En la sección 4.2 se introducen algunos conceptos vinculados a las técnicas de alto desempeño que serán utilizados en las secciones siguientes. Las métricas más usadas para la evaluación de algoritmos paralelos se presentan en la sección 4.3. Las implementaciones paralelas de una metaheurística se suelen clasificar según las estrategias seguidas para la incorporación del paralelismo. Las taxonomías propuestas para la paralelización de ACO y una propuesta de ampliación se describen en la sección 4.4. Posteriormente, en la sección 4.5 se reseñan las principales aplicaciones de paralelismo a ACO. Finalmente, en la sección 4.6 se presentan las conclusiones del relevamiento.

4.1. Arquitecturas de computadoras paralelas

Varias taxonomías han sido propuestas para clasificar las computadoras paralelas. La más utilizada corresponde a Flynn [76] y se caracteriza por considerar en forma independiente los flujos de instrucciones y de datos. Para cada uno de los flujos se distingue entre únicos y múltiples, determinando cuatro categorías. A continuación se describen brevemente las características de cada categoría:

- SISD (instrucciones únicas, datos únicos): corresponde a las computadoras monoprocesadores.
- SIMD (instrucciones únicas, datos múltiples): se caracterizan por tener múltiples elementos de procesamiento que en un mismo instante de tiempo ejecutan las mismas instrucciones sobre diferentes datos. El control se realiza en forma centralizada, sin tener autonomía de procesamiento los elementos de procesamiento. Debido a las restricciones causadas por el sincronismo y por la distribución geográfica de los datos, las computadoras típicamente se utilizan para aplicaciones de propósito específico [10].
- MISD (instrucciones múltiples, datos únicos): permite ejecutar múltiples instrucciones sobre un único grupo de datos. Es un modelo poco usado en la práctica. Un ejemplo de esta arquitectura son los arreglos sistólicos, en los cuales los datos circulan desde la memoria hacia un conjunto de procesadores, que trabaja superponiendo etapas (pipeline).
- MIMD (instrucciones múltiples, datos múltiples): se caracterizan por permitir que múltiples procesadores ejecutan en cualquier instante de tiempo distintas instrucciones sobre distintos datos. El control se realiza en forma descentralizada por cada uno de los procesadores que tienen autonomía de procesamiento.

Se han propuesto refinamientos a la clasificación, considerando aspectos que no estaban presentes en la taxonomía original. En la figura 4.1 se presenta una versión ampliada de la taxonomía de Flynn [10].

Sobre la categoría SIMD puede realizarse un refinamiento que discrimine entre vector de procesadores y arreglo de procesadores. Los vectores de procesadores se caracterizan por ser unidimensionales, es decir que trabajan sobre un vector de datos, y realizan

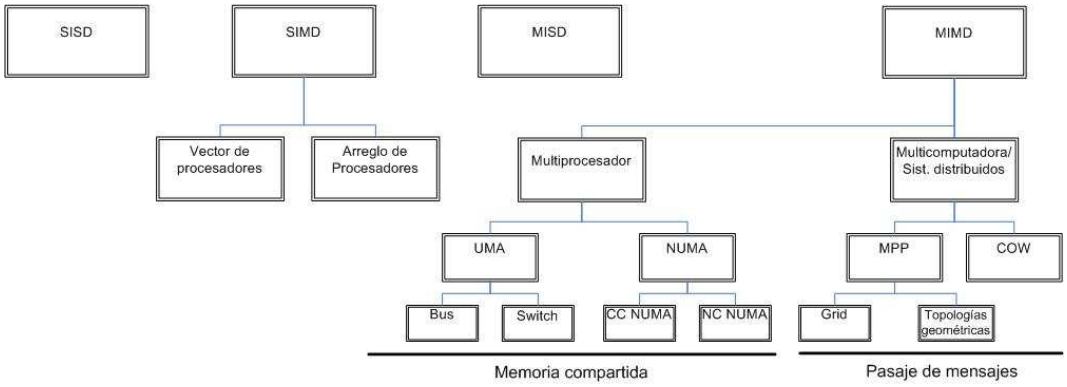

Figura 4.1: Extensión a la taxonomía de Flynn

las operaciones sobre el vector completo. En el caso de los arreglos de procesadores, su dimensión no está restringida y las operaciones se realizan sobre las componentes de los arreglos.

La categoría MIMD puede dividirse en dos, dependiendo si la memoria está compuesta por un único espacio de direcciones (memoria compartida) o si existen varios espacios diferentes (memoria distribuida). A continuación se presenta el refinamiento de la categoría MIMD:

- Multiprocesadores: son computadoras en las que todos los procesadores tienen acceso sobre el mismo espacio de memoria. También se conocen como computadoras de memoria compartida o computadoras paralelas fuertemente acopladas. La forma de comunicación y sincronización entre los procesos se realiza a través de la lectura y escritura de la memoria global. Se utilizan ampliamente en la actualidad, aunque tienen costos elevados presentan limitaciones en cuanto a la cantidad de procesadores que pueden manejar [10]. Presentan como desventaja que debido a su alto acoplamiento, las fallas en la memoria provocadas por un procesador pueden causar una falla general en todos los procesadores. Pueden existir limitaciones en la escalabilidad debido a que el bus de datos que accede a la memoria global es un cuello de botella. Pueden ser clasificadas según las características del tiempo de acceso a memoria en:
  - UMA (Uniform Memory Adress): corresponde a un tiempo de acceso a memoria constante para todos los procesadores. Dentro de la categoría UMA se incluyen los multiprocesadores simétricos (SMP, Symmetric Multiprocessor). Considerando la forma en que se realizan las conexiones entre los procesadores, es posible distinguir las computadoras paralelas conectadas mediante un bus o un switch.
  - NUMA (Non Uniform Memory Adress): corresponde a un tiempo de acceso a memoria inconstante para los procesadores. Se caracteriza porque los procesadores actúan en forma asincrónica y por incorporar cachés. La utilización de cachés introduce el problema de coherencia entre los cachés y la memoria global. Es posible realizar otro grado de refinamiento a partir de como se manejan los cachés, ya sea en forma coherente (CC-NUMA, Coherent Cache-Non Uniform Memory Adress) o no coherente (NC-NUMA, Not Coherent Cache-Non Uniform Memory Adress).

- Multicomputadoras: se caracterizan porque cada procesador dispone de una memoria independiente que solamente puede ser accedida por el propio procesador, sin existir una memoria global a todo el sistema. También se conocen como sistemas distribuidos, computadoras de memoria distribuida o computadoras paralelas débilmente acopladas. No utilizan una memoria global, sino que la forma de comunicación y sincronización entre los procesos se realiza a través del pasaje de mensajes entre los procesadores en forma explícita. Presentan como ventajas significativas su alta escalabilidad y disponibilidad, ya que un fallo en uno de los procesadores no afecta necesariamente al resto. Pueden ser clasificadas en:
  - COW (Cluster Of Workstations): está formada por los clusters de PCs, un conjunto de computadoras personales interconectadas a través de una red para comunicación de datos. El número de PCs que se pueden interconectar suele estar limitado por las características inherentes a la tecnología de red utilizada.
  - Computadoras masivamente paralelas (Massively Parallel Processor): está formada por computadoras con miles de procesadores. Puede refinarse de acuerdo a si el sistema es fuertemente acoplado o no. En el primero de los casos resultan sistemas con topologías geométricas como el hipercubo o el toro. El segundo caso corresponde a los sistemas Grid o metacomputadoras, en los que los equipos suelen pertenecer a diferentes organizaciones y estar geográficamente dispersos, utilizando la infraestructura provista por redes de área global (como Internet) para realizar la interconexión.

4.2. Conceptos de técnicas de alto desempeño

En esta sección se presenta la definición de algunos conceptos relacionados a la temática técnicas de alto desempeño que son utilizados en las secciones siguientes.

4.2.1. Granularidad

La granularidad de un algoritmo paralelo cuantifica el tamaño de las tareas individuales que deben ser realizadas por cada proceso. Cuando la granularidad es fina, el tamaño de las tareas es pequeño y por consiguiente el tiempo de procesamiento de cada tarea suele ser reducido. Cuando la granularidad es gruesa, el tamaño de las tareas es grande y el tiempo de procesamiento suele ser más elevado. En general, cuanto más pequeña es la granularidad mayor es el potencial para paralelizar el algoritmo, pero puede existir una degradación al aumentar los tiempos utilizados para sincronizar y comunicar los procesos. Los algoritmos de granularidad fina requieren una gran cantidad de comunicación entre los procesos, por lo que se suele asociarlos con implementaciones en computadoras con memoria compartida. Por otro lado, los algoritmos de granularidad gruesa necesitan una menor comunicación entre los procesos, por lo que se suele asociarlos con implementaciones en computadoras con memoria distribuida.

4.2.2. Modelos de comunicación entre procesos

Los modelos de comunicación entre procesos establecen las características de la comunicación entre los procesos paralelos. Los modelos comúnmente utilizados son: el modelo

maestro-esclavo, el modelo cliente-servidor y el modelo peer to peer. El modelo maestro-esclavo presenta un proceso distinguido (maestro) y varios procesos idénticos (esclavos). El proceso maestro controla a los procesos esclavos, asignándoles datos o tareas. Los procesos esclavos realizan el procesamiento y solamente se comunican con el maestro para enviarle los resultados de la tarea asignada. El modelo cliente-servidor presenta procesos de un tipo (clientes) que solicitan servicios a procesos de otro tipo (servidores). Los procesos servidores están permanentemente activos esperando por solicitudes de los clientes. El modelo peer to peer corresponde a un modelo de comunicación en el cual cada parte tiene las mismas capacidades para establecer la comunicación.

4.2.3. Estrategias de computación paralela para la división de problemas

La metodología de la computación paralela consiste en dividir el problema para su resolución simultánea en forma cooperativa por diferentes unidades de proceso. Para realizar la división se utilizan como estrategias: la descomposición funcional y la descomposición de dominio. La descomposición funcional corresponde a la división de las distintas tareas a efectuar por el algoritmo entre las unidades de proceso. La descomposición de dominio corresponde a la división de los datos del problema y su asignación a las unidades de proceso que realizan las mismas tareas.

4.3. Medidas de desempeño

La evaluación del desempeño de un algoritmo secuencial se suele realizar siguiendo un enfoque teórico que consiste en analizar su complejidad computacional en función del tamaño de su entrada. Sin embargo, en algunos casos (e.g.: el algoritmo es no determinístico) ese enfoque resulta impracticable. En esos casos, la evaluación del desempeño se realiza en forma empírica considerando el tiempo de ejecución del algoritmo sobre alguna plataforma específica. Evidentemente, esto implica que la evaluación del desempeño realizada de esta forma tenga una fuerte dependencia de los recursos computacionales disponibles.

Para la evaluación del desempeño de un algoritmo paralelo existe una mayor dependencia de la arquitectura subyacente y de las características propias de los recursos utilizados, por lo cual el enfoque usualmente utilizado es el empírico. Otro aspecto que incide en la utilización de un enfoque empírico es la necesidad de determinar una medida de la ganancia al utilizar un algoritmo paralelo respecto a la implementación secuencial.

La primera opción para realizar la evaluación del desempeño computacional es utilizar el tiempo de ejecución, al igual que en los algoritmos secuenciales. Sin embargo, el tiempo de ejecución depende fuertemente del tamaño del problema considerado y de la cantidad de procesadores disponibles. Al incrementarse el tamaño del problema y mantenerse fija la cantidad de procesadores, evidentemente el tiempo de ejecución aumenta. Por el contrario, si se mantiene fijo el tamaño del problema y se aumenta la cantidad de procesadores, el tiempo de ejecución suele disminuir. Este enfoque también cuenta con la desventaja de no permitir apreciar la posible ganancia provocada por la incorporación del paralelismo.

La comunidad científica ha propuesto una gran variedad de métricas para evaluar distintos aspectos del desempeño de los algoritmos paralelos. A continuación se presenta

una breve descripción de algunas de las métricas más usadas: speedup, eficiencia y eficacia.

Para evaluar el desempeño de metaheurísticas paralelas se suele realizar ajustes sobre las métricas generales considerando que se trata de algoritmos no determinísticos. La única métrica propuesta que no cumple esa premisa es el speedup para metaheurísticas paralelas [46, 148] que se describe al final de esta sección.

4.3.1. Speedup absoluto y algorítmico

El speedup es una métrica que permite apreciar el desempeño relativo al aumentar la cantidad de procesadores, en comparación con la utilización de un único procesador. Existen dos formulaciones del speedup: el absoluto y el algorítmico.

El speedup absoluto, también llamado speedup fuerte, vincula el tiempo de ejecución del mejor algoritmo serial conocido para el problema considerado (T_{serial}) y el tiempo total de ejecución del algoritmo paralelo ejecutado sobre N procesadores (T_N) como se muestra en la ecuación 4.1.

$$S_N^{abs} = \frac{T_{serial}}{T_N} \tag{4.1}$$

Esta definición es poco utilizada en la práctica, debido a que no siempre se conoce el mejor algoritmo serial para poder realizar la comparación. Se suele utilizar como alternativa el speedup algorítmico, también conocido como speedup débil. El speedup algorítmico considera el tiempo de ejecución de un algoritmo serial propio en lugar del mejor algoritmo serial. En la ecuación 4.2 se presenta su formulación, siendo T_1 el tiempo del algoritmo serial en un procesador y T_N el tiempo del algoritmo paralelo ejecutando sobre N procesadores.

$$S_N = \frac{T_1}{T_N} \tag{4.2}$$

Cuando se trabaja con algoritmos no determinísticos, como las metaheurísticas, el speedup suele considerar el tiempo promedio de ejecución del algoritmo serial y del algoritmo paralelo. Alba [10] propone no utilizar un criterio de esfuerzo prefijado para calcular el speedup algorítmico, sino un nivel fijo de calidad de resultados, distinguiendo entre la comparación entre la ejecución en paralelo y una versión secuencial (ortodoxo) o la versión paralela ejecutada sobre un procesador (versus panmixia).

Para el speedup es posible obtener resultados sublineales ($S_N < N$), lineales ($S_N = N$) y superlineales ($S_N > N$). Es deseable obtener speedups lineales porque indican que al utilizar n procesadores se obtiene un factor n de reducción del tiempo de ejecución. En la práctica existen restricciones que pueden imposibilitar la concreción de ese objetivo, como por ejemplo las demoras introducidas por retardos en las comunicaciones, el intercambio de datos y la sincronización de procesos, y la existencia en el algoritmo de tareas intrínsecamente no paralelizables.

4.3.2. Eficiencia

La eficiencia es una métrica que corresponde al valor normalizado del speedup respecto a la cantidad de procesadores utilizados como se muestra en la ecuación 4.3.

$$E_N = \frac{S_N}{N} = \frac{T_1}{N * T_N} \tag{4.3}$$

Los valores de eficiencia suelen estar entre 0 y 1, siendo los valores cercanos a 1 los deseables, ya que se corresponden con valores de speedup lineal.

4.3.3. Eficacia

La eficacia es una métrica que mide la relación costo-beneficio de incorporar un nuevo procesador a la ejecución del algoritmo paralelo. El costo de agregar un nuevo procesador se considera como el inverso de la eficiencia. Esta elección del costo no es arbitraria, sino que asigna un costo elevado a eficiencias bajas y un costo cercano al unitario a eficiencias altas (cercanas a 1). En la ecuación 4.4 se presenta la formulación de la métrica eficacia.

$$e_N = E_N * S_N = \frac{{S_N}^2}{N} \tag{4.4}$$

Un aumento en la eficacia indica que la ganancia obtenida al agregar un nuevo procesador es superior a su costo. Cuando se produce una disminución en la eficacia, el costo de agregar un procesador es superior a su beneficio. Por lo tanto, el máximo valor de la eficacia indica el mejor compromiso costo-beneficio para el algoritmo paralelo considerado.

4.3.4. Speedup para metaheurísticas paralelas

La métrica speedup para metaheurísticas paralelas permite evaluar las mejoras en el tiempo de ejecución para obtener niveles fijos de calidad de resultados al realizar ejecuciones paralelas independientes de un algoritmo secuencial [46, 148]. Se define el speedup como la relación entre el tiempo necesario para obtener una solución de calidad Q en la versión serial (T_1^Q) y en las ejecuciones paralelas independientes sobre N procesadores (T_N^Q). En la ecuación 4.5 se muestra la formulación de esta métrica.

$$S_N^Q = \frac{T_1^Q}{T_N^Q} \tag{4.5}$$

Es deseable que el speedup sea lineal, lo que indicaría que el tiempo que lleva alcanzar un cierto nivel de calidad en la solución es proporcional a la cantidad de procesadores utilizados. Esta condición se presenta cuando la distribución de la variable aleatoria "tiempo de ejecución necesario para obtener una solución de calidad Q" es exponencial [46, 148].

Esta métrica no ha sido muy utilizada en la práctica debido a que solamente es aplicable sobre una implementación particular del paralelismo, las ejecuciones paralelas independientes.

4.4. Estrategias para paralelizar la metaheurística ACO

El estudio sistemático de la aplicación de técnicas de programación paralela sobre ACO es reciente y varios autores han coincidido en que no se ha realizado un trabajo exhaustivo como para otras metaheurísticas [68, 100]. Dorigo y Stützle señalan que a

pesar de existir una cantidad importante de trabajos que abordan la implementación paralela de ACO, permanecen como problemas abiertos la implementación de versiones paralelas eficientes y el análisis del tipo de mejoras que pueden obtenerse sobre las versiones secuenciales [68]. Por otro lado, es posible detectar una escasez de propuestas de taxonomías de estrategias para la aplicación del paralelismo.

En el resto de la sección se presentan estrategias para aplicar paralelismo sobre ACO. En primer lugar se comentan brevemente algunas taxonomías generales para técnicas metaheurísticas, y seguidamente se presentan las propuestas para ACO. Sin embargo, como se muestra en este trabajo, estas propuestas no son del todo satisfactorias. Posteriormete, se comenta la taxonomía comúnmente aceptada para Evolutionary Algorithms (EA), ya que tiene la madurez necesaria para servir de base para detectar carencias en las propuestas existentes para ACO. Finalmente, se presenta una propuesta para complementar la taxonomía para ACO de Randall y Lewis, tratando de superar algunas de las carencias detectadas.

4.4.1. Taxonomías genéricas de metaheurísticas paralelas

La definición de estrategias para la aplicación de paralelismo sobre metaheurísticas intenta reflejar características generales que trascienden a una técnica en particular. Utilizar un enfoque genérico puede resultar poco flexible para incorporar particularidades, pero permite una primera conceptualización sobre las ideas que fueron utilizadas en la paralelización de varias metaheurísticas. En la práctica, las taxonomías genéricas son poco utilizadas por la comunidad académica cuando se trabaja sobre una metaheurística específica.

La clasificación propuesta por Cung et al. [46] distingue según el método de exploración entre trayectoria única y trayectorias múltiples. La categoría de trayectoria única paraleliza la búsqueda de una solución, aplicando paralelismo en la evaluación de la función de costo o en la descomposición de dominio cuando se explora un vecindario de soluciones, y por consiguiente los algoritmos resultantes tienen granularidad fina o media. La categoría de trayectorias múltiples busca en paralelo más de una solución compartiendo información entre sí (búsquedas cooperativas) o realizando búsquedas independientes. Los algoritmos resultantes tienen granularidad media o gruesa.

Por otro lado, la taxonomía propuesta por Crainic et al. [42] plantea tres dimensiones como criterios de clasificación:

- *Cardinalidad del control de la búsqueda:* cuando se trabaja con más de un proceso es necesario determinar quién y cómo controla el proceso de búsqueda de la solución. Este control puede ser realizado por un único proceso ($1C$) o en forma conjunta por varios procesos (pC).

- *Control de la búsqueda y comunicaciones:* típicamente se distingue entre comunicación sincrónica y asincrónica. Esta dimensión también incorpora la noción de conocimiento para reflejar el intercambio de información que implica conocimiento derivado de la búsqueda. De esta forma resultan cuatro clases: rígido (RS) y conocimiento-sincronizado (KS), correspondientes a una comunicación sincrónica; y colegiado (C) y conocimiento-colegiado (KC), correspondientes a una comunicación asincrónica.

- *Diferencias en la búsqueda:* en esta dimensión se refleja la utilización de las mismas soluciones y las mismas estrategias de búsqueda por los distintos proce-

sos. Se establecen cuatro clases: mismo punto/población inicial, misma estrategia de búsqueda ($SPSS$); mismo punto/población inicial, diferentes estrategias de búsqueda ($SPDS$); múltiples puntos/poblaciones iniciales, misma estrategia de búsqueda ($MPSS$); y múltiples puntos/poblaciones iniciales, diferentes estrategias de búsqueda ($MPDS$).

La taxonomía de Crainic et al. [42] es más amplia que la de Cung et al. [46], considerando algunos aspectos que no suelen ser recogidos comúnmente en las clasificaciones.

4.4.2. Taxonomía de paralelismo aplicado a ACO

Existen pocos artículos que aborden en forma concreta la discusión de las posibles estrategias para aplicar paralelismo sobre ACO. La mayoría de los trabajos que abordan el tema se limitan a realizar una propuesta concreta de implementación y a relevar otras propuestas.

El trabajo de Randall y Lewis [140] incluye una de las pocas definiciones de estrategias de paralelismo aplicado a ACO existentes. La propuesta de los autores distingue cinco categorías:

- *Colonias de hormigas paralelas independientes (parallel independent ant colonies):* corresponde a la ejecución en paralelo de un algoritmo ACO secuencial sobre un conjunto de procesadores disponibles. Las ejecuciones son completamente independientes, sin existir comunicación entre las colonias, que pueden utilizar los mismos o distintos parámetros.

- *Colonias de hormigas paralelas con interacción (parallel interacting ant colonies):* corresponde a la ejecución en paralelo de un algoritmo ACO secuencial sobre un conjunto de procesadores disponibles. Con una frecuencia fija de iteraciones, las colonias sincronizan sus matrices de feromona, obteniendo los valores de la mejor colonia hasta el momento. El volumen de datos que es necesario transmitir entre las colonias es alto ya que debe comunicarse la matriz de feromona completa.

- *Hormigas paralelas (parallel ants):* corresponde con un modelo maestro-esclavo que trabaja sobre una colonia en forma paralela. El proceso maestro se encarga del manejo global de la matriz de feromona y cada proceso esclavo construye una solución como si se tratara de una hormiga de la colonia. La comunicación involucra el envío de las soluciones (o la actualización correspondiente de la matriz de feromona) desde los esclavos al maestro y el envío de la actualización de la matriz de feromona desde el maestro a los esclavos.

- *Evaluación en paralelo de componentes de soluciones (parallel evaluation of solution elements):* corresponde a la ejecución de una colonia en forma secuencial, en la que cada hormiga evalúa en forma paralela las componentes que puede incorporar a la solución que está construyendo. Se utiliza un modelo maestro-esclavo, siendo el maestro la hormiga y los esclavos los encargados de evaluar las componentes. Randall y Lewis señalan que este tipo de estrategias puede ser recomendable para problemas con fuertes restricciones.

- *Combinación paralela de hormigas y evaluación de componentes de soluciones (parallel combination of ants and evaluation of solution elements):* corresponde

a la combinación de las estrategias hormigas paralelas y evaluación en paralelo de componentes de soluciones. El modelo implica tener dos niveles de paralelismo maestro-esclavo, uno a nivel de la colonia y las hormigas, y otro a nivel de cada hormiga y las evaluaciones de las componentes de las soluciones.

Las estrategias identificadas por Randall y Lewis presentan muchos puntos de contacto con las planteadas por Cung y sus coautores. Es posible identificar la correspondencia entre las categorías: evaluación en paralelo de componentes de soluciones y trayectoria única; hormigas paralelas y trayectoria única; colonia de hormigas paralelas independientes y trayectorias múltiples búsquedas independientes; y colonia de hormigas paralelas con interacción y trayectorias múltiples búsquedas cooperativas.

La categoría colonia de hormigas paralelas con interacción presenta una restricción demasiado fuerte: la necesidad de sincronizar todas las matrices de feromona. Esta restricción no sólo condiciona el mecanismo de búsqueda, ya que todas las colonias llevan a cabo una búsqueda similar, sino que implica la necesidad de determinar la mejor colonia hasta el momento. En el artículo no se establece cómo determinar cuál es la mejor colonia de hormigas, aunque algunos criterios simples son la colonia que ha obtenido la mejor solución hasta el momento o la colonia que obtuvo la mejor solución en la última iteración. Para determinar la mejor colonia puede requerirse un agente adicional que decida cuál es la mejor colonia, para evitar la realización de comunicaciones entre todas las colonias.

La categoría evaluación en paralelo de componentes de soluciones no ha sido llevada a la práctica y parece de difícil aplicación debido a que su granularidad es muy fina y requiere mucha comunicación entre los procesos.

Janson et al. [100] no realizan un planteo explícito de taxonomías de paralelismo aplicado a ACO. Sin embargo, presentan varios aspectos relativos a algunas implementaciones paralelas que conviene considerar. Sus detalles se comentan a continuación.

En primer lugar, los autores señalan que el proceso de creación de una solución por parte de una hormiga no suele ser separado entre varios procesadores, ya que es una tarea básicamente secuencial. Por este motivo, la granularidad mínima del algoritmo suele ser la construcción de una solución. Algunos problemas pueden requerir paralelizar el cálculo de la calidad de una solución, si es una tarea compleja, pero este hecho esta más fuertemente relacionado a dificultades particulares de un problema específico y no a características intrínsecas al modelo de paralelismo de ACO. Este aspecto del proceso de creación de las soluciones va en correlación con la dificultad señalada para llevar a la práctica la categoría evaluación en paralelo de componentes de soluciones.

En segundo término, se consideran solamente dos criterios para clasificar a los ACO paralelos. El primer criterio consiste en distinguir si el paralelismo es realizado sobre un algoritmo ACO estándar o si el algoritmo ACO paralelo está especialmente diseñado. En un caso se establece como objetivo central la disminución del tiempo de ejecución sin cambiar el comportamiento, y en el otro el aumento de la eficiencia, pudiendo modificar el comportamiento del algortimo ACO estándar. El segundo criterio corresponde a si el enfoque es centralizado o descentralizado.

En tercer lugar, los autores abordan específicamente las particularidades del modelo multicolonia, que consiste en tener varias colonias de hormigas que utilizan cada una su propia matriz de feromona. La propuesta original de dicho modelo tiene como objetivos la mejora en la calidad de los resultados y su aplicación a problemas multiobjetivo. Aunque es posible implementar una multicolonia en forma secuencial, resulta natural

la aplicación de técnicas de computación de alta performance para mejorar los tiempos de ejecución. Janson y sus coautores resumen los principales aspectos que deben ser considerados para diseñar un algoritmo ACO multicolonia. Estos aspectos son:

- *Estructura de comunicación y topología:* típicamente se han utilizado las topologías completa, anillo, hipercubo y aleatoria.
- *Tipo de información intercambiada entre las colonias:* la información intercambiada puede ser la matriz de feromona, vectores de feromona (si la actualización en la feromona depende solamente de la calidad de la solución) o soluciones (migrantes, la mejor solución global a las colonias, la mejor solución en una vecindad, la mejor solución local a una colonia).
- *La forma de utilización de la información recibida de otras colonias:* las opciones generalmente utilizadas son sustituir una solución elite cuando la solución recibida es mejor, actualizar la matriz de feromona con la solución recibida o agregar a la generación actual la solución recibida.
- *Cadencia de la comunicación:* las variantes típicamente utilizadas han sido realizar la comunicación en cada iteración, con una frecuencia fija de iteraciones y dependiente de la calidad de las soluciones obtenidas.
- *Homogéneo vs. heterogéneo:* en la variante heterogénea las colonias pueden utilizar diferentes parámetros e inclusive diferentes variantes de ACO. Es posible utilizar un enfoque heterogéneo en cada iteración, donde las colonias tengan diferentes parámetros o un enfoque heterogéneo entre las iteraciones, donde las colonias usan las mismos parámetros que se modifican cada una cierta cantidad de iteraciones.

Existe una fuerte similitud entre el modelo multicolonia y las colonias de hormigas paralelas con interacción. El enfoque presentado por Janson y sus coautores es más completo que el de Randall y Lewis, ya que flexibiza la restricción de sincronizar las matrices de feromona, permitiendo agrupar una mayor cantidad de propuestas en la categoría.

4.4.3. Taxonomía de paralelismo aplicado a EA

Las implementaciones paralelas para EA han suscitado mayor interés por parte de la comunidad científica que las implementaciones paralelas para ACO. Su utilización está bastante extendida, lo que ha contribuido a una amplia comprensión y desarrollo de los modelos de paralelismo que es conveniente aplicar sobre EA.

Actualmente, las estrategias para aplicar paralelismo a EA están bien definidas y están resumidas en una taxonomía que es aceptada por la comunidad académica. La taxonomía establece las siguientes categorías [32, 109]:

- *Ejecuciones paralelas independientes:* este modelo consiste en la ejecución en paralelo del mismo algoritmo secuencial. Se caracteriza por tener más de una población pero sin ningún tipo de interacción entre ellas.
- *Modelo maestro-esclavo:* este modelo se caracteriza por trabajar sobre una única población. El maestro se encarga de la ejecución del EA y utiliza a los esclavos para

la evaluación de la función de fitness o para la aplicación de operadores evolutivos. El modelo de búsqueda no presenta diferencias con la de una ejecución serial, el paralelismo solamente se utiliza para mejorar el tiempo de ejecución.

- *Modelo distribuido o de islas:* este modelo se caracteriza por trabajar con una población dividida en varias subpoblaciones. Cada subpoblación trabaja en forma independiente y con cierta frecuencia comunica individuos (migrantes) a las otras subpoblaciones. Los principales parámetros que deben ser considerados en el intercambio de individuos son la tasa de migración, el intervalo entre migraciones, las políticas de selección y/o remplazo de migrantes y la topología de interconexión de las subpoblaciones.

- *Modelo celular:* este modelo se caracteriza por trabajar con una única población que se estructura en vecindades pequeñas. Cada individuo interactúa solamente con los individuos vecinos. Las vecindades tiene solapamientos para permitir que buenas soluciones se esparzan a toda la población.

- *Otros modelos o híbridos:* se caracterizan por poseer características de más de un modelo. Un ejemplo de esta categoría es utilizar dos modelos en forma jerárquica.

Existen muchos puntos de contacto entre los modelos identificados por esta taxonomía y los propuestos por Randall y Lewis. La principal diferencia que se puede apreciar es la ausencia en ACO de una categoría de características similares al modelo celular de EA. Esta ausencia puede explicarse porque la matriz de feromona, que mantiene la memoria de la búsqueda, debe ser utilizada por todas las hormigas para generar las soluciones. Utilizar un modelo celular implicaría tener una matriz de feromona por hormiga, construida a partir de las soluciones que generó y de las que generaron sus vecinos. Este enfoque implica cambiar sustancialmente la forma de implementar el ACO y su funcionamiento.

4.4.4. Ampliación de la taxonomía de paralelismo aplicado a ACO

La taxonomía de Randall y Lewis presenta fundamentalmente dos aspectos que pueden ser corregidos para permitir un mayor alineamiento con los modelos de paralelismo para EA: la ausencia de un modelo celular y las fuertes restricciones impuestas sobre el modelo de colonias de hormigas paralelas con interacción. A partir del relevamiento de implementaciones de ACO paralelos, se incorpora un modelo maestro-esclavo de grano grueso. A continuación, se presenta una propuesta de taxonomía ampliada que incorpora dichas correcciones.

La taxonomía de estrategias para implementaciones de ACO paralelos que se propone tiene las siguientes categorías:

- *Ejecuciones paralelas independientes:* se corresponde con la categoría colonias de hormigas paralelas independientes de Randall y Lewis.

- *Modelo maestro-esclavo:* en esta categoría es posible distinguir tres subcategorías. La primer subcategoría es un modelo maestro-esclavo de grano grueso, suele corresponder a una descomposición de dominio del problema. Los esclavos resuelven subproblemas en forma independiente, mientras que el maestro maneja la información global del problema, encargándose de construir una solución completa a

partir de las soluciones parciales. La segunda subcategoría es un modelo maestro-esclavo de grano fino, en el cual los esclavos se encargan de construir soluciones completas. Los esclavos no tienen porqué corresponderse con una única hormiga, sino que pueden agrupar a varias. El maestro se encarga del manejo de la matriz de feromona y cada uno de los esclavos comunica su(s) solución(es) al maestro. La subcategoría anteriormente descripta es más amplia que la categoría hormigas paralelas de Randall y Lewis. La tercera subcategoría tiene grano muy fino y se caracteriza porque los esclavos se encargan de evaluar la incorporación de una componente a una solución, correspondiéndose con la categoría evaluación en paralelo de componentes de soluciones de Randall y Lewis.

- *Modelo distribuido, de islas o multicolonia:* se corresponde con el modelo multicolonia descripto por Janson y sus coautores. Un caso particular que queda comprendido en este modelo es la categoría colonia de hormigas paralelas con interacción de Randall y Lewis. Las implementaciones comprendidas en este modelo son de grano grueso.

- *Modelo celular:* Se caracteriza por trabajar con una única población que se estructura en vecindades pequeñas. Cada hormiga interactúa solamente con las hormigas vecinas, teniendo cada vecindad su propia matriz de feromona. La actualización del rastro de feromona en cada matriz considera solamente las soluciones construida por las hormigas de la vecindad. Las vecindades tiene solapamientos para permitir que buenas soluciones se esparzan a toda la población. A partir del relevamiento de aplicaciones de paralelismo sobre ACO todavía no existen implementaciones de este modelo. Las implementaciones comprendidas en este modelo son de grano fino utilizando múltiples matrices de feromona, a diferencia del modelo maestro-esclavo de grano fino que utiliza una única matriz de feromona.

- *Otros modelos o híbridos:* en esta categoría se incluyen las implementaciones paralelas que utilizan más de un modelo. Un caso particular de híbrido es la categoría combinación paralela de hormigas y evaluación de componentes de soluciones de Randall y Lewis, ya que utiliza dos modelos maestro-esclavo en forma jerárquica.

4.5. Aplicaciones de paralelismo a ACO

En esta sección se presenta un resumen de artículos relevantes sobre la implementación de ACO paralelos. La estructura de la sección es la siguiente. En primer lugar se presentan algunos trabajos pioneros que tienen un indudable valor histórico pero escasa utilidad práctica. Seguidamente, se comentan artículos que implementan ACO paralelos siguiendo un modelo maestro-esclavo. Posteriormente, se reseñan los trabajos que siguen un modelo multicolonia. Finalmente, se presentan trabajos que evalúan implementaciones de más de un modelo de paralelismo.

4.5.1. Trabajos pioneros

Una de las primeras aproximaciones al problema de implementar un algoritmo ACO paralelo se atribuye a Bolondi y Bondaza [23] en el año 1993. La implementación propuesta según comentan Middendorf et al. [121] fue de grano muy fino y correspondía a colocar una hormiga en cada uno de los procesadores. En la descripción de la propuesta

no queda claro si se maneja una única matriz de feromona o una por hormiga. Tampoco es claro si cada hormiga comunica sus soluciones a todas las hormigas o si se definían vecindades. Middendorf et al. señalaron como una falencia de la propuesta que al aumentar la cantidad de procesadores no escalaba. A partir de esa falencia es razonable asumir que se trabajaba con única matriz de feromona y que las hormigas usaban una topología completa para comunicarse las soluciones.

En 1998, Bullnheimer et al. [29] estudiaron el efecto de las comunicaciones para una implementación paralela de AS mediante el modelo maestro-esclavo. Se formularon dos propuestas de paralelización del AS. La primera propuesta fue una implementación paralela sincrónica en la cual cada esclavo genera una solución, siendo el maestro el encargado de calcular la matriz de feromona en forma global y distribuirla a los esclavos. Esa propuesta presenta como dificultad que es necesario sincronizar todos los esclavos para pasar a la siguiente iteración. La segunda propuesta consistía en una implementación paralela parcialmente asincrónica, donde cada esclavo ejecuta durante una cierta cantidad de iteraciones realizando actualizaciones locales de la matriz de feromona en forma independiente. Con una cierta frecuencia de iteraciones, se realiza una sincronización global de los rastros. La segunda de las propuestas podría ser asimilable a un modelo multicolonia en la cual cada esclavo corresponde a una colonia. El interés de los autores fue evaluar el efecto de las comunicaciones por lo cual solamente realizan una simulación utilizando N-MAP [29] sobre instancias de tamaño hasta 500. Las pruebas no involucraron la implementación de ninguna variante concreta, ni ningún problema específico, lo que representa un serio defecto del trabajo. Las métricas utilizadas para la evaluación fueron speedup, eficiencia y eficacia, comprobándose que la implementación parcialmente asincrónica tiene un mejor desempeño que la implementación sincrónica. De todas formas, la implementación sincrónica en instancias grandes logra speedups casi lineales.

Posteriormente, Stützle [148] realizó un estudio sobre la ejecución en forma paralela e independiente de $\mathcal{MM}$AS con búsqueda local para el TSP. Su objetivo fue evaluar si es conveniente ejecutar en paralelo o realizar una única ejecución de mayor tiempo, estudiando solamente el efecto del paralelismo en la calidad de las soluciones obtenidas. La evaluación se realizó sobre instancias de la biblioteca TSPLIB [143] de hasta 1173 ciudades. Para realizar una evaluación justa, a las ejecuciones en paralelo se les otorga un tiempo máximo de procesamiento igual a la k-ésima parte del tiempo que a la ejecución serial (siendo k la cantidad de procesadores). Los resultados fueron favorables a las ejecuciones paralelas obteniendo en todos los casos soluciones de mayor calidad que el algoritmo secuencial.

Los trabajos pioneros tienen valor histórico por tratarse de los primeros intentos de aplicar paralelismo sobre ACO. Sin embargo, son muy limitados en sus alcances, por lo que no permiten extraer conclusiones definitivas. Solamente es posible señalar que el modelo maestro-esclavo potencialmente permite obtener speedups casi lineales y que el modelo de ejecuciones paralelas independientes puede permitir mejorar la calidad de las soluciones obtenidas.

4.5.2. Modelo maestro-esclavo

A continuación se presenta un resumen de artículos relevantes en los cuales se implementaron ACO paralelos siguiendo un modelo maestro-esclavo:

- **Parallel Ant Colonies for Combinatorial Optimization Problems (1999)** y **Parallel Ant Colonies for the Quadratic Assignment Problem (2001)**

  Talbi et al. [156, 157] presentaron una implementación paralela de ANTabu para el QAP. ANTabu es un híbrido que incorpora a ACO un mecanismo de búsqueda local basado en TS. La implementación paralela consiste en un maestro-esclavo de grano fino sincrónico. El maestro maneja la información global de la colonia que consiste en la matriz de feromona y la mejor solución encontrada hasta el momento. Los esclavos cumplen con las tareas propias del proceso de búsqueda de cada hormiga, construyendo las soluciones y aplicando la búsqueda local basada en TS. Los esclavos envían las soluciones que generan al maestro y este envía la matriz de feromona a los esclavos. Los autores solamente evaluaron la calidad de las soluciones obtenidas por su propuesta comparando contra las obtenidas por otra variante de hormigas (HAS-QAP), un TS paralelo, variantes de GA y de VNS. Se utilizaron instancias de la biblioteca QAPLIB [30] de hasta 256 locaciones. La evaluación fue realizada sobre un cluster de 10 PCs utilizando un tiempo fijado previamente, obteniendo buenos resultados que aventajan a la mayoría de las otras variantes.

- **A Parallel Implementation of Ant Colony Optimization (2002)**

  Randall y Lewis [140] abordaron el TSP mediante una implementación paralela de ACS siguiendo un modelo maestro-esclavo de grano fino. La propuesta presenta como particularidad que la actualización local del rastro de feromona la realiza el proceso maestro. Cada esclavo envía al maestro el componente que incluyó al construir su solución. Posteriormente, el maestro envía al esclavo la actualización local del rastro de feromona correspondiente. La evaluación se realizó sobre instancias de la biblioteca TSPLIB de hasta 657 ciudades, considerando solamente el speedup y la eficiencia y omitiendo la calidad de las soluciones obtenidas. La plataforma utilizada fue un cluster utilizando entre 2 y 8 procesadores. El speedup está muy por debajo de ser lineal y la eficiencia máxima (0.83) se obtiene utilizando 2 procesadores, siendo las instancias más grandes las que presentan los mejores valores del speedup. Los pobres resultados obtenidos muestran que el enfoque seguido para la actualización local no es un esquema eficiente para el paralelismo.

- **Parallel Framework for Ant-like Algorithms (2004)** y **Multi-Level Parallel Framework (2005)**

  Craus y Rudeanu [43] implementaron un framework reutilizable para la ejecución de algoritmos secuenciales en ambientes paralelos siguiendo un modelo maestro-esclavo. Los objetivos de diseño del framework fueron obtener un buen nivel de abstracción y optimizar la comunicación entre los procesos. El framework utiliza checkpoints para la comunicación entre el maestro y los esclavos. Los esclavos solicitan intercambios de datos con el maestro por turnos en forma asíncrona. Solamente se intercambia entre el maestro y el esclavo la información que se ha modificado desde el último checkpoint de ese esclavo. El framework fue evaluado utilizando un ACO para resolver el TSP sobre una instancia de 229 ciudades de la la biblioteca TSPLIB. La plataforma utilizada fue un Sun Fire 15K HPC con 48 procesadores. La evaluación solamente consideraba el speedup, siendo este casi lineal cuando se utilizan hasta 25 procesadores y degradándose cuando se utilizan más procesadores.

Craus y Rudeanu [44] complementaron su trabajo previo implementando un framework piramidal siguiendo un modelo maestro-esclavo con la inclusión de submaestros, que actúan como concentradores de los esclavos que están bajo su órbita. El funcionamiento siguió las mismas pautas que su trabajo previo, reduciendo la comunicación al maestro por la incorporación de los submaestros. Los autores comprobaron que era posible mantener el comportamiento casi lineal del speedup con más de 25 procesadores.

- **Parallel Ant Systems for the Capacitated Vehicle Routing Problem (2004)**

  Doerner et al. [58] abordaron el Vehicle Routing Problem (VRP) mediante una implementación paralela de AS_{rank} siguiendo un modelo maestro-esclavo que permite asignar a cada procesador más de una hormiga. El objetivo de la propuesta consistió en lograr reducir el tiempo de ejecución del algoritmo. La evaluación de la propuesta fue realizada sobre instancias clásicas del VRP [34] de hasta 199 clientes. Se evaluó el speedup y la eficiencia del ACO paralelo sobre un cluster Beowulf, considerando 2, 4, 8, 16 y 32 procesadores. Los resultados obtenidos indicaron que el speedup fue sublineal y la eficiencia decreció al aumentar la cantidad de procesadores. Los autores señalaron que un buen compromiso entre el desempeño y la cantidad de procesadores para el tamaño de problema resuelto se da utilizando 8 procesadores (speedup 6 y eficiencia superior a 0.7).

- **Comparing Parallelization of an ACO: Message Passing vs. Shared Memory (2005)** y **A Shared Memory Parallel Implementation of Ant Colony Optimization (2005)**

  Delisle et al. [49] utilizaron un enfoque de paralelismo similar al de Randall y Lewis [140] para la variante ACS aplicado al TSP, siendo uno de los escasos trabajos que abordan la implementación de un ACO paralelo en una arquitectura con memoria compartida. La implementación utiliza dos regiones críticas para que sólo una hormiga a la vez actualice la mejor solución encontrada y el rastro de feromona localmente. La actualización global del rastro de feromona requiere sincronizar los procesos concurrentes. La evaluación fue realizada utilizando 16 procesadores Power 3 en un nodo NH2 de un IBM/P 1600 sobre instancias de la biblioteca TSPLIB de hasta 657 ciudades. La evaluación experimental comparó el speedup y la eficiencia con los valores obtenidos por Randall y Lewis. Para entre 2 y 8 procesadores, se obtuvo un mejor desempeño que Randall y Lewis manteniendo la calidad de las soluciones obtenidas. Sin embargo, algunos aspectos sugieren que la comparación no fue justa, ya que utilizan diferentes definiciones de speedup, la implementación de Delisle et al. solamente realiza sincronizaciones de las actualizaciones locales de los rastros de feromona con una cierta frecuencia y no en todas las iteraciones como Randall y Lewis, y la plataforma de ejecución utilizada es muy diferente. Los autores evaluaron el efecto que tiene aumentar la cantidad de hormigas por procesador reduciendo la cantidad de iteraciones de forma de mantener fija la cantidad de soluciones construidas sobre el speedup y la eficiencia, obteniendo mejoras considerables a costa de disminuir la calidad de las soluciones obtenidas. Los autores también propusieron una implementación paralela mediante un modelo híbrido en dos niveles. El primer nivel es una multicolonia en la que las colonias se comuniquen entre sí mediante pasaje de mensajes y en el segundo nivel la comunicación

dentro de las colonias se realiza mediante la propuesta de los autores para memoria compartida. Esa propuesta no ha sido llevada a la práctica.

Posteriormente, Delisle et al. [50] complementaron su trabajo, estudiando el efecto de utilizar distintas plataformas de prueba. La evaluación fue realizada sobre dos equipos diferentes, un IBM/P 1600 y un SGI Origin 3800. Se evaluó el speedup y la eficiencia para 2, 4, 8 y 16 procesadores, reportando que el equipo IBM presenta sistemáticamente un mejor desempeño. Los autores también evaluaron el efecto de modificar el tipo de nodos del equipo IBM, pasando de un NH2 Power 3 a un Regatta Power 4, obteniendo mejoras significativas en el desempeño. La experiencia realizada muestra como la evolución tecnológica puede provocar mejoras en el desempeño de los algoritmos paralelos, dejando claro que pueden existir limitaciones en el desempeño que no sean atribuibles al algoritmo paralelo.

- **A Parallel Version of the D-Ant Algorithm for the Vehicle Routing Problem (2005)**

  Doerner et al. [59] aplicó al VRP una variante específica de ACO conocida como D-Ant, que se basa en la descomposición del problema en subproblemas de menor dimensión. El algoritmo planteado tiene cuatro etapas: generar una solución inicial usando Savings based ACO (SACO); determinar centros de gravedad para cada ruta de la solución; obtener clusters a partir de los centros de gravedad y los consumidores; y resolver los subproblemas de cada cluster mediante colonias SACO. Cada colonia trabaja en forma independiente y posteriormente se unen las soluciones de los subproblemas. La matriz de feromona se maneja en forma global, y cada colonia utiliza una submatriz. Un proceso maestro se encarga de realizar la evaporación y el refuerzo de la mejor solución obtenida hasta el momento. El enfoque corresponde al modelo maestro-esclavo de grano grueso, en el que cada esclavo trabaja sobre un distinto subproblema. La evaluación experimental se realizó utilizando instancias clásicas del VRP de entre 150 y 199 clientes. La plataforma de evaluación fue un cluster de 16 procesadores. Se comprobó que es posible mantener la calidad de las soluciones reduciendo el tiempo de ejecución. Se obtuvo el mejor valor de eficiencia (0.6) trabajando con dos procesadores y el mejor valor de speedup (3.85) trabajando con ocho procesadores.

- **A Grid Ant Colony Algorithm for the Orienteering Problem (2005)**

  La única referencia de implementación de un ACO paralelo en un entorno grid corresponde a Mocholí et al. [123]. Los autores descartaron que las hormigas o las colonias trabajaran sobre todo el espacio de búsqueda debido a que el problema considerado (Orienteering Problem) suele aplicarse sobre instancias de miles de nodos. El enfoque planteado se caracteriza por separar en clusters de acuerdo a la distancia euclidiana entre los nodos, obtener soluciones parciales sobre los grafos clusterizados utilizando colonias de hormigas independientes y unir las soluciones parciales para obtener una solución completa al problema. Al unir las soluciones parciales es posible que no se obtenga una solución factible, realizándose búsquedas locales hasta factibilizar la solución. El enfoque planteado utiliza un modelo maestro-esclavo en el que los esclavos trabajan sobre distintos subproblemas. Los servicios de alto nivel del grid son provistos por web services. La evaluación experimental se realizó sobre instancias propias, generadas aleatoriamente de hasta 10.000 nodos. La plataforma de evaluación fue un cluster de 32 PCs. El

estudio mostró que el tiempo de ejecución del algoritmo propuesto decrece exponencialmente al aumentar la cantidad de colonias. Los resultados indicaron que al aumentar la cantidad de colonias se mejora la calidad de las soluciones obtenidas.

- **A Study of Distributed Parallel Processing for Queen Ant Strategy in Ant Colony Optimization (2005)**

  Iimura et al. [98] presentaron la paralelización de una variante poco difundida de ACO llamada Queen Ant Strategy (AS_{queen}). AS_{queen} se caracteriza por tener una reina y un grupo de agentes que generan soluciones. La reina da directivas a los agentes para diversificar o explotar la búsqueda, a partir de la calidad de las soluciones generadas por los agentes. AS_{queen} funciona como un conjunto de colonias con un control centralizado. El modelo de paralelismo planteado corresponde a un maestro-esclavo de grano grueso, aunque en este caso no se realiza una descomposición de dominio. Los esclavos trabajan sobre el mismo problema en forma independiente enviando periódicamente su mejor solución al maestro. El maestro es quien dirige la búsqueda, enviando las directivas sobre la forma de proseguir la búsqueda a los esclavos. La evaluación experimental fue realizada utilizando un cluster heterogéneo de hasta 9 PCs sobre una instancia de la biblioteca TSPLIB de 76 ciudades. Al aumentar la cantidad de grupos de agentes se aprecian mejoras leves en el costo promedio de las soluciones obtenidas y en la cantidad de veces que se obtiene el óptimo, reduciendo el tiempo de ejecución.

- **Cunning Ant System for Quadratic Assignement Problem with Local Search and Parallelization (2007)**

  Tsutsui y Liu [161] presentaron un algoritmo paralelo de *c*AS aplicado al QAP, diseñado con el objetivo de reducir el tiempo de ejecución del algoritmo serial que incorpora una búsqueda local con elevado tiempo de ejecución (99 % del tiempo total de ejecución del algoritmo). Se propuso un modelo de paralelismo novedoso para los algoritmos ACO, que corresponde a un maestro-esclavo en el que los esclavos solamente se ocupan de ejecutar las búsquedas locales. El modelo de paralelismo es un caso particular de la subcategoría en la cual los esclavos se encargan de construir las soluciones completas, aunque en este caso sólo realizan mejoras sobre las soluciones construidas por el maestro. La evaluación experimental fue realizada sobre instancias de la biblioteca QAPLIB de hasta 150 locaciones. La plataforma de evaluación utilizada fue dos PCs con cuatro núcleos cada una. Solamente se evaluó el impacto del paralelismo en el tiempo de ejecución, comprobando que si el overhead introducido por las comunicaciones es bajo, es posible lograr mejoras significativas en el tiempo de ejecución.

La tabla 4.1 presenta un resumen de las aplicaciones de paralelismo que siguen un modelo maestro-esclavo sobre ACO relevadas. En las columnas se indica: autores principales (Autores), año de publicación (Año), variante de ACO utilizada (Variante), problema abordado (Problema), estrategia de paralelismo utilizada (Modelo) y plataforma utilizada (Plataforma). La categoría ACO corresponde a una variante específica para el problema abordado mientras que la categoría ND indica que la información no está disponible. M-E, MultiP y MultiC abrevian Maestro-esclavo, Multiprocesador y Multicomputadora respectivamente.

Tabla 4.1: Propuestas de ACO paralelos modelo maestro-esclavo

Autores	Año	Variante	Problema	Modelo	Plataforma
Talbi et al.	1999, 2001	ANTabu	QAP	M-E grano fino	COW
Randall y Lewis	2002	ACS	TSP	M-E grano fino	COW
Craus y Rudeanu	2004, 2005	ND	TSP	M-E grano fino y grano fino con submaestros	MultiC
Doerner et al.	2004	AS_{rank}	VRP	M-E grano fino	COW
Delisle et al.	2005, 2005	ACS	TSP	M-E grano fino	MultiP
Doerner et al.	2005	ACO	VRP	M-E grano grueso (resuelven distintos subproblemas)	COW
Mocholí et al.	2005	ACO	OP	M-E grano grueso (resuelven distintos subproblemas)	Grid
Iimura et al.	2005	AS_{queen}	TSP	M-E grano grueso (multicolonia con control centralizado)	COW
Tsutsui y Liu	2007	cAS	QAP	M-E grano fino	Híbrido

El modelo de maestro-esclavo ha sido ampliamente utilizado en la implementación de ACOs paralelos, mostrando que es posible obtener reducciones en el tiempo de ejecución y un buen desempeño computacional del algoritmo. El modelo maestro-esclavo de grano fino es el que generalmente ha sido implementado. La ausencia de implementaciones del modelo maestro-esclavo de grano muy fino a los seis años de propuesta la taxonomía de Randall y Lewis pone en tela de juicio si realmente debería ser considerada como una subcategoría. Por otro lado, se han relevado tres trabajos donde se implementa un maestro-esclavo grano grueso que no estaba presente en la taxonomía de Randall y Lewis. Gran parte de los trabajos relevados fueron implementados sobre clusters aunque existen implementaciones para las plataformas más comunes. El problema más usado para la evaluación experimental es el TSP, seguido por el QAP y el VRP.

4.5.3. Modelo multicolonia

El modelo multicolonia fue utilizado originalmente por Michel y Middendorf [119] para resolver el problema Shortest Common Supersequence, aunque los autores no establecen haber realizado una implementación paralela. En el algoritmo propuesto, las colonias intercambian las mejores soluciones con una frecuencia fija. Cuando se realiza la actualización de la matriz de feromona, cada colonia incorpora la mejor solución recibida hasta el momento siguiendo una estrategia elitista. El estudio de la calidad de las soluciones obtenidas mostró que al utilizar una multicolonia se obtienen soluciones

levemente superiores que al utilizar una única colonia.

Existen otros dos trabajos que exploraron ideas novedosas para el modelo multicolonia, aunque sin proponerse una implementación paralela.

El primero de los trabajos corresponde a Kawamura et al. [103]. En su propuesta cada colonia tiene su propia matriz de rastro de feromona y recibe efectos positivos o negativos (aumentando o reduciendo la probabilidad de elegir una componente) de las otras colonias. Cada colonia debe conocer las matrices de feromona de las otras colonias. Una posible implementación paralela de la propuesta de Kawamura et al. debería considerar ese hecho, siendo una alternativa realizarla en una plataforma con memoria compartida para evitar el envío de las matrices.

El segundo de los trabajos corresponde a Hara et al. [94]. En su propuesta la cantidad de colonias varía en forma dinámica durante la ejecución del algoritmo. La propuesta se caracteriza por utilizar mecanismos de fisión para dividir colonias, y de extinción para eliminarlas. El mecanismo de fisión produce dos colonias a partir de una, quedando las colonias acopladas hasta que se encuentre una mejor solución. El mecanismo de extinción elimina la colonia con la solución de peor calidad cuando dos colonias han alcanzado similares mejores soluciones hasta el momento. Una posible implementación paralela de esta propuesta podría presentar alguna dificultad por la inclusión de una etapa de búsqueda dependiente en la cual dos colonias actúan fuertemente acopladas.

A continuación se presenta un resumen de artículos relevantes en los cuales se implementaron ACO paralelos siguiendo un modelo multicolonia:

- **Information Exchange in Multi Colony Ant Algorithms (2000)** y **Multi Colony Ant Algorithms (2002)**

  Middendorf et al. [120] propusieron un modelo multicolonia para la variante AS con elitismo aplicado al TSP. Cada colonia utiliza su propia matriz de feromona. El intercambio de información entre las colonias se produce cada un número fijo de iteraciones. En evaluaciones preliminares ([106]) se pudo comprobar que es mejor intercambiar las mejores soluciones que las matrices de feromona completas, por tanto se planteron cuatro propuestas para el intercambio de información entre las colonias: enviar la mejor solución global a todas las colonias; enviar las mejores soluciones locales utilizando una topología de interconexión de anillo unidireccional, modificando solamente la mejor solución de la colonia destino en caso de ser necesario; enviar migrantes utilizando una topología de interconexión de anillo unidireccional, actualizando la matriz de feromona con las mejores soluciones, ya sean generadas por la colonia o recibidas como migrantes; y la aplicación a la vez de las dos propuestas anteriores. La evaluación experimental fue realizada sobre una instancia de la biblioteca TSPLIB de 101 ciudades. Solamente se estudió la calidad de las soluciones obtenidas comparando con un modelo multicolonia sin intercambios. Los mejores resultados se obtuvieron utilizando solamente las mejores soluciones locales. Los autores analizaron la frecuencia de intercambio, afirmando que debe existir un equilibrio para no degradar la calidad de las soluciones obtenidas.

  Posteriormente, Middendorf et al. [121] extendieron su trabajo anterior incorporando la evaluación del modelo multicolonia aplicado al QAP. La propuesta es básicamente la misma que en el trabajo anterior, pero evaluando solamente el intercambio de las mejores soluciones locales y el modelo sin intercambios. La evaluación experimental fue realizada sobre una instancia de la biblioteca QAPLIB de

60 locaciones. Los resultados obtenidos son levemente inferiores a los del trabajo previo. Se evaluó la conveniencia de ejecutar una colonia contra una multicolonia construyendo la misma cantidad de soluciones sobre el TSP, constatándose que el enfoque multicolonia es superior.

- **Colonia de hormigas en un ambiente paralelo asíncrono (2002)**

  Barán y Almirón [13] presentaron una implementación paralela multicolonia de ACS aplicada al TSP. La implementación fue asincrónica y utilizaba una topología de conexión completa. La evaluación experimental fue realizada sobre una instancia de la biblioteca TSPLIB de 100 ciudades. La plataforma utilizada fue un cluster de PCs homogéneo de cuatro computadoras. La calidad de los resultados obtenidos al paralelizar es levemente superior a la del algoritmo serial. Se reportan muy buenos resultados de desempeño computacional, obteniendo un speedup superior a 3.9 y una eficiencia del 98 % al ejecutar en cuatro PCs. Los resultados reportados son muy buenos pero no puede realizarse una evaluación definitiva de la propuesta, ya que las pruebas fueron realizadas sobre solamente cuatro computadoras.

- **A New Approach to Exploiting Parallelism in Ant Colony Optimization (2002)**

  Piriyakumar y Levi [137] abordaron el TSP mediante una variante paralela que incorpora características de $\mathcal{MM}$AS. La implementación paralela propuesta es sincrónica y utiliza un modelo multicolonia. Cada colonia realiza la actualización local del rastro de feromona y la actualización global se lleva a cabo en forma conjunta cada una cantidad fija de iteraciones. Periódicamente las colonias comunican su mejor solución al resto para que sea utilizada en la actualización global. La evaluación experimental se realizó en un equipo Cray T3E, utilizando una instancia de la biblioteca TSPLIB de 52 ciudades. El enfoque seguido por los autores para la evaluación experimental es poco ortodoxo. Los indicadores utilizados fueron el costo de la solución, el tiempo total de cómputo, el tiempo por procesador y la relación entre el tiempo de comunicación y el tiempo ocioso con respecto al tiempo total de cómputo. El estudio mostró que es posible obtener resultados de buena calidad manteniendo valores acotados de tiempo ocioso de los procesadores y tiempo de comunicación.

- **Parallel Ant Colony Systems (2003)** y **Ant Colony System with Communication Strategies (2004)**

  Chu et al. [37] propusieron una implementación paralela de la variante ACS para el TSP con el objetivo de mejorar la calidad de las soluciones obtenidas. El modelo de paralelismo utilizado fue multicolonia. Cada colonia ejecuta un ACS y con una frecuencia fija de iteraciones comunica al resto su mejor solución hasta el momento. Cada colonia realiza su actualización de feromona independientemente y en las iteraciones en las cuales recibe soluciones de otras colonias incorpora una actualización adicional. Se propusieron tres formas diferentes de utilizar las soluciones intercambiadas: incrementar la feromona de las componentes pertenecientes a la mejor solución de todos las colonias (PACS1); incrementar la feromona de las componentes pertenecientes a la mejor solución del vecindario (PACS2); y realizar los incrementos de feromona correspondientes a PACS1 y PACS2 (PACS3). La evaluación experimental solamente consideró la calidad de las soluciones obtenidas

sobre tres instancias de la biblioteca TSPLIB de hasta 225 ciudades. La evaluación mostró que las tres variantes de PACS obtienen mejoras de más de un 4 % y 9 % con respecto a ACS y AS.

Posteriormente, Chu et al. [36] extendieron su trabajo previo incorporando la evaluación de distintas formas de utilizar las soluciones intercambiadas. Las estrategias evaluadas fueron: incrementar la feromona de las componentes pertenecientes a la mejor solución de todos las colonias, incrementar la feromona de las componentes pertenecientes a la mejor solución del vecindario, realizar el incremento adicional de acuerdo a un anillo unidireccional, realizar el incremento adicional de acuerdo a un anillo bidireccional y realizar el incremento adicional de acuerdo a más de una estrategia. La evaluación experimental tuvo las mismas características que en su trabajo previo. La evaluación mostró que las variantes de PACS propuestas superan ampliamente a AS y a ACS.

- **Adaptive Parallel Ant Colony Algorithm (2005)**

  Chen y Zhang [33] propusieron una implementación paralela multicolonia para el TSP. La propuesta se caracteriza por colocar una colonia en cada procesador y por intercambiar las mejores soluciones entre pares de colonias en forma adaptativa, de acuerdo a la bondad de la soluciones obtenidas por cada colonia. Se presenta una formulación adaptativa para ajustar la frecuencia entre los intercambios de información a partir de la diversidad de las soluciones generadas. La evaluación experimental fue realizada en una computadora masivamente paralela Dawn 2000 sobre instancias de la biblioteca TSPLIB de hasta 318 ciudades. Se evaluó la calidad de las soluciones obtenidas y el tiempo de ejecución para un ACO, una multicolonia con intercambio circular de la mejor solución local y la variante paralela propuesta, siendo superior en calidad y en tiempo la propuesta adaptativa. Los autores también evalúan el mecanismo adaptativo de ajuste de la frecuencia de intercambios de información entre las colonias. Se constató que se obtienen soluciones de mejor calidad que usando intervalos fijos de intercambio, aunque a costa de un mayor tiempo de ejecución. En el estudio se evaluó el speedup del algoritmo propuesto sobre 5, 10, 15, 20, 25, 30 y 35 procesadores. Se constató que el speedup fue sublineal, mejorando al aumentar el tamaño de las instancias del problema consideradas.

- **A Parallel Ant Colony Algorithm for Bus Network Optimization (2007)**

  Yang et al. [172] abordaron el problema Urban Bus Network Design (UBND) que consiste en maximizar la densidad de viajes directos en una red de ómnibus, considerando la demanda. El ACO propuesto se caracteriza por incorporar un término local en la formulación de la cantidad de feromona a depositar por una hormiga. Se utiliza un modelo multicolonia en la que cada colonia resuelve el mismo problema independientemente usando su propia matriz de feromona. Con una frecuencia fija de iteraciones se produce la migración de la mejor solución de cada colonia, considerando una topología de interconexión de anillo unidireccional. Los autores realizaron la evaluación de su propuesta sobre un cluster de 8 PCs, estudiando la calidad de las soluciones obtenidas y el tiempo de ejecución. Se constató la obtención de mejoras en ambos aspectos en comparación con la utilización de una sola colonia. No se realizaron estudios de speedup ni de eficiencia.

- **Exchange Strategies for Multiple Ant Colony System (2007)**

  Ellabib et al. [71] plantearon un modelo multicolonia de ACS (MACS) que utiliza un mecanismo adaptativo para la actualización global de feromona que asigna pesos dependientes de la calidad de las soluciones obtenidas por una colonia con respecto al resto. El modelo también incorpora un modulo de intercambio que encapsula la comunicación entre las colonias. Cada colonia realiza su búsqueda en forma independiente y con una frecuencia fija se comunica con las otras colonias a través del módulo de intercambio. Las implementaciones de MACS fueron sincrónicas y se consideraron tres topologías de interconexión diferentes (estrella, hipercubo y anillo unidireccional). El algoritmo propuesto fue evaluado sobre un cluster de 8 PCs, realizando un estudio de la calidad de las soluciones obtenidas en dos etapas. En la primera etapa, se estudia el efecto de utilizar las distintas topologías de interconexión sobre los resultados para el VRPTW (sobre instancias estándar propuestas por Solomon [147]) y el TSP (instancias de la biblioteca TSPLIB de hasta 225 ciudades). Se constató que la topología de estrella obtiene los mejores resultados. En una segunda etapa se compara la calidad de las soluciones obtenidas mediante MACS con topología de estrella y otras variantes de ACO sobre el VRPTW y el TSP. Para el VRP, MACS superó en calidad de resultados a ACS, mientras que para el TSP, los resultados de MACS superan a los obtenidos por PACS. No se realizaron estudios de speedup ni de eficiencia.

La tabla 4.2 presenta un resumen de las aplicaciones de paralelismo siguiendo un modelo multicolonia sobre ACO relevadas. En las columnas se indica: autores principales (Autores), año de publicación (Año), variante de ACO utilizada (Variante), problema abordado (Problema), estrategia de paralelismo utilizada (Modelo) y plataforma utilizada (Plataforma). La categoría ACO corresponde a una variante específica para el problema abordado mientras que la categoría ND indica que la información no está disponible. Multicolonia se abrevia como MC.

El modelo de multicolonia ha sido utilizado ampliamente en la implementación de ACOs paralelos, mostrando que es posible obtener mejoras en la calidad de las soluciones obtenidas en comparación con los algoritmos secuenciales. También es posible reducir el tiempo de ejecución del algoritmo, llegando algunas implementaciones a alcanzar un speedup casi lineal. La mayor parte de los trabajos relevados fueron implementados sobre clusters. El problema más usado para la evaluación experimental es el TSP.

4.5.4. Más de un modelo

A continuación se presenta un resumen de artículos relevantes en los cuales se implementaron más de un modelo de ACO paralelos:

- **Parallel Ant System for the Set Covering Problem (2002)**

  Rahoual et al. [139] abordaron el Set Covering Problem (SCP) con una variante de ACO específicamente diseñada para el problema. Se incorporó un operador de búsqueda local, debido a que se obtuvieron pobres resultados al utilizar un algoritmo ACO puro. Los resultados obtenidos al utilizar el operador de búsqueda local son sustancialmente mejores que los resultados del ACO puro sin embargo, el tiempo de ejecución del algoritmo se duplica. Con el objetivo de reducir el tiempo de ejecución, se plantearon dos implementaciones paralelas, una mediante

Tabla 4.2: Propuestas de ACO paralelos modelo multicolonia

Autores	Año	Variante	Problema	Modelo	Plataforma
Middendorf et al.	2000, 2002	AS	TSP	MC	ND
Barán y Almirón	2002	ACS	TSP	MC	COW
Piriyakumar y Levi	2002	$\mathcal{MM}$AS	TSP	MC	MPP
Chu et al.	2003, 2004	ACS	TSP	MC	ND
Chen y Zhang	2005	ND	TSP	MC	MPP
Yang et al.	2007	ACO	UBND	MC	COW
Ellabib et al.	2007	ACS	VRPTW y TSP	MC	COW

ejecuciones independientes y otra que corresponde a un maestro-esclavo de grano fino. La evaluación experimental fue realizada en un cluster de 40 PCs. La implementación mediante ejecuciones independientes logró speedup casi lineal y eficiencias cercanas a 1, debido a que la comunicación entre procesos es escasa. En el caso de la implementación maestro-esclavo, la eficiencia depende fuertemente del tamaño de las instancias consideradas. En pruebas realizadas sobre 20 PCs, el speedup promedio fue 10.95 y la eficiencia varió entre 0.21 y 0.83. No se realizó un estudio sobre la calidad de las soluciones obtenidas por los ACO paralelos.

- **A Parallel Ant Colony Optimization Algorithm for All-Pair Routing in MANETs (2003)**

  Islam et al. [99] abordaron el problema de determinar el mejor camino entre todos los pares de nodos de una red *MANET* (Mobile Ad-hoc Network) utilizando un ACO paralelo. En cada procesador se coloca una hormiga y una parte de la información disponible de toda la red. Cada hormiga determina las mejores rutas para los pares de nodos del subgrafo correspondiente a su procesador. Para completar las soluciones, las hormigas deben interactuar para obtener las rutas entre los nodos que están en procesadores distintos. Los autores no aclararon cómo se realiza la comunicación entre las hormigas para construir las soluciones completas, ni qué estrategia se sigue para la paralelización del ACO. La evaluación experimental fue realizada sobre un cluster de 10 PCs, analizando el tiempo de ejecución y el speedup. Se verificó que el tiempo de ejecución disminuye al aumentar la cantidad de procesadores utilizados. Se constató que más del 90 % del tiempo de ejecución se utiliza en comunicación; esto es causado por las rutas que deben ser determinados entre nodos que fueron asignados a distintos procesadores.

- **Analyzing the Behavior of Parallel Ant Colony Systems for Large Instances of the Task Scheduling Problem (2005)** y **Two models of parallel ACO algorithms for the minimum tardy task problem(2007)**

  Alba et al. [9] propusieron la aplicación de paralelismo sobre ACS para el Minimum Tardy Task Problem (MTTP). Se implementaron tres modelos de paralelismo: ejecuciones paralelas independientes, modelo de islas con topología de interconexión de anillo y un modelo maestro-esclavo de grano fino. En el modelo de islas, con una frecuencia fija de iteraciones se comunica el mejor individuo a la

colonia determinado por la topología de interconexión. Para actualizar el rastro de feromona cada colonia considera las soluciones generadas en la iteración y el migrante recibido. Para realizar la evaluación de las propuestas se consideraron las siguientes implementaciones: secuencial, ejecuciones paralelas independientes, el modelo de islas en forma sincrónica y asincrónica y el maestro-esclavo en forma sincrónica y asincrónica. La evaluación experimental se realizó sobre un cluster de 3 PCs, utilizando instancias generadas por los autores. Los resultados obtenidos permiten concluir que la calidad de las soluciones obtenidas y la iteración en la que obtiene la mejor solución son similares para todas las propuestas. Los tiempos de ejecución de las variantes asincrónicas son mejores que los de las variantes sincrónicas. Asimismo, las ejecuciones paralelas independientes requieren menor tiempo que el modelo de islas, al no existir comunicación. Los autores también evaluaron el efecto en el speedup y la eficiencia de las implementaciones paralelas al incrementar el tamaño de los problemas para una cantidad fija de procesadores. Los resultados obtenidos son irregulares y no permiten extraer conclusiones terminantes, registrando speedups superiores a 1 y eficiencias superiores a 0.3 para 3 procesadores. Se señala que el problema presenta algunas particularidades, como el tamaño variable de las soluciones que pueden influir en los resultados obtenidos.

Posteriormente, Alba et al. [8] ampliaron el estudio sobre los modelos de ejecuciones paralelas independientes y de islas asincrónico con topologías de interconexión de anillo y estrella bidireccional. La evaluación experimental de los modelos de paralelismo se realizó sobre un cluster de 8 PCs. La calidad de los resultados obtenidos es mejor para las variantes que involucran comunicación, utilizando un criterio de parada de esfuerzo prefijado. La eficiencia es casi lineal, obteniendo con 8 procesadores 0.98 para ejecuciones independientes, 0.91 para el modelo de islas con topología de anillo y 0.69 para el modelo de islas con topología de estrella.

- **Parallel Ant Colony Optimization for 3D Protein Structure Prediction using HP Lattice Model (2005)**

  Chu et al. [35] presentaron una aplicación de ACO sobre un problema de predicción de la estructura de proteínas. Se plantearon tres implementaciones paralelas: un maestro-esclavo de grano fino, una multicolonia con intercambio circular de migrantes y una multicolonia en la que las colonias comparten la matriz de feromona. La evaluación experimental fue realizada sobre un cluster de PCs, utilizando 3, 4 y 5 procesadores. La evaluación de los resultados fue poco ortodoxa, utilizándose la métrica cantidad de tics de CPU necesarios para encontrar el óptimo. Los resultados indicaron que el mejor desempeño lo obtiene la multicolonia con intercambio de migrantes, seguida de la multicolonia con matriz de feromona compartida.

- **Communication Strategies for Parallel Cooperative Ant Colony Optimization on Clusters and Grids (2005)** y **Parallel Cooperative Savings Based Ant Colony Optimization - Multiple Search and Decomposition Approaches (2006)**

  Doerner et al. [15] propusieron una implementación paralela de la variante SACO para el VRP con el objetivo de reducir el tiempo de ejecución y mejorar la calidad de las soluciones obtenidas. Para incorporar el paralelismo se plantearon tres estrategias: una multicolonia, un modelo maestro-esclavo de grano fino y un modelo híbrido que estructura en forma jerárquica una multicolonia en la que cada colonia

se implementa con un modelo maestro-esclavo. Se plantearon seis estrategias distintas de comunicación para el modelo multicolonia. La evaluación experimental se realizó utilizando instancias clásicas para el VRP de entre 75 y 199 clientes. La plataforma de ejecución utilizada fue un cluster de 64 procesadores. El estudio mostró que para el modelo maestro-esclavo la eficiencia mejora al trabajar con instancias de tamaño creciente del problema, obteniendo una eficiencia superior a 0.7 para 8 procesadores. Sin embargo, al considerarse más de 8 procesadores la eficiencia comienza a degradarse. No se realizaron pruebas para el modelo multicolonia, sino que directamente se estudió el modelo híbrido. Para el modelo híbrido, solamente se realizaron estudios de la calidad de las soluciones y el tiempo de ejecución, no realizando estudios sobre speedup ni eficiencia. Los autores también discutieron los aspectos tecnológicos de una posible implementación grid mediante web services, pero no lo implementaron.

Posteriormente, Doerner et al. [57] ampliaron sus trabajos previo, incorporando el estudio de un maestro-esclavo grano grueso en el que los esclavos trabajan sobre distintos subproblemas [59]. La evaluación experimental fue extendida considerando instancia de mayor complejidad (hasta 480 clientes), comprobando que se producen mejoras en la eficiencia al crecer el tamaño de las instancias.

- **Parallel Ant Colony Optimization for the Traveling Salesman Problem (2006)**

  Manfrin et al. [111] presentaron un modelo paralelo de $\mathcal{MM}$AS aplicado al TSP que omite la fase de reinicialización y utiliza la mejor solución hasta el momento para actualizar el rastro de feromona. Se propusieron los modelos de ejecuciones paralelas independientes y multicolonia sincrónicas y asincrónicas, considerando varias topologías de interconexión. El modelo multicolonia utiliza una frecuencia fija para intercambiar las mejores soluciones entre pares de colonias. Se sustituye la mejor solución encontrada por la colonia receptora cuando la solución recibida tenga mayor calidad que ella. La evaluación experimental se realizó en un cluster homogéneo de 8 PCs sobre instancias de la biblioteca TSPLIB de hasta 2392 ciudades. Se estudió solamente la calidad de las soluciones halladas por los algoritmos considerando como criterio de parada un tiempo de ejecución fijo, incluyendo una versión puramente secuencial. Se constató que todas las implementaciones propuestas presentaron mejores soluciones que la versión secuencial, destacándose las ejecuciones paralelas independientes por obtener los mejores resultados. Los autores atribuyeron este hecho a que el tiempo de ejecución es relativamente alto, teniendo las ejecuciones independientes múltiples búsquedas mientras que en las multicolonias, todas las colonias convergen rápidamente a la misma solución. Para comprobar esa afirmación, se redujo la frecuencia de la comunicación entre las colonias, obteniendo mejoras en los resultados. A partir de este experimento, establecieron que para obtener buenos resultados, la comunicación entre las colonias debería ser más sofisticada y dependiente del tamaño del problema y del tiempo de ejecución. Otras mejoras sugeridas por los autores para evitar la convergencia prematura en las variantes multicolonia son: incorporar mecanismos de reinicialización, utilizar otros criterios de aceptación de las soluciones migrantes (por ejemplo: cuando difieren menos de una cierta cantidad de componentes) y que cada colonia explore diferentes regiones del espacio de búsqueda.

- **Parallel Ant Colony Optimization for the Quadratic Assignment Problem with Symmetric Multi Processing (2008)**

  Tsutsui [160] extendió un trabajo previo [161] (presentado en la sección 4.5.2) presentando un modelo maestro-esclavo (propuesto en el trabajo original), un modelo de islas y un modelo de ejecuciones paralelas independientes. El modelo de islas trabaja en forma sincrónica con una topología completa y un proceso central que se encarga de intercambiar las mejores soluciones entre las islas con una cierta frecuencia. Se plantean dos estrategias de intercambio: la mejor solución se envía a todas las islas y la mejor solución es enviada solamente a la isla que produjo la peor solución. En ambas estrategias, la isla receptora incorpora la solución recibida, eliminando a su peor solución. Tsutsui realizó la evaluación de las implementaciones paralelas sobre un equipo con 4 núcleos usando un criterio de parada de tiempo prefijado igual al del algoritmo secuencial. La evaluación experimental fue realizada sobre instancias de la biblioteca QAPLIB de hasta 100 locaciones. El estudio de la calidad de las soluciones obtenidas mostró que los modelos paralelos obtuvieron mejores soluciones que el algoritmo secuencial, destacándose el modelo maestro-esclavo por presentar los mejores resultados.

La tabla 4.3 presenta un resumen de las aplicaciones de paralelismo en las que se plantea más de un modelo sobre ACO relevadas. En las columnas se indica: autores principales (Autores), año de publicación (Año), variante de ACO utilizada (Variante), problema abordado (Problema), estrategia de paralelismo utilizada (Modelo) y plataforma utilizada (Plataforma). La categoría ACO corresponde a una variante específica para el problema abordado mientras que la categoría ND indica que la información no está disponible. Maestro-esclavo, multicolonia y ejecuciones paralelas independientes se abrevian como M-E, MC y EPI respectivamente.

Tabla 4.3: Propuestas de ACO paralelos con más de un modelo

Autores	**Año**	**Variante**	**Problema**	**Modelo**	**Plataforma**
Rahoual et al.	2002	ACO	SCP	EPI y M-E grano fino	COW
Islam et al.	2003	ACO	MANET	ND	COW
Alba et al.	2005, 2007	ACS	MTTP	EPI, M-E grano fino y MC	COW
Chu et al.	2005	ACO	PSPP	M-E grano fino y MC	COW
Doerner et al.	2005, 2006	ACO	VRP	MC, M-E grano fino, híbrido y M-E grano grueso (trabajan sobre diferentes subproblemas)	COW
Manfrin et al.	2006	$\mathcal{MM}$AS	TSP	EPI y MC	COW
Tsutsui	2008	cAS	QAP	EPI, MC y M-E grano fino	Híbrido

Generalmente, la utilización de más de un modelo de paralelismo ha involucrado la evaluación de los modelos de ejecuciones paralelas independientes, multicolonia y maestro-esclavo. Las ejecuciones paralelas independientes se han mostrado como una alternativa sencilla para la implementación del paralelismo, obteniendo speedup casi lineales y mejoras en la calidad de las soluciones obtenidas en comparación con los algoritmos secuenciales. Algunos autores han planteado dudas sobre la conveniencia o no de utilizar estrategias sofisticadas, cuando el modelo más simple permite alcanzar buenos resultados. Sin embargo, el éxito parece estar vinculado a que en el modelo de ejecuciones paralelas independientes se realizan varias exploraciones cortas del espacio de búsqueda, evitando así situaciones que se pueden producir en recorridas largas, como son alcanzar situaciones de estancamiento o de pérdida de diversidad de la búsqueda. Los modelos multicolonia y maestro-esclavo han confirmado las virtudes señaladas en sus respectivas evaluaciones, siendo en la práctica un poco superior en términos de eficiencia el modelo multicolonia. Por otro lado, los modelos híbridos han sido escasamente implementados. Las implementaciones de varios modelos relevadas han utilizado mayoritariamente como plataforma un cluster de PCs, existiendo una amplia diversidad de problemas abordados.

4.6. Conclusiones

En este capítulo se presentó una introducción a los conceptos propios de las técnicas de alto desempeño, destacándose una descripción de las arquitecturas de computadoras paralelas existentes y las métricas más usadas para la evaluación de algoritmos paralelos.

Asimismo, se presentó un relevamiento sobre las estrategias de paralelismo aplicado a ACO. No existe una taxonomía de estrategias de aplicación de paralelismo a ACO ampliamente reconocida por la comunidad académica. La taxonomía de Randall y Lewis no es del todo satisfactoria, por lo cual se propuso una ampliación que incorpora una categoría novedosa sobre ACO, el modelo celular.

Se realizó un amplio relevamiento de trabajos que implementan versiones paralelas de ACO, centrado en trabajos realizados sobre clusters de PCs y equipos de memoria compartida. Es posible apreciar un desbalance entre los distintos modelos de paralelismo, siendo multicolonia y maestro-esclavo de grano fino los más implementados, siguiéndoles las ejecuciones paralelas independientes. Por otro lado, no existen implementaciones del modelo celular y del modelo maestro-esclavo de grano muy fino, mientras que los modelos híbridos son escasos. La mayor parte de las implementaciones ha sido desarrollada utilizando pasaje de mensajes sobre arquitecturas de tipo cluster, existiendo escasas experiencias de implementaciones con equipos de memoria compartida. El problema elegido para evaluar las diferentes propuestas ha sido mayoritariamente el TSP, aunque recientemente se ha comenzado a diversificar el espectro de problemas utilizados.

A partir del relevamiento realizado, es posible concluir que la aplicación de paralelismo a ACO es promisoria, existiendo una cantidad considerable de trabajos que muestran que es posible obtener mejoras en la calidad de las soluciones obtenidas en comparación con los algoritmos secuenciales, reducir el tiempo de ejecución y obtener speedups casi lineales. Asimismo, es posible identificar dos líneas de aplicación de paralelismo a ACO poco exploradas: la implementación sobre equipos de memoria compartida y la implementación de un modelo celular. Se considera conveniente concentrarse en la implementación de un modelo celular para ACO, al no existir implementaciones conocidas de este modelo.

Capítulo 5

Caso de aplicación: el Problema de Steiner Generalizado

"Quise continuar la huella de los antiguos cantores"
Fernando Cabrera, *Disolvente*

La evaluación de las ideas presentes en las técnicas metaheurísticas para la resolución de problemas de optimización requiere la puesta en práctica de esas ideas, mediante su aplicación sobre un problema concreto y la realización de estudios empíricos.

El Problema de Steiner Generalizado (GSP, por sus siglas en inglés [170]) modela el diseño de redes de comunicaciones confiables en las cuales se exigen requisitos de conexión tratando de garantizar con alta probabilidad la comunicación entre nodos distinguidos, llamados terminales. El GSP es un problema NP-difícil que ha sido poco abordado por la comunidad científica. En el InCo se han aplicado técnicas evolutivas para resolver el GSP utilizando un enfoque en el que se eliminan paulatinamente enlaces de la red original, obteniendo resultados de muy buena calidad. Por otro lado, se han formulado propuestas que trabajan en forma constructiva, pero no han logrado obtener resultados de similar calidad. La experiencia de trabajo en el InCo sobre el GSP y la característica de ser un problema fuertemente ligado a la construcción de caminos, vuelven al GSP un candidato ideal para la puesta en práctica de las ideas de las técnicas ACO presentadas en los capítulos anteriores.

En este capítulo se presenta una descripción del problema considerado para explorar la técnica ACO, así como un relevamiento de los enfoques existentes para su resolución. La estructura del capítulo es la siguiente. En la próxima sección se presenta el problema estudiado, su modelo matemático y variantes del problema. Posteriormente, se resumen los trabajos relacionados con la aplicación de técnicas metaheurísticas para la resolución del problema. Luego, se presenta un resumen de trabajos de ACO aplicado a problemas con características similares al GSP. Finalmente, en la última sección se presentan las conclusiones del relevamiento.

5.1. El Problema de Steiner Generalizado

Diseñar una topología de interconexión entre los nodos de una red de forma que se cumplan algunas características de confiabilidad es uno de los problemas más importantes en el ámbito de las redes de comunicaciones. La confiabilidad de una red es una

medida que evalúa su capacidad de continuar operativa en caso de fallos en sus componentes y permite determinar la posibilidad de éxito en la comunicación entre pares de nodos. Es posible plantear modelos que minimizan el costo total de la red satisfaciendo los requisitos de conexión sin agregar redundancia de caminos. Pero las soluciones que se obtienen son de baja confiabilidad, ya que la red no continúa operativa si se producen fallas en los nodos o en los enlaces.

El GSP consiste en el diseño de una subred de mínimo costo que verifique una cierta cantidad de requerimientos prefijados de conexión entre pares de nodos distinguidos denominados terminales. Al utilizarse el GSP para el diseño de redes de comunicaciones se garantiza un cierto nivel de confiabilidad debido a la existencia de caminos alternativos entre nodos terminales. El funcionamiento de la red resultante es más robusto, siendo más resistente a fallas en sus componentes.

5.1.1. Formalización del GSP

La formulación del GSP es la siguiente [45, 144, 170]. Dados:

- $G = (V, E)$ un grafo no dirigido, siendo V el conjunto de nodos, E el conjunto de aristas y C la matriz de costos asociados a las aristas.
- T un conjunto distinguido de nodos terminales, tal que $2 \leq |T| \leq |V|$.
- $R = \{r_{ij} | i, j \in T\}$ una matriz de dimensión $|T| * |T|$ cuyos elementos $r_{ij} \in \mathbb{Z}^+$ indican la cantidad de caminos disjuntos exigidos entre todo par de nodos de T.

El GSP consiste en hallar un subgrafo G_T de G de costo mínimo, tal que para todo par de nodos $i, j \in T$ existan r_{ij} caminos disjuntos (que no compartan aristas), entre los nodos i y j en G_T. Se dice que los nodos i y j son r_{ij} arista-conexos en G_T. Existe otra variante del problema, nodo-conexa, en la cual los caminos disjuntos no deben compartir los nodos.

No se plantean requisitos de conectividad sobre los nodos que no pertenecen al conjunto de nodos terminales. Estos nodos son llamados *nodos de Steiner*, pudiendo o no formar parte de la solución óptima.

El GSP puede ser modelado matemáticamente como un problema de programación lineal entera. El modelo en su versión arista-conexa se basa en definir una variable de decisión binaria x_{ij} para cada arista $(i, j) \in E$ que indica la presencia o ausencia de la arista en la solución. Adicionalmente, se utiliza la variable real y_{ij}^{kl} para indicar la utilidad (cantidad de veces que se utiliza) de la arista (i, j) en la dirección i a j, en un camino que une el nodo terminal k con el nodo terminal l.

Un modelo matemático del GSP como problema de programación lineal entera se presenta en la fórmula 5.1 [127, 144]. En el modelo, la fórmula 5.1a presenta la función objetivo del problema que corresponde a la minimización de la suma de los costos de las aristas incluidas en la solución. La restricción de la fórmula 5.1b acopla la utilización de las aristas en cualquier sentido (y_{ij}^{kl} y y_{ji}^{kl}) a la solución del problema (x_{ij}). Las fórmulas 5.1c y 5.1d incorporan al modelo las restricciones que aseguran que se cumpla los requerimientos de conectividad para el par de nodos terminales k y l. La fórmula 5.1e corresponde a la restricción de la conservación de flujo. Finalmente, las restricciones de integridad y no negatividad para las variables usadas en el modelo están dadas por las

fórmulas 5.1f, 5.1g y 5.1h. El conjunto de aristas $\{(i,j)|u_{ij} = 1\}$, siendo $U = \{u_{ij}\}$ la solución óptima al problema, define al subgrafo solución G_T.

$$\text{mín} \sum_{(i,j)\in E} C_{ij} \cdot x_{ij} \tag{5.1a}$$

sujeto a:

$$x_{ij} \geq y_{ij}^{kl} + y_{ji}^{kl}, \forall (i,j) \in E, \forall k,l \in T, k \neq l \tag{5.1b}$$

$$\sum_{(k,j)\in E} y_{kj}^{kl} \geq r_{kl}, \forall k,l \in T, k \neq l \tag{5.1c}$$

$$\sum_{(i,l)\in E} y_{il}^{kl} \geq r_{kl}, \forall k,l \in T, k \neq l \tag{5.1d}$$

$$\sum_{(p,j)\in E} y_{pj}^{kl} - \sum_{(i,p)\in E} y_{ip}^{kl} \geq 0, \forall k,l \in T, \forall p \in V \setminus \{k,l\} \tag{5.1e}$$

$$x_{ij} \in \{0,1\}, \forall (i,j) \in E \tag{5.1f}$$

$$y_{ij}^{kl} \geq 0, y_{ij}^{kl} \in \mathbb{R}, \forall (i,j) \in E, \forall k,l \in T, k \neq l \tag{5.1g}$$

$$r_{kl} \geq 0, \forall k,l \in T, k \neq l \tag{5.1h}$$

5.1.2. Variantes y complejidad de los problemas de Steiner

El GSP pertenece a la clase de los problemas de Steiner. En esta clase se encuentran otros problemas que flexibilizan las condiciones de conectividad impuestas sobre los pares de nodos terminales en el GSP. Los problemas de k-conexión exigen una cantidad fija k de caminos disjuntos entre pares de nodos terminales, existiendo una formulación que impone como condición que los caminos no pueden compartir aristas y otra en la que no pueden compartir nodos. Un caso particular de problema de k-conexión es el problema del árbol de Steiner (STP, por sus siglas en inglés). El STP exige únicamente la existencia de un camino entre pares de nodos terminales, por lo que la solución al problema es un árbol de cubrimiento de costo mínimo que incluye a todos los nodos terminales. Otro tipo de problemas se caracteriza por estar formulados sobre grafos en los que los nodos están dados por sus coordenadas geométricas en un plano y el costo asociado a las aristas es la distancia euclidiana entre los nodos. Un caso particular de este tipo de problemas es el problema rectilíneo del árbol de Steiner (RSMT, por sus siglas en inglés) en el que las aristas son horizontales (ambos nodos de la arista tienen la misma coordenada y) o verticales (ambos nodos de la arista tienen la misma coordenada x). El problema rectilíneo del árbol de Steiner, al igual que STP, solamente exige la existencia de un camino entre los pares de nodos terminales.

Recientemente, Voß realizó una amplia reseña sobre el STP y su aplicación en telecomunicaciones [164]. En el trabajo de Voß se describían las variantes del problema y los enfoques más utilizados para la resolución del STP. Aunque la reseña incluía varias generalizaciones del STP, no incorpora los problemas de k-conexión ni el GSP.

Karp probó en 1972 que el STP, el problema más simple de la clase de los problemas de Steiner, es NP-difícil [102]. El GSP también pertenece a la clase de problemas NP-difíciles [45]. La complejidad de los problemas de Steiner hace que sean menos tratables con algoritmos exactos al aumentar el tamaño de las instancias consideradas, por lo que

la utilización de algoritmos exactos para su resolución queda limitada a instancias de tamaño reducido.

5.2. Antecedentes sobre el GSP

El GSP ha sido poco abordado por la comunidad científica. Del relevamiento bibliográfico solamente surgen pocos trabajos que han estudiado en profundidad este problema y solamente un enfoque que utiliza ACO para su resolución.

El enfoque existente para resolver el GSP mediante un algoritmo ACO fue propuesto por Robledo en 2001 [144]. Robledo estudio el GSP y realizó un relevamiento de los enfoques existentes para su resolución. Adicionalmente, consideró otros problemas de diseño de redes que presentan fuerte similitudes al GSP. El trabajo de Robledo se centró en proponer algoritmos exactos para la resolución de dichos problemas, orientando sus propuestas hacia algoritmos paralelo-distribuidos. Por otro lado, Robledo analizó la posibilidad de utilizar metaheurísticas para obtener soluciones aproximadas al problema, señalando la dificultad de definir estructuras de vecindad que permitan explotar búsquedas locales. A partir de su análisis, Robledo propuso un algoritmo ACO con el objetivo de mostrar la factibilidad de utilizar dicha metaheurística para la resolución del GSP.

El esquema general del método planteado por Robledo se presenta a continuación. Para cada par de terminales y para cada uno de los requisitos de conexión, se utiliza un ACO independiente para obtener el camino más corto entre los terminales. Cada ACO construye en forma secuencial el camino más corto entre los nodos a partir del conjunto de aristas disponibles, siguiendo un procedimiento similar al utilizado para resolver el TSP [68]. En la propuesta de Robledo, el criterio de selección del camino solución de un requisito es el que minimice el costo del camino dividido la cantidad de terminales que lo componen, privilegiando los caminos que pasan por más terminales ante costos similares. De este modo, se incorpora a la solución parcial aristas que posiblemente sean reutilizadas para resolver otros requisitos de conexión. Las aristas utilizadas para construir cada camino son eliminadas de las aristas disponibles para el mismo par de terminales, para asegurar que los caminos construidos sean disjuntos. Cada vez que se procesa un nuevo par de terminales, todas las aristas se consideran como disponibles. El procedimiento de construcción de caminos descripto garantiza la factibilidad de las soluciones. El enfoque utilizado por Robledo carece de una visión global de la solución que está siendo construida, ya que todos los subproblemas se resuelven en forma independiente, sin considerar la estrecha relación existente entre los caminos construidos. La evaluación experimental fue realizada sobre grafos propuestos por el propio autor, no pudiendo obtener el óptimo en instancias pequeñas del problema (por ejemplo: 10 nodos, 15 aristas y 4 terminales) y mostrando una escalabilidad pobre para instancias del GSP de mayor dimensión. Los inconvenientes de la propuesta de Robledo pueden atribuirse a que las aristas que están incluidas en una solución parcial no son privilegiadas cuando se construyen nuevos caminos. La incorporación de mecanismos que permitan privilegiar las aristas que ya forman parte de la solución podría contribuir a disminuir el costo total de la red construida.

Posteriormente, Aroztegui et al. [11] abordaron la resolución del GSP utilizando un algoritmo genético paralelo (PGA, por sus siglas en inglés). El PGA propuesto se basa en una codificación binaria sencilla, en la que cada cromosoma consta de un arreglo

de bits que representan a las aristas del grafo original. La presencia o ausencia de una arista en un grafo solución se indica con el valor 1 o 0 respectivamente, en la posición correspondiente a la arista en el cromosoma. Al utilizar la representación binaria los operadores evolutivos resultan sencillos, pero tienen el inconveniente de que los cromosomas resultantes pueden representar a una solución no factible del GSP. Por este motivo, se debe incorporar en los operadores evolutivos la verificación de la factibilidad de los cromosomas creados al aplicarlos. El enfoque de la propuesta adopta como criterio descartar los individuos no factibles, a partir de una propuesta previa de Esbensen sobre el problema del árbol de Steiner [73]. La decisión de trabajar con individuos factibles evita agregar complejidad en el diseño de los operadores evolutivos, pero requiere incorporar un procedimiento adicional que determine si el grafo representado cumple las restricciones de conectividad, luego de aplicar los operadores evolutivos. El operador de mutación utilizado es el de inversión del valor de un alelo, mientras que se usa el operador de cruzamiento de un punto. En la propuesta de Aroztegui et al. [11] se considera como función de aptitud la sustracción del valor de costo de la solución representada por el individuo del valor del costo del grafo original. La población se inicializa aleatoriamente, mediante la eliminación de aristas del grafo original. El algoritmo paralelo propuesto trabajó con 4 subpoblaciones, aplicando un operador de migración considerando topologías de conexión de anillo unidireccional y grafo completo. La evaluación experimental fue realizada sobre los grafos propuestos por Robledo e instancias de tamaño mediano generadas aleatoriamente (hasta 100 nodos, 500 aristas y 25 terminales), obteniendo resultados que mostraron la aplicabilidad del enfoque propuesto. El modelo de poblaciones distribuidas utilizado permitió obtener mejoras significativas de eficiencia respecto al modelo serial.

En paralelo al trabajo anterior, Pedemonte y Nesmachnow [136] realizaron un estudio empírico de operadores de cruzamiento sobre un AG serial, considerando las familias de operadores de cruzamiento de n puntos y uniforme parametrizados. La evaluación experimental fue realizada sobre los grafos utilizados por Aroztegui et al., mostrando que las estrategias disruptivas obtuvieron los mejores resultados. En particular, los resultados obtenidos por el operador de cruzamiento uniforme lo posicionaron como un operador promisorio para el GSP.

Calegari [31] evaluó alternativas para la representación planteada para el PGA. Las alternativas consideradas por Calegari fueron las codificaciones basadas en lista de caminos y en subgrafos de requerimientos. En la representación mediante lista de caminos se utiliza un arreglo de genes del tamaño de la cantidad de restricciones existentes, en el que cada gen tiene una lista de los caminos disjuntos posibles entre los terminales asociados a la restricción. Cada camino se compone de la lista de identificadores de los nodos que lo integran. Inicialmente, los individuos se generan como grafos aleatorios que cumplan las restricciones y para cada restricción se almacenan los caminos disjuntos necesarios para satisfacerla. Se utiliza el operador de cruzamiento de un punto, intercambiando para cada restricción todos los caminos. La mutación se realiza sobre un gen y consiste en obtener tantos caminos disjuntos como sean necesarias para satisfacer la restricción. La principal ventaja que tienen los operadores evolutivos diseñados es que mantienen la factibilidad de los individuos, evitando la realización de chequeos explícitos. La representación basada en subgrafos de requerimientos es similar a la de la lista de caminos, pero para cada requerimiento se almacena su subgrafo solución. Los operadores evolutivos son similares a los de la representación basada en lista de

caminos. La evaluación experimental de las propuestas de Calegari fue realizada sobre un subconjunto de las instancias utilizadas por Robledo y Aroztegui et al., obteniendo en las primeras iteraciones soluciones de buena calidad, pero en iteraciones sucesivas no se logran mejoras, produciéndose un estancamiento en la evolución del algoritmo. Los problemas que presentan las propuestas de Calegari son la independencia con la que se resuelven los subproblemas (perdiendo de vista la estrecha relación existente entre los caminos construidos para distintas restricciones) y la excesiva redundancia en la codificación (no es considerada al diseñar los operadores evolutivos).

Siguiendo el enfoque de la propuesta PGA, Nesmachnow et al. [127, 128, 129] analizaron comparativamente métodos secuenciales y paralelos aplicados a la resolución del GSP. Los métodos comparados fueron un GA, SA, CHC, VNS y los *híbridos débiles* (combinación de más de un algoritmo) GA+SA y GA+VNS. Todos los algoritmos propuestos utilizan la misma codificación binaria y operadores simples, sin incluir conocimiento específico del problema y descartando las soluciones no factibles generadas. El SA utiliza la inversión de los valores de cinco aristas como operador de transición y un esquema proporcional para el decaimiento de la temperatura. El CHC utiliza selección elitista, cruzamiento uniforme con restricciones en la recombinación y la reinicialización está basada en el operador de transición de SA (aplicado iterativamente hasta obtener una solución factible). El VNS define las vecindades a partir de la distancia de Hamming en la codificación binaria. En los híbridos débiles se incorporan SA y VNS como operadores internos del GA, aplicándose con posterioridad a los operadores evolutivos tradicionales. Los algoritmos paralelos utilizan un modelo de poblaciones distribuidas, aplicando un operador de migración considerando una topología de conexión de anillo unidireccional. La evaluación experimental fue realizada sobre las instancias de mediano tamaño utilizadas previamente por Aroztegui et al. e instancias replicadas con óptimos conocidos (hasta 208 nodos, 481 aristas y 78 terminales). Los algoritmos secuenciales CHC y el híbrido GA+VNS fueron capaces de obtener las mejores soluciones. Los resultados de los algoritmos paralelos se obtuvieron trabajando con 8 subpoblaciones, mostrando que tanto el GA como el híbrido GA+SA son capaces de hallar las mejores soluciones en términos de costo y speedup.

5.3. ACO aplicado a problemas similares al GSP

En esta sección se presenta un resumen de problemas que presentan fuertes similitudes con el GSP y que han sido abordados mediante la técnica ACO. La sección se divide en tres subsecciones que están organizadas de acuerdo a las características de los problemas. En primer lugar, se reseñan trabajos en los cuales se estudian otros problemas de Steiner. Posteriormente, se presentan trabajos que involucran determinar caminos arista-disjuntos entre pares de nodos. Finalmente, se reseñan trabajos sobre problemas vinculados a la obtención de árboles de cubrimiento.

Existen otras aplicaciones de ACO sobre problemas de diseño de redes que no presentan suficientes puntos de contacto con el GSP como para ameritar incluir su reseña en este capítulo. Se ha omitido explícitamente el análisis de problemas de diseño de redes que involucran el manejo de flujos en función de las demandas [7, 87, 141] porque difiere del modelo considerado en el GSP.

El relevamiento permite concluir que cada propuesta de ACO aplicada a un problema particular incorpora una cantidad significativa de información propia del problema

considerado. Esta situación contrasta con otras propuestas de resolución, mostrando que ACO es una técnica menos genérica que otra metaheurísticas.

5.3.1. Otros problemas de Steiner

En esta subsección se presentan los trabajos vinculados a problemas de la clase de Steiner. La subsección se estructura del siguiente modo. En primer lugar se presentan los trabajos que abordan el problema del árbol de Steiner. Posteriormente, se introduce el problema Multi-source Single-destination Data-Centric routing y se comentan los trabajos vinculados a su resolución. Finalmente, se presenta los trabajos que abordan el RSMT.

Problema del árbol de Steiner

A pesar de los puntos de contacto existentes entre el STP y la metodología constructiva de las soluciones seguida por ACO, el STP ha sido poco abordado con ACO. Dos enfoques han sido propuestos para la resolución del STP mediante ACO y se describen a continuación.

Das et al. [48, 88, 146] propusieron un enfoque caracterizado por considerar varias hormigas que cooperan para construir un único árbol. Las hormigas se ubican sobre terminales distintos y por turnos se mueven por una arista, incorporándola a su solución. Para seleccionar el orden en que las hormigas realizan los movimientos, los autores consideran diversos esquemas de ordenamiento: fijo, aleatorio o desordenado. Cuando dos hormigas se encuentran, fusionan sus soluciones y solamente una continua el proceso de construcción. Das et al. consideran la posibilidad de que una hormiga llegue a un callejón sin salida tal que no pueda realizar más movimientos sin volver a un nodo ya visitado, proponiendo deshacer movimientos hasta otro nodo en la solución (solamente se realiza un reposicionamiento para que la hormiga pueda continuar con el proceso de construcción, no se eliminan aristas de la solución). La visibilidad de una arista se define mediante una expresión inversamente proporcional a su costo y al costo mínimo para fusionar la solución considerada con otra solución usando esa arista. La evaluación experimental fue realizada sobre instancias sin reducir del tipo B de la OR-library [3], obteniendo soluciones a un 3 % del óptimo en el peor caso.

Posteriormente, Luyet et al. [110] propusieron otro enfoque para aplicar ACO al STP, en el que cada hormiga construye una solución independiente del resto. Cada hormiga en cada paso debe agregar un terminal a su solución, por lo que incorpora un camino que permita conectar la solución al terminal. La visibilidad se calcula a partir de una expresión inversamente proporcional al costo del camino más corto entre la solución y el terminal considerado. El depósito de feromona se realiza en los nodos de Steiner, calculando el valor asociado a la experiencia adquirida en la búsqueda como el promedio de los rastros de feromona en los nodos de Steiner del camino más corto entre la solución y el terminal considerado. La propuesta de Luyet et al. incorpora una actualización local de los rastros de feromona. Luyet et al. aplicaron reducciones a los grafos previamente a la ejecución del algoritmo ACO. Las reducciones pueden ser de dos tipos: exclusiones o inclusiones. Las exclusiones corresponden a identificar aristas y/o nodos de Steiner que no forman parte de la solución óptima. Por el contrario, las inclusiones identifican aristas y/o nodos de Steiner que necesariamente formarán parte de la solución óptima. La utilización de reducciones sobre las instancias del STP es fuertemente recomendada

por Voß [164]. Luyet et al. evaluaron experimental su propuesta sobre instancias del tipo B, C y D de la OR-library, obteniendo resultados competitivos con un TS que presenta excelentes resultados para el STP, aunque a costa de un mayor tiempo de ejecución.

Multi-source Single-destination Data-Centric routing

El problema Multi-source Single-destination Data-Centric routing (MSDC, por sus siglas en inglés) consiste en obtener el ruteo óptimo en una red en la que se dispone de múltiples fuentes y los datos deben enviarse hacia un único destino. En el MSDC los datos pueden ser agregados en nodos intermedios, por lo cual el objetivo del problema es encontrar el árbol óptimo que incluye los nodos fuentes y destino de forma de que realice las agregaciones óptimas, es decir evitando el envío de datos duplicados a través de la red. El MSDC presenta muchos puntos de contacto con el STP. Por otro lado, el MSDC se diferencia del STP en que no existe un conocimiento global de los nodos y los enlaces, resultando natural la utilización de algoritmos de resolución distribuidos.

Das et al. [146] abordaron el MSDC mediante un ACO dinámico que utiliza dos tipos de hormigas, unas que se mueven desde las fuentes al destino obteniendo información de la red y otras que se mueven desde el destino a las fuentes actualizando la información de las rutas. La propuesta de Das et al. es una adaptación de su enfoque previo para el STP, ya que construyen un árbol de Steiner considerando la agregación de los costos. La evaluación experimental fue realizada sobre instancias de hasta 200 nodos. Los resultados obtenidos fueron superiores a los obtenidos mediante la técnica address-centric routing, en la que el ruteo se realiza de forma de obtener los caminos de costo óptimo entre las fuentes y el destino, sin considerar especialmente las agregaciones.

Problema rectilíneo del árbol de Steiner

Hu et al. [97, 171] propusieron un enfoque para el RSMT idéntico al utilizado por Das et al. para el STP. La propuesta de Hu et al. incorpora la realización de reducciones de exclusión sobre los grafos en una etapa previa a la ejecución del algoritmo ACO. También se incorporan mejoras que dependen fuertemente de las particularidades de las topologías resultantes al considerar un problema rectilíneo. Hu et al. realizaron la evaluación experimental de su propuesta sobre instancias generadas mediante GeoSteiner [1] de hasta 500 terminales. Si bien GeoSteiner es la implementación de un algoritmo exacto para la resolución del RSMT, también provee de rutinas que permiten la generación de instancias para el problema. La evaluación de la propuesta fue realizada comparando contra la heurística específica BGTC (Batched Greedy Triple Construction) y GeoSteiner. Para instancias pequeñas de hasta 50 terminales, los resultados obtenidos por la propuesta de Hu et al. son casi óptimos con tiempos de ejecución bajos. Al considerarse instancias de mayor complejidad, la propuesta de Hu et al. obtuvo soluciones a un 1 % del óptimo, utilizando un tiempo de ejecución significativamente menor que GeoSteiner. Sin embargo, la heurística BGTC superó al algoritmo ACO, ya que presentó mejores resultados y tuvo un menor tiempo de ejecución.

5.3.2. Problema de caminos arista-disjuntos

Dado un grafo y una colección de pares de nodos que representan requisitos de conexión, el problema de caminos arista-disjuntos (EDP, por su sigla en inglés) consiste en obtener la mayor cantidad de caminos mutuamente disjuntos en sus aristas sobre el

grafo. El EDP pertenece a la clase de problemas NP-difíciles [102]. Existen dos líneas de investigación distintas sobre la aplicación de ACO al EDP, cuyos detalles se presentan a continuación.

Blesa y Blum [17, 18] propusieron un enfoque ACO caracterizado porque cada solución es construida cooperativamente entre varias hormigas, aunque en cada iteración se construye más de una solución. La construcción de una solución se realiza utilizando tantas hormigas como requisitos de conexión existan, asignándose a cada hormiga para su resolución un subproblema distinto. El ACO incluye una etapa de depuración de la solución a partir de la combinación de las soluciones obtenidas para cada subproblema, en la que se van retirando de la solución aquellos caminos que tienen más aristas en común con el resto de los caminos hasta obtener un conjunto de caminos que sea arista-disjuntos. El manejo de los rastros de feromona propuesto utiliza una matriz de feromona independiente para cada subproblema. Blesa y Blum consideran que una solución es mejor que otra si la solución tiene una mayor cantidad de caminos arista-disjuntos. Si las soluciones empatan en la cantidad de caminos disjuntos, se considera como mejor solución la que tiene una menor cantidad de aristas que pertenezcan a la intersección de sus caminos. La regla de transición de estados planteada para el EDP busca reflejar ambos criterios, incorporando un término que refleja el uso de la arista en la solución que se está construyendo, mientras que se considera como valor heurístico el costo de la arista a incluir sumado al costo del camino más corto para completar la solución. La propuesta incorpora una evaporación adicional al finalizar la construcción de una solución para generar diversidad. El algoritmo ACO incluye un mecanismo para intentar escapar de óptimos locales, que consiste en eliminar parte de la solución cuando se detecta el estancamiento del algoritmo. Blesa y Blum evaluaron su propuesta comparándola contra un algoritmo greedy con inicios múltiples (MSGA, por sus siglas en inglés) que construye en forma secuencial el camino más corto para un requisito de conexión y marca las aristas como no disponibles para el resto de los requisitos. El criterio de parada utilizado para la comparación fue de esfuerzo prefijado. Las instancias de evaluación (hasta 500 nodos, 1020 aristas y 75 requisitos de conexión) fueron generadas por los autores. Los resultados obtenidos por el ACO propuesto fueron superiores a los del MSGA, encontrando mejores soluciones y más rápidamente (en etapas previas de la búsqueda).

Posteriormente, Blesa y Blum [19] realizaron mejoras sobre su propuesta, introduciendo la construcción en paralelo de soluciones, permitiendo intercalar etapas de las distintas hormigas en lugar de construir las soluciones en forma secuencial. Las soluciones construidas en paralelo difieren de las construidas secuencialmente, debido a que la regla de transición de estados tiene un término que refleja el uso de la arista en la solución que se está construyendo y la evaporación adicional al completarse una solución. Otra mejora introducida es el manejo de una lista de candidatos para restringir la cantidad de aristas consideradas en cada paso de la construcción. Los resultados obtenidos por la propuesta mejorada son superiores a los obtenidos por la propuesta original.

Un aspecto negativo de la propuesta de Blesa y Blum es que utiliza una gran cantidad de parámetros, siendo diez para la propuesta original y quince para la propuesta mejorada. Si bien seis parámetros están determinados previamente, es necesario calibrar una cantidad importante de parámetros o guiarse puramente por la experiencia existente sobre problemas similares.

El otro enfoque existente para la resolución del EDP mediante ACO fue realizado por

Nowé et al. [131]. Estos autores proponen la utilización de diferentes clases de hormigas (Multi-type Ant Colony) caracterizadas por utilizar distintos tipos de feromona. Cada hormiga es atraída por los rastros de feromona de su tipo y repelida por los rastros de los otros tipos de feromona. Los autores señalan que la inclusión de distintos tipos de hormigas es provechosa en problemas en los cuales se combina la colaboración con la competencia. El EDP es un problema que presenta características de colaboración (se debe maximizar la cantidad de requisitos de conexión satisfechos) y de competencia (los caminos no pueden tener aristas en común, por lo que se compite por la utilización de las aristas). En la propuesta de Nowé et al. cada componente tiene asociado un conjunto de rastros de feromona por cada tipo de hormiga. En el caso del EDP los tipos de rastros coinciden con la cantidad de requisitos de conexión. Para reflejar en la regla de transición de estados los múltiples tipos de rastro de feromona se incorpora un término que depende en forma inversamente proporcional a la cantidad de feromona de otro tipo presente en la componente. De esta forma, cada tipo de hormiga trata de obtener la mejor solución a un subproblema intentado que sea disjunta a las soluciones obtenidas por los otros tipos de hormiga. Con ese mismo objetivo se modifica la definición del costo de la solución, penalizando la utilización de aristas compartidas entre distintos tipos de hormigas. De este modo se procura evitar que todos los tipos de hormiga converjan al mismo camino, ya que los tipos de hormiga no tienen asignado previamente un subproblema específico. El riesgo de que más de un tipo de hormigas intente resolver el mismo subproblema se acrecienta cuando existe más de un requisito de conexión sobre el mismo par de nodos. Nowé et al. trabajaron solamente con grafos pequeños (hasta 15 nodos, 24 aristas y 4 requisitos de conexión), para mostrar la viabilidad de su planteo. Posteriormente, extendieron su trabajo sobre un subconjunto de las instancias utilizadas por Blesa y Blum, logrando resultados de similar calidad [166].

5.3.3. Problemas de cubrimiento

Dado un grafo, el problema del árbol de cubrimiento de costo mínimo (MST, por sus siglas en inglés) consiste en obtener el subgrafo acíclico que contenga todos los nodos del grafo y que minimice su costo de construcción. La versión tradicional del MST puede ser resuelto mediante algoritmos exactos con tiempo polinomial, como los propuestos por Kruskal [107] y Prim [138]. Sin embargo, se han propuesto variantes del MST que imponen restricciones adicionales sobre el árbol de cubrimiento y que pertenecen a la clase de problemas NP-difíciles. En esta subsección se presentan variantes del MST que han sido resueltas utilizando ACO. La subsección se estructura del siguiente modo. En primer lugar se presentan los trabajos que abordan el problema generalizado del árbol de cubrimiento de costo mínimo. Posteriormente, se comentan los trabajos vinculados al problema del árbol de cubrimiento de costo mínimo con restricciones de grado. Finalmente, se presentan los trabajos que abordan el problema del árbol de cubrimiento de costo mínimo con restricciones de diámetro.

Problema generalizado del árbol de cubrimiento de costo mínimo

En el problema generalizado del árbol de cubrimiento de costo mínimo (GMST, por sus siglas en inglés) el conjunto de nodos del grafo se encuentra particionado en subconjuntos disjuntos. El GMST consiste en obtener el árbol de cubrimiento de costo mínimo incluyendo un nodo de cada uno de los subconjuntos disjuntos. Algunos autores

suelen distinguir entre el E-GSMT, en el cual se debe utilizar solamente un nodo de cada subconjunto, y el L-GMST, donde se debe utilizar por lo menos un nodo de cada subconjunto.

Shyu et al. [145] abordaron el L-GMST mediante ACO utilizando un enfoque que se caracteriza por considerar en la regla de transición de estados las aristas salientes de nodos que forman parte del árbol y entrantes en nodos que no forman parte del árbol. Este enfoque es una leve modificación del típicamente utilizado para el TSP, ya que en ese caso solamente se consideran las aristas salientes del último nodo incluido en la solución. Shyu et al. plantean la utilización de tres fórmulas distintas para la visibilidad de las componentes en la regla de transición de estados, de forma de ponderar en forma distinta los nodos que pertenecen a subconjuntos no visitados, al mismo subconjunto y a subconjuntos ya visitados. La evaluación experimental de la propuesta fue realizada sobre instancias propuestas por Dror et al. [70] de hasta 500 nodos, 5000 aristas y 50 subconjuntos. El algoritmo ACO obtuvo resultados de calidad similar a un GA propuesto por Dror et al., utilizando la cuarta parte del tiempo de ejecución.

Posteriormente, Das et al. [47] plantearon otro enfoque para el L-GMST, que utiliza el mecanismo de construcción de soluciones que se describe a continuación. Inicialmente, todas las aristas y subconjuntos de nodos se marcan como disponibles y sin cubrir. El primer subconjunto de nodos se selecciona aleatoriamente en forma proporcional a la cantidad de nodos que lo componen, marcándolo como cubierto. A partir de los nodos del subconjunto considerado, se selecciona una arista que una nodos entre distintas particiones, mediante la regla de transición de estados. La arista seleccionada se incluye en la solución y se marcan sus nodos como visitados. Para seleccionar el resto de las aristas, se construye para cada nodo marcado un bosque en el subconjunto al cual pertenece y se consideran como elegibles las aristas entre particiones que entran o salen de un nodo que es hoja de uno de los bosques. Mediante el procedimiento descrito previamente se van incorporando las aristas entre particiones. Finalmente, cuando todos los subconjuntos han sido conectados entre si, se agregan las aristas necesarias dentro de los subconjuntos a partir de los bosques determinados previamente para que el árbol resulte conexo. La variante de ACO implementada por Das et al. utiliza dos tipos de hormigas que tienen una ponderación distinta para la visibilidad, resultando un tipo con una estrategia más greedy que el otro. La cantidad total de hormigas es constante durante la ejecución y con el paso de las iteraciones aumenta la cantidad de hormigas que tienen una estrategia más greedy. De este modo se pretende que la regla de transición de estados pase de ser probabilística en las primeras iteraciones a ser prácticamente determinista en las últimas. Los autores evaluaron su propuesta contra el mismo GA utilizado en la comparación del ACO de Shyu et al. sobre un subconjunto de las instancias de prueba. Sin embargo, realizan solamente tres ejecuciones por instancia, con lo cual la validez estadística de los resultados no queda demostrada y por tanto no se pueden sacar conclusiones definitivas. De todas formas, los resultados obtenidos son similares a los obtenidos con GA, requiriendo la construcción de una menor cantidad de soluciones.

Problema del árbol de cubrimiento de costo mínimo con restricciones de grado

Otra variante del MST que ha sido abordado mediante ACO es el problema del árbol de cubrimiento de costo mínimo con restricciones de grado (DCMST, por sus siglas en inglés). El DCMST consiste en hallar el árbol de cubrimiento de costo mínimo tal que

el grado de cada nodo sea menor que una cierta cantidad dada. El DCMST también pertenece a la clase de problemas NP-difíciles.

Bui et al. [25] afrontaron el DCMST mediante un algoritmo inspirado en ACO. La propuesta de Bui et al. distingue dos etapas, una de exploración y otra de construcción del árbol. En la etapa de exploración se coloca inicialmente una hormiga en cada nodo, que recorrerá todos los nodos de acuerdo al rastro de feromona. Las hormigas depositan feromona a medida que usan las aristas, de forma de influenciar la exploración del resto de las hormigas. En la etapa de construcción del árbol solución, se seleccionan las n (n es un párametro del algoritmo) aristas en las que se depositó más feromona y se las ordena ascendentemente por su costo. El árbol solución se construye a partir de las aristas ordenadas, mediante una versión modificada del algoritmo de Kruskal que asegura que no se violen las restricciones de grado. Si no fuera posible completar el árbol, se seleccionan las siguientes n aristas y se repite el proceso hasta completar una solución. La propuesta de Bui et al. incorpora un mecanismo de escape para evitar la convergencia a óptimos locales que consiste en evaporar parte del rastro de feromona depositado en las mejores aristas hasta que el nivel de feromona sea similar al del resto de las aristas. Los autores evaluaron su propuesta trabajando sobre siete conjuntos de grafos estándar para el DCMST (hasta 1000 nodos y restricción de grado 5) obteniendo mejoras sobre los resultados reportados con otras técnicas. Por ejemplo, sobre un conjunto de grafos aleatorios obtienen mejoras del 80 % con respecto a un GA. Sin embargo, los resultados obtenidos sobre un grupo de grafos diseñados específicamente para engañar a estrategias greedy están entre un 15 % y un 30 % de la mejor solución conocida.

Problema del árbol de cubrimiento de costo mínimo con restricciones de diámetro

El diámetro de un árbol es la máxima cantidad de aristas que componen a todos los posibles caminos del árbol. El problema del árbol de cubrimiento de costo mínimo con restricciones de diámetro (BDMST, por sus siglas en inglés) impone como condición sobre el árbol de cubrimiento que el diámetro sea menor que una cierta constante dada D. El BDMST pertenece a la clase de problemas NP-difíciles cuando D cumple que $4 \leq D \leq |V|$.

Kopinitsch et al. [91, 105] propusieron un ACO para el BDMST caracterizado porque la expresión para el cálculo de las probabilidades en la regla de transición de estados no considera un término heurístico, sino que solamente considera el valor de los rastros de feromona. Adicionalmente, el rastro de feromona está asociado a los nodos y al nivel que tendrán en el árbol. Por lo cual, la probabilidad en la regla de transición de estados resulta asociada al nivel que tendrá el nodo en el árbol, si se produce su incorporación a la solución. La propuesta de Kopinitsch et al. utiliza como búsqueda local sobre las soluciones para mejorarlas un VND. La evaluación experimental fue realizada sobre grafos (hasta 500 nodos y diámetro máximo 20) originalmente diseñados para el problema euclídeo del árbol de Steiner de la OR-Library. Kopinitsch et al. evaluaron su propuesta contra un EA y un VNS. Si se establece como criterio de parada un tiempo fijo reducido, los mejores resultados son obtenidos por VNS, seguido de ACO y EA. Sin embargo, si el tiempo establecido para la ejecución es más largo, los mejores resultados son obtenidos por ACO, seguido de VNS y EA. Considerando que el tiempo de las ejecuciones largas es inferior a los 67 minutos, es un escenario realista de ejecución para los algoritmos de resolución del BDMST.

5.4. Conclusiones

En este capítulo se presentó el Problema de Steiner Generalizado, que será utilizado para poner en práctica las ideas de la metaheurística ACO. El GSP ha sido poco abordado por la comunidad científica y dentro de los algoritmos propuestos para su resolución se destaca CHC por la calidad de las soluciones obtenidas y por su relativamente bajo costo computacional.

A partir del relevamiento realizado surgen tres motivaciones para la utilización de ACO para la resolución del GSP, cuyos detalles se presentan a continuación.

El GSP involucra la determinación de caminos óptimos entre nodos. Por otro lado, ACO se ha mostrado como una alternativa exitosa para la resolución de problemas que involucran la obtención de caminos óptimos, como el TSP. Por este motivo, ACO se considera una técnica promisoria para la resolución del GSP. Sin embargo, la optimalidad de caminos en el GSP no está asociada solamente al costo individual del camino, sino que también se vincula con la reutilización de aristas que compongan otros caminos. Por estas características, resolver el GSP requiere un balance entre aspectos locales (el costo de los caminos) y globales (la reutilización de aristas). La aplicación de ACO a la resolución del GSP debe incorporar mecanismos que aseguren considerar los aspectos globales del problema.

Los problemas de la clase de Steiner, el problema de determinación de caminos arista-disjuntos y los problemas de cubrimiento presentados en este capítulo tienen varios puntos de contacto con el GSP, existiendo recientemente importantes avances en la aplicación de ACO sobre estos tipos de problemas. Estos avances y la actual carencia de una propuesta ACO competitiva para la resolución del GSP también motivan el interés en utilizar ACO sobre el GSP.

Los enfoques previos que logran buenos resultados para el GSP se caracterizan por tener un enfoque *desconstructivo*, parten del grafo original y retiran aristas hasta obtener el grafo solución. Por otro lado, los enfoques constructivos que parten de un grafo vacío y agregan aristas hasta obtener el grafo solución, como los propuestos por Robledo y Calegari no han sido exitosos. La construcción incremental de soluciones es una característica inherente a la metaheurística ACO. Proponer un enfoque constructivo exitoso constituye el tercer motivo para la utilización de ACO para resolver el GSP.

Capítulo 6

ACO aplicado al GSP

"Y sal ahí, a defender el pan y la alegría.
Y sal ahí, para que sepan que esta boca es mía."
Joaquín Sabina, *Esta boca es mía*

Este capítulo describe las propuestas de aplicación de ACO a la resolución del GSP formuladas en este trabajo. La estructura del capítulo es la siguiente. La sección 6.1 presenta una descripción general del esquema de resolución planteado para el GSP. Posteriormente, la sección 6.2 describe en detalle la formulación de los algoritmos secuenciales, organizados según el enfoque utilizado para obtener caminos entre los terminales. La novedosa formulación de un modelo celular para la metaheurística ACO y su aplicación al GSP se presenta en la sección 6.3. Finalmente, la sección 6.4 presenta las conclusiones del capítulo.

6.1. Enfoque general de la solución

Los principios generales de los algoritmos propuestos en este trabajo se presenta a continuación. Cada hormiga construye una solución completa procesando iterativamente los requerimientos de conectividad. Cuando una hormiga procesa un par de nodos terminales correspondientes a un requisito de conectividad, toma en cuenta las aristas incluidas en la solución, tratando de reutilizarlas evitando incrementar el costo de la solución. Una vez que las hormigas culminan la etapa de construcción de soluciones, se deposita feromona sobre las aristas que componen las soluciones. Se utiliza solamente una matriz de almacenamiento global de los rastros de feromona para la resolución del problema. En el algoritmo 7 se presenta la estructura general del algoritmo ACO propuesto para la resolución del GSP.

En el algoritmo 7, PT es la matriz que almacena los rastros de feromona, s_{best} es la mejor solución encontrada por el algoritmo y s_a es la solución construida por la hormiga a. M es una matriz que almacena los requerimientos de conectividad (M(i,j) representa el número de caminos disjuntos entre los terminales i y j). E es el conjunto de aristas, E_d es el conjunto de aristas disponibles para construir caminos entre un par de terminales (pueden ya formar parte de la solución), E_u es el conjunto de aristas utilizadas en la construcción de un nuevo camino por la subrutina *buildPath* o de los caminos ya presentes en la solución para un par de terminales y E_p es el conjunto de aristas *privilegiadas*, las aristas que todavía no fueron utilizadas para la construcción de

Algoritmo 7 Esquema general del ACO propuesto para el GSP

```
s_best = E
PT = initializePheromoneTrails()
while not stopCriteria() do
   for all a ∈ A do
      s_a = empty()
      M = getRequiments()
      for all i, j ∈ T|i ≠ j do
         M(i,j) = M(i,j) - inducedDisjointPaths(s_a,i,j)
         E_u = edgesInducedDisjointPaths(s_a,i,j)
         E_d = removeEdges(E,E_u)
         E_p = removeEdges(s_a,E_u)
         while M(i,j) >0 do
            E_u = buildPath(i,j,E_d,E_p)
            E_d = removeEdges(E_d,E_u)
            M(i,j) = M(i,j) - 1
            s_a = addEdges(s_a,E_u)
         end while
      end for
      if cost(s_a) < cost(s_best) then
         s_best = s_a
      end if
   end for
   PT = applyOnlineDelayedPheromoneUpdate(PT,(s_a|a ∈ A))
end while
return s_best
```

caminos entre el par de terminales considerado, pero que como ya forman parte de la solución resulta conveniente utilizarlas.

Cada vez que una hormiga procesa un nuevo par de terminales, calcula cuántos caminos disjuntos existen entre el par de terminales en la solución parcial, mediante la subrutina *inducedDisjointPaths*. Si la cantidad de caminos es mayor o igual que la cantidad de requisitos de conexión, el requerimiento considerado ya está satisfecho en la solución; en caso contrario, se determinan las aristas utilizadas para construir los caminos disjuntos mediante la subrutina *edgesInducedDisjointPaths*. En el algoritmo 7 se presentan *inducedDisjointPaths* y *edgesInducedDisjointPaths* como subrutinas diferentes, aunque por razones de eficiencia computacional se implementaron como una única subrutina. Para determinar la cantidad de requisitos satisfechos y las aristas utilizadas se usa el algoritmo de Ford-Fulkerson [78] para encontrar caminos entre los pares de terminales, considerando uno como origen y el otro como destino. Asumiendo capacidad unitaria en cada arista, el flujo máximo entre el origen y el destino determina el número máximo de caminos disjuntos entre los nodos. Si no se han satisfecho todos los requisitos de conexión para un par de terminales, se marcan las aristas obtenidas mediante *edgesInducedDisjointPaths* como usadas (E_u) y no son consideradas para la construcción de caminos disjuntos para dicho par. Si existen aristas que no fueron usadas para el par de terminales considerados pero forman parte de la solución parcial, se marcan como *privilegiadas* (E_p) debido a que resulta conveniente su uso en la construcción de

los restantes caminos ya que no incrementan el costo de la solución.

La hormiga construye un camino entre los terminales utilizando las aristas disponibles (E_d) y *privilegiadas*, y considerando el rastro de feromona depositada y la visibilidad de las componentes mediante la subrutina *buildPath*. Se evaluaron varias alternativas para el mecanismo de construcción de los caminos, cuyos detalles se presentan en las próximas secciones. Para asegurar que los caminos construidos entre terminales sean disjuntos, las aristas utilizadas por la hormiga en la construcción de cada camino son marcadas como usadas y se eliminan de las disponibles.

Cuando una hormiga culmina la construcción de su solución, se evalúa si su costo es menor que el la mejor solución encontrada por el algoritmo. La función a optimizar es el costo total de la red construida, sumando el costo de cada una de las aristas que la componen.

Una vez que todas las hormigas finalizan la construcción de las soluciones, se realiza el depósito y evaporación de los rastros de feromona sobre las aristas del grafo.

El esquema general de resolución descrito no requiere chequeos de factibilidad, pero la ejecución del algoritmo de Ford-Fulkerson para determinar cuántos caminos existen entre un par de terminales incrementa su complejidad computacional. Sin embargo, el algoritmo propuesto explota la información presente en la solución parcial que está siendo construida, evitando resolver requerimientos que están satisfechos y minimizando la cantidad de caminos disjuntos que construye cada hormiga para cada par de terminales. Asimismo, el algoritmo explota la información disponible en la solución de mejor forma, porque considera explícitamente las aristas *privilegiadas*, que están en la solución y que no fueron usadas para el par de terminales considerado.

6.2. Algoritmos secuenciales

La estructura general del algoritmo ACO propuesto para la resolución del GSP presentada en la sección anterior deja algunos aspectos abiertos: la variante de ACO usada, la regla de transición de estados empleada, la forma de calcular la visibilidad en la regla de transición de estados, el mecanismo de actualización del rastro de feromona y el mecanismo de construcción de caminos utilizado por las hormigas. Se evaluaron varias alternativas para cada uno de estos aspectos, presentándose en esta sección las versiones secuenciales del ACO estudiadas. Las versiones secuenciales se clasifican de acuerdo al mecanismo de construcción de caminos utilizado por las hormigas.

6.2.1. Enfoque basado en aristas

El enfoque de construcción basado en aristas consiste en que la hormiga incremente iterativamente la solución mediante la incorporación de aristas hasta completar el camino. Cada hormiga debe realizar las elecciones partiendo de uno de los terminales hasta llegar al otro terminal. La construcción iterativa del camino mediante la elección de aristas es no determinística y puede fallar en la obtención de un camino entre un par de terminales (i.e.: se puede llegar a un nodo que no tenga disponibles aristas de salida). Por este motivo, se realizan un cierto número de reintentos, tratando de evitar descartar la solución que se está construyendo.

Para esta primera aproximación, el interés fundamental era mostrar la factibilidad de utilizar un ACO para la resolución del GSP, por lo que todas las versiones implementadas del enfoque de construcción basado en aristas se basaron en la variante AS (presentada

en la sección 3.2), descartándose la utilización de variantes más sofisticadas. AS utiliza la regla de transición de estados que se presenta en la ecuación 6.1 y la regla de actualización de los rastros de feromona que se presenta en las ecuaciones 6.2 y 6.3.

$$p(c_{i,x_i}|s^p) = \begin{cases} \frac{[\tau_{i,x_i}]^\alpha [\eta_{i,x_i}]^\beta}{\sum_{c_{j,x_j} \in J(s^p)} [\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta} & \text{si } c_{i,x_i} \in J(s^p) \\ 0 & \text{en otro caso} \end{cases} \tag{6.1}$$

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \sum_{a \in A} \Delta\tau_{i,x_i}^{s_a} \quad \forall \tau_{i,x_i} \in T \tag{6.2}$$

$$\text{siendo} \quad \Delta\tau_{i,x_i}^{s_a} = \begin{cases} F(s_a) & \text{si } c_{i,x_i} \in s_a \\ 0 & \text{en otro caso} \end{cases} \tag{6.3}$$

La visibilidad (η_{i,x_i}) debe considerar el costo de la arista y la contribución de la arista para obtener una solución factible. Lograr un compromiso entre ambos aspectos no es una tarea trivial. Una dificultad similar surgió en la aplicación de EA a la resolución del GSP al intentar considerar soluciones no factibles [127], ya que no es sencillo obtener una función que cuantifique la distancia al conjunto de soluciones factibles. En las ecuaciones 6.4 y 6.5 se presentan las dos funciones estudiadas para la visibilidad: la tradicional función inversa del costo (6.4) y una función alternativa que intenta privilegiar las aristas conectadas a terminales (6.5). En dichas ecuaciones, *cost* representa el costo de la arista y *incidentTerminal* cuenta la cantidad de terminales en los que la arista considerada incide (los valores posibles son 0, 1 y 2).

$$\eta_{i,x_i} = \frac{1}{cost(c_{i,x_i})} \tag{6.4}$$

$$\eta_{i,x_i} = \frac{incidentTerminal(c_{i,x_i}) + 1}{cost(c_{i,x_i})} \tag{6.5}$$

La regla de actualización del rastro de feromona requiere definir una expresión para la función de calidad de una solución ($F : S \mapsto \mathbb{R}^+$). En las ecuaciones 6.6, 6.7 y 6.8 se presentan las tres alternativas estudiadas: la tradicional (6.6), una alternativa que considera explícitamente la cantidad de requisitos que la arista contribuye a satisfacer (6.7) y una alternativa normalizada (6.8), en la que la función de calidad previa se divide entre la cantidad total de contribuciones de todas las aristas. En las expresiones de las ecuaciones, c_{i,x_i} es la arista considerada, la función *requirementsSatisfied* evalúa la cantidad de requisitos que la arista considerada colabora a satisfacer y *cost* es el costo total de la solución.

$$F(s_a) = \frac{1}{cost(s_a)} \tag{6.6}$$

$$F(s_a, c_{i,x_i}) = \frac{requirementsSatisfied(c_{i,x_i})}{cost(s_a)} \tag{6.7}$$

$$F(s_a, c_{i,x_i}) = \frac{requirementsSatisfied(c_{i,x_i})}{cost(s_a) * \sum_{c_{j,x_j}} requirementsSatisfied(c_{j,x_j})} \tag{6.8}$$

Las aristas privilegiadas deberían ser elegidas más frecuentemente que las aristas que no forman parte de la solución parcial, por lo cual se formuló la regla de transición de estados alternativa que se presenta en la ecuación 6.9. La regla de transición de estados alternativa incorpora un factor multiplicativo (γ) que incrementa la probabilidad de elegir aristas privilegiadas. En la ecuación 6.9, P es el conjunto de aristas privilegiadas. Si se considera $\gamma = 1$, no se distingue entre aristas privilegiadas y no privilegiadas, obteniéndose la regla de transición de estados tradicional (presentada en la ecuación 6.1).

$$p(c_{i,x_i}|s^p) = \begin{cases} \frac{\gamma * [\tau_{i,x_i}]^\alpha [\eta_{i,x_i}]^\beta}{\sum_{c_{j,x_j} \in J(s^p) \text{y} \in P} \gamma * [\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta + \sum_{c_{j,x_j} \in J(s^p) \text{y} \notin P} [\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta} & \text{si } c_{i,x_i} \in J(s^p) \text{ y} \in P \\ \frac{[\tau_{i,x_i}]^\alpha [\eta_{i,x_i}]^\beta}{\sum_{c_{j,x_j} \in J(s^p) \text{y} \in P} \gamma * [\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta + \sum_{c_{j,x_j} \in J(s^p) \text{y} \notin P} [\tau_{j,x_j}]^\alpha [\eta_{j,x_j}]^\beta} & \text{si } c_{i,x_i} \in J(s^p) \text{ y} \notin P \\ 0 & \text{en otro caso} \end{cases} \tag{6.9}$$

Con el objetivo de evaluar independientemente el beneficio de cada una de las ideas planteadas para la resolución del GSP, se diseñaron siete versiones de ACO siguiendo un enfoque orientado a aristas. Las características de dichas versiones se describen a continuación:

- *Puro*: está fuertemente inspirado en el enfoque tradicionalmente usado para resolver el TSP mediante AS [68]. Se utiliza la visibilidad tradicional (presentada en la ecuación 6.4), la función de calidad de la solución para el depósito de feromona tradicional (presentada en la ecuación 6.6) y la regla de transición de estados (presentada en la ecuación 6.1).

- *Term*: está diseñado modificando la visibilidad incorporando la función alternativa que incorpora la cantidad de incidencias en terminales de la arista (presentada en la ecuación 6.5). La función de calidad para el depósito de feromona y la regla de transición de estados utilizadas es idéntica a la variante *Puro*.

- *Alternativo* y *Alternativo normalizado (Alt. Norm.)*: están diseñados modificando la función de calidad utilizada para el depósito de feromona en las soluciones. La variante *Alternativo* utiliza la función alternativa que considera cuántos requerimientos ayuda a satisfacer la arista (presentada en la ecuación 6.7), mientras que la variante *Alt. Norm.* utiliza la función alternativa normalizada (presentada en la ecuación 6.8). Ambas variantes utilizan las mismas regla de transición de estados y visibilidad que la variante *Puro*.

- *Privilegiada2*, *Privilegiada5* y *Privilegiada10*: están diseñados modificando el valor de γ de la regla de transición de estados alternativa (presentada en la ecuación 6.9) de formar de privilegiar aristas presentes en la solución parcial al momento de construir los caminos. Los valores de γ considerados son 2, 5 y 10 respectivamente. Las tres variantes utilizan las mismas función de calidad y visibilidad que la variante *Puro*.

6.2.2. Enfoque basado en caminos

El enfoque basado en aristas adolece de una falta de visión completa del camino construido por una hormiga. Cuando se debe decidir qué arista incorporar al camino que está siendo construido, la hormiga solamente considera un valor heurístico y un valor histórico asociado a la arista, pero no utiliza ningún mecanismo que permita estimar la contribución de la arista para completar el camino entre el par de terminales.

El mecanismo de construcción basado en caminos pretende paliar el defecto señalado en el enfoque orientado a aristas, aplicando a la resolución del GSP una idea similar a la presentada por Luyet et al. [110] para resolver el STP (descrito en el relevamiento del capítulo 5). El mecanismo consiste en que la hormiga determine los K caminos más cortos entre el par de terminales considerado y realice la elección entre ellos. En el apéndice A se incluye una reseña del problema de los K caminos más cortos y los principales algoritmos para su resolución sin considerar restricciones sobre los caminos y considerando caminos sin ciclos. Debido a las características del GSP, el algoritmo para la construcción de los caminos debía obtener caminos sin ciclos, optándose por utilizar el algoritmo de Yen [173]. El algoritmo de Yen implementado incluye una representación de la lista de caminos considerados como un montículo por razones de eficiencia. En caso de empate en el costo del K-ésimo camino más corto, se devuelven todos los caminos que tengan el mismo costo que el K-ésimo camino más corto. El algoritmo de Yen es determinista, por lo cual si no se obtienen caminos la solución es no factible y no se requiere ningún mecanismo de reintentos, en contraposición a lo que sucede en el enfoque con aristas.

En el ACO que utiliza el enfoque basado en caminos, la visibilidad considera el costo que agrega el camino completo a la solución, sumando el costo de cada una de las aristas que la componen y restando el costo de las aristas *privilegiadas* que ya pertenecen a la solución parcial. En las ecuaciones 6.10 y 6.11 se presenta la función para la visibilidad, siendo p_i el camino considerado, *cost* el costo del camino o la arista y P el conjunto de aristas *privilegiadas*.

$$\eta_{p_i} = \frac{1}{cost(p_i)} \tag{6.10}$$

$$cost(p_i) = \sum_{c_{i,x_i} \in p_i} cost(c_{i,x_i}) - \sum_{c_{i,x_i} \in P} cost(c_{i,x_i}) \tag{6.11}$$

El valor asociado a la experiencia adquirida en la búsqueda se calcula como el promedio de los rastros de feromona en las aristas que no forman parte de la solución, es decir aquellas que pertenecen al camino y que no pertenecen al conjunto de aristas *privilegiadas*. En la ecuación 6.12 se presenta la función para el cálculo de la experiencia adquirida, siendo p_i el camino considerado y P el conjunto de aristas *privilegiadas*.

$$\tau_{p_i} = \frac{\sum_{c_{i,x_i} \in p_i \text{ y } c_{i,x_i} \notin P} \tau_{c_{i,x_i}}}{|c_{i,x_i} \in p_i \text{ y } c_{i,x_i} \notin P|} \tag{6.12}$$

Se diseñaron dos versiones de ACO con un enfoque orientado a caminos, utilizando las variantes AS e Hyper-Cube Framework con el mecanismo de actualización de feromona de la variante $\mathcal{MM}$AS (HCF-$\mathcal{MM}$AS, presentada en la sección 3.6). HCF-$\mathcal{MM}$AS utiliza la regla de actualización de los rastros de feromona que se presenta en las ecuaciones 6.13 y 6.14, siendo s_{best} la mejor solución de la iteración.

$$\tau_{i,x_i} \leftarrow (1-\rho) * \tau_{i,x_i} + \rho \Delta\tau_{i,x_i}^{s_{best}} \quad \forall \tau_{i,x_i} \in T \tag{6.13}$$

$$\Delta\tau_{i,x_i}^{s_{best}} = \begin{cases} 1 & \text{si } c_{i,x_i} \in s_{best} \\ 0 & \text{en otro caso} \end{cases} \tag{6.14}$$

A continuación se describen las características de las dos versiones implementadas:

- *Caminos*: es una implementación tradicional de AS utilizando el enfoque de construcción basado en caminos entre pares de terminales. La función de calidad de una solución utilizada por la regla de actualización del rastro de feromona es la tradicional (presentada en la ecuación 6.6).
- *HCF-$\mathcal{MM}AS$*: es una implementación del enfoque de construcción basado en caminos entre pares de terminales utilizando la variante HCF-$\mathcal{MM}$AS.

Búsquedas locales

Al utilizar una aproximación constructiva al GSP se busca obtener un grafo solución que no contenga un subgrafo que también sea solución al problema, evitando la incorporación de aristas superfluas que generen más caminos de los necesarios entre pares de terminales. El enfoque de construcción basado en caminos también persigue ese objetivo, aunque el grafo que se obtiene al aplicar el mecanismo descrito en esta sección puede contener un subgrafo que sea solución al GSP. Para paliar esta falencia se planteó la utilización de una búsqueda local que permita mejorar la solución construida. La búsqueda local consiste en aplicar el mecanismo de construcción partiendo de la solución que se quiere mejorar y no del grafo original.

Para ejemplificar la utilidad del mecanismo de búsqueda local considérese el grafo original de la figura 6.1 y los requisitos de conexión: $req(c, a) = 2$ y $req(a, b) = 2$. Un posible grafo solución construido por una hormiga se muestra en la figura 6.2. Sin embargo, la solución obtenida contiene un subgrafo que cumple los requisitos de conexión impuestos, presentado en la figura 6.3. La utilización de la búsqueda local planteada permite concentrarse en un conjunto potencialmente distinto de caminos a los considerados originalmente, restringiendo de esta forma la búsqueda y permitiendo encontrar subgrafos que estén contenidos en la solución. La solución obtenida por la aplicación de la búsqueda local puede contener un subgrafo solución del problema, por lo que es posible aplicar iterativamente la búsqueda local sobre las soluciones obtenidas hasta que no se produzcan mejoras.

A partir de las ideas planteadas en esta subsección, se diseñaron dos versiones del mecanismo de búsqueda local, que se describen a continuación:

- *BL*: se aplica el enfoque de construcción basado en caminos sobre el grafo solución construido por la hormiga.
- *BL iterada*: se aplica iterativamente el enfoque de construcción basado en caminos sobre el grafo solución construido por la hormiga mientras que se produzcan mejoras en la solución obtenida.

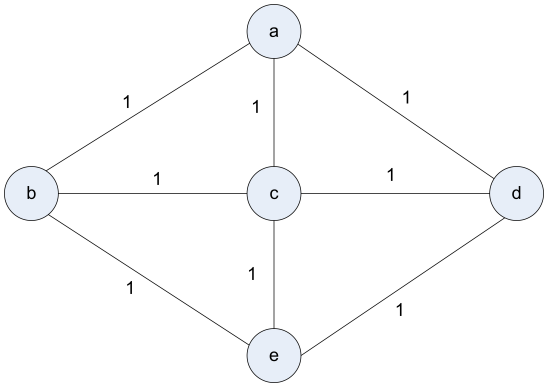

Figura 6.1: Grafo original

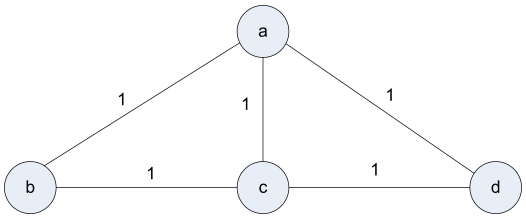

Figura 6.2: Posible grafo solución obtenido por una hormiga siguiendo el enfoque de construcción de caminos

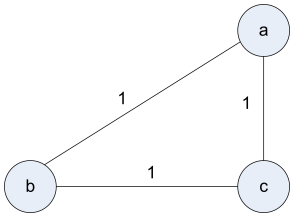

Figura 6.3: Subgrafo solución contenido en el grafo solución

6.3. Modelo celular

En el modelo celular cada hormiga se ubica en una celda de una grilla rectangular de dos dimensiones y utiliza su propia matriz de feromona para la construcción de soluciones. Se define una vecindad alrededor de cada hormiga para realizar la actualización de la matriz de rastros de feromona asociada a la hormiga. La solución utilizada para actualizar los rastros es la mejor solución encontrada por una hormiga de la vecindad. Las vecindades tienen solapamientos para que el efecto de encontrar una mejor solución pueda propagarse paulatinamente al resto de las vecindades. En este caso, a diferencia del modelo celular de EA, el efecto de propagación es indirecto, ya que la mejor solución del vecindario afecta la matriz de feromona y no directamente a otras soluciones. Los restantes aspectos del algoritmo no presentan diferencias con la versión *HCF-$\mathcal{MM}AS$*.

Por tratarse de la primera aproximación a un modelo celular para ACO se optó por seguir un enfoque simple, considerando que las actualizaciones sobre las matrices de feromona de cada una de las celdas se producen en forma sincrónica y paralela, es decir que todas las hormigas realizan la actualización a la vez. Se utilizaron estructuras de

vecindario tradicionales: el de Von Neumann y el de Moore. En la figura 6.4 se presenta la vecindad de Von Neumann, que se compone de cinco celdas: la hormiga se ubica en la celda central y sus adyacentes ubicadas arriba, abajo, a la izquierda y a la derecha en la grilla. En la figura 6.5 se presenta la vecindad de Moore, que se compone de nueve celdas: las cinco de la vecindad de Von Neumann y las cuatro adyacentes siguiendo las diagonales.

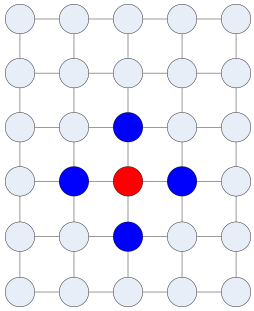

Figura 6.4: Vecindad de Von Neumann

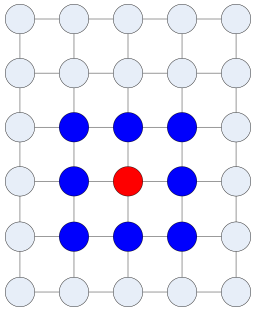

Figura 6.5: Vecindad de Mooore

Se diseñaron dos versiones de ACO siguiendo un modelo celular, cuyas características se describen a continuación:

- *Von*: es una implementación del modelo celular de ACO tomando como base las características de la versión *HCF-$\mathcal{MMAS}$* y utilizando una estructura de vecindario de Von Neumann.
- *Moore*: es una implementación del modelo celular de ACO tomando como base las características de la versión *HCF-$\mathcal{MMAS}$* y utilizando una estructura de vecindario de Moore.

La implementación paralelo-distribuida del modelo celular de ACO no presenta dificultades. Cada proceso tiene asignado un grupo de celdas contiguas de la grilla original. En cada iteración, cada proceso debe construir las soluciones respectivas a partir de las matrices de feromona correspondientes a cada celda. Cuando el proceso terminó de construir todas las soluciones, debe enviar las que están en celdas del borde de su sección de grilla (las soluciones que están en las celdas de la primera y última fila, y de la primera y última columna) a los procesos vecinos. Después de recibir las soluciones de sus vecinos, cada proceso actualiza el rastro de feromona de cada una de las celdas.

6.4. Conclusiones

En este capítulo se presentaron las ideas planteadas para la aplicación de ACO a la resolución del GSP. Los algoritmos tienen una estructura general que se caracteriza por construir las soluciones procesando iterativamente los requerimientos. Se intenta evitar la resolución de requisitos de conexión que la solución parcial ya satisface, fomentando la reutilización de las aristas que la componen. Se utiliza una única matriz de feromona global a todo el problema, realizando los depósitos de feromona sobre las aristas que componen la solución. Los algoritmos planteados evitan la realización de chequeos de factibilidad.

Se formularon dos mecanismos distintos para la construcción de los caminos entre los terminales por parte de las hormigas. El primer mecanismo consiste en la incorporación de aristas hasta completar el camino, formulándose siete versiones que difieren en la visibilidad, la función de calidad de la solución para el depósito de feromona y la regla de transición de estados. El segundo mecanismo consiste en la determinación de los K caminos más cortos entre el par de terminales y la elección de uno de ellos, formulándose dos versiones que difieren en la variante de ACO utilizada.

Se diseñó un mecanismo de búsqueda local que explota la idea de aplicar el procedimiento constructivo sobre el grafo solución, restringiendo la búsqueda a un conjunto potencialmente distinto de caminos.

Se formuló un modelo celular para ACO caracterizado por ubicar en cada celda una hormiga con su propia matriz de feromona. La actualización de los rastros de feromona de cada matriz se realiza en forma sincrónica a partir de las mejores soluciones construidas por las hormigas de la vecindad de la celda. Se formularon dos variantes del ACO celular que difieren en las vecindades utilizadas. Asimismo, se brindaron las pautas para la implementación paralela del modelo celular propuesto.

Capítulo 7

Evaluación experimental

"La experiencia es la evidencia"
Héroes del Silencio, *¡Rueda, fortuna!*

Este capítulo presenta la evaluación experimental de las propuestas de aplicación de ACO a la resolución del GSP formuladas en este trabajo. La estructura del capítulo es la siguiente. La sección 7.1 comenta las instancias de calibración y de evaluación utilizadas en las pruebas, detallando sus principales características y las mejores soluciones conocidas para cada instancia. Posteriormente, la sección 7.2 presenta las pruebas realizadas sobre los algoritmos secuenciales propuestos, incluyendo la calibración y la evaluación de los enfoques basados en aristas y en caminos. La evaluación del enfoque basado en caminos incorpora el estudio del efecto del tamaño de la población, de la cantidad de caminos y de incorporar operadores de búsqueda local. La evaluación del modelo celular propuesto para la metaheurística ACO y su implementación paralela se comenta en la sección 7.3. Finalmente, la sección 7.4 presenta las conclusiones del capítulo.

7.1. Instancias

El GSP ha sido poco abordado por la comunidad académica, no existiendo instancias de pruebas que sean ampliamente reconocidas. En trabajos previos, Robledo [144] y Nesmachnow et al. [127, 128, 129] abordaron parcialmente el diseño de instancias para el GSP. En esta sección se presentan algunas de las instancias diseñadas en esos trabajos separadas en dos conjuntos, uno para la calibración y otro para la evaluación de las propuestas de ACO implementadas.

7.1.1. Calibración

En su tesis de maestría, Robledo [144] diseñó instancias de hasta 20 vértices, 40 aristas y 10 terminales para el GSP. Sin embargo, las instancias eran de escasa complejidad y no representaron una dificultad real para algunos enfoques utilizados para resolver el GSP [127]. Nesmachnow [128] generó replicaciones de algunos de los grafos propuestos originalmente por Robledo, obteniendo instancias de mayor tamaño y con óptimo conocido, aunque formadas por grafos desconexos. El mecanismo de generación de instancias es sumamente útil, ya que los algoritmos propuestos por Nesmachnow et al. [129] no explotan explícitamente particularidades de la topología de la instancia utilizada.

En la tabla 7.1 se presenta el conjunto de instancias de calibración. La numeración de los grafos coincide con la numeración seguida por Nesmachnow, siendo la cantidad de replicaciones 10 para el Grafo 1, 13 para el Grafo 2, 10 para el Grafo 3 y 10 para el Grafo 4. Los grafos utilizados para la replicación son un hexaedro completo para el Grafo 1, un decaedro con alta conectividad para el Grafo 2 (la relación entre la cantidad de aristas del grafo y la cantidad de aristas si el grafo fuera completo es mayor que 0.82) y topologías irregulares y con baja conectividad para el Grafo 3 y 4 (la relación entre la cantidad de aristas del grafo y la cantidad de arista si el grafo fuera completo es menor que 0.25). En la tabla 7.1 se presenta la cantidad de vértices, terminales y aristas de cada uno de los grafos de calibración, así como sus costos totales y óptimos y la cantidad total de requisitos de conectividad.

Tabla 7.1: Datos del conjunto de instancias de calibración

	Grafo 1	**Grafo 2**	**Grafo 3**	**Grafo 4**
Vértices	60	130	160	150
Terminales	40	26	60	50
Aristas	150	481	250	260
Requisitos	160	52	360	180
Costo total	350	624	490	710
Costo óptimo	180	247	390	360

En la tabla 7.2 se presentan los resultados obtenidos por Nesmachnow [128] sobre las instancias del conjunto de calibración con los algoritmos: SA, VNS, GA, GA+VNS, GA+SA y CHC (presentados en la sección 5.2). El criterio de parada utilizado fue de esfuerzo prefijado en 2000 generaciones para los algoritmos evolutivos y de 10000 iteraciones para el SA y el VNS. Los algoritmos poblacionales utilizaron 120 individuos inicializados mediante la eliminación de un 5 % de aristas en forma aleatoria. Los resultados que se presentan en la tabla 7.2 son el mejor costo obtenido y el costo promedio de 10 ejecuciones independientes. En el trabajo de Nesmachnow no se reporta el tiempo de ejecución promedio para las instancias del conjunto de calibración considerado. Sin embargo, a partir de los tiempos de ejecución reportados para instancias con una mayor cantidad de replicaciones, es posible afirmar que los tiempos de ejecución promedios son menores a 15 minutos para el CHC, a 22 minutos para el GA, a 5 minutos para el SA, a 9 minutos para el VNS, a 36 minutos para el GA+VNS y a 29 minutos para el GA+SA sobre una computadora Intel Pentium 4 de 2.4 Ghz con 512 Mb RAM y con sistema operativo SuSE Linux 8.1.

7.1.2. Evaluación

Nesmachnow et al. [11, 127, 129] diseñaron un generador de instancias aleatorias para el GSP. El generador selecciona aleatoriamente puntos en el plano euclídeo para representar a los vértices e incluye las aristas con probabilidad inversamente proporcional a la distancia euclidiana entre los vértices. El costo de las aristas es proporcional a la distancia euclidiana entre los nodos que conecta. La cantidad de requerimientos para cada par de terminales se selecciona en forma aleatoria entre 0 y 4. El generador de instancias se encuentra disponible públicamente [4].

En la tabla 7.3 se presenta el conjunto de instancias de evaluación compuesto por

Tabla 7.2: Resultados obtenidos por Nesmachnow [128] sobre las instancias de calibración

		Grafo 1	**Grafo 2**	**Grafo 3**	**Grafo 4**
SA	*Promedio*	201.4	293.7	402.6	406.9
	Mejor	200	291	402	401
VNS	*Promedio*	198.9	290.3	401.1	398.9
	Mejor	194	287	398	389
GA	*Promedio*	191.7	275.3	396.9	383.5
	Mejor	191	274	396	378
GA+VNS	*Promedio*	188.5	272.3	395.0	377.5
	Mejor	187	268	394	372
GA+SA	*Promedio*	192.1	274.1	397.1	383.5
	Mejor	191	272	396	383
CHC	*Promedio*	188.8	277.5	394.2	395.0
	Mejor	183	269	391	382

tres grafos representativos de redes de tamaño mediano. El nombre de la instancia indica la cantidad de vértices y terminales del grafo. Las instancias fueron generadas con el mecanismo previamente descripto, excepto por los costos de las aristas del *Grafo 100-10* que fueron seleccionados aleatoriamente entre 0 y 20. En la tabla 7.3 se presenta la cantidad de vértices, terminales y aristas de cada uno de los grafos de evaluación, así como su costo total y mejor costo conocido, la cantidad total de requisitos de conectividad y la relación entre la cantidad de aristas del grafo y la cantidad de aristas si el grafo fuera completo (se presenta en la tabla como conectividad promedio). Las instancias del conjunto de evaluación se encuentran públicamente disponibles [4]. La instancia disponible públicamente del *Grafo 75-25* tiene solamente 20 terminales (no se modificó el nombre de la instancia para mantener la coherencias con los trabajos previos).

Tabla 7.3: Datos del conjunto de instancias de evaluación

	Grafo 100-10	**Grafo 75-25**	**Grafo 50-15**
Vértices	100	75	50
Terminales	10	20	15
Aristas	500	360	249
Requisitos	138	362	214
Conectividad promedio	0.1	0.13	0.2
Costo total	4925	6294.93	10949.98
Mejor costo conocido	291	773.09	1353.49

En la tabla 7.4 se presentan los resultados obtenidos por Nesmachnow et al. [129] sobre las instancias del conjunto de evaluación con los algoritmos: GA+SA1, GA+SA2, CHC1, CHC2, GA y SA (presentados en la sección 5.2). El criterio de parada utilizado fue de esfuerzo prefijado en 2000 generaciones para los algoritmos poblacionales y de 10000 iteraciones para el SA. Los algoritmos poblacionales utilizaron 120 individuos inicializados mediante la eliminación de un 5 % de aristas en forma aleatoria. Los resultados que se presentan en la tabla 7.4 son el costo promedio, la desviación estándar, el

mejor costo y el tiempo promedio de 30 ejecuciones independientes. La plataforma de ejecución utilizada fue una computadora Intel Pentium 4 de 2.4 Ghz con 512 Mb RAM y con sistema operativo SuSE Linux 8.1.

Tabla 7.4: Resultados de Nesmachnow et al. [129] sobre las instancias de evaluación

		Grafo 100-10	**Grafo 75-25**	**Grafo 50-15**
GA+SA1	*Promedio*	433.5	931.04	1576.66
	Desv. Est.	36.7	39.8	70.7
	Mejor	376	865.59	1475.72
	Tiempo promedio	520.0	1229.4	462.6
GA+SA2	*Promedio*	453.5	959.53	1619.37
	Desv. Est.	30.2	49.2	85.3
	Mejor	418	867.87	1411.64
	Tiempo promedio	139.2	319.8	126.0
CHC1	*Promedio*	476.0	2637.91	3303.67
	Desv. Est.	42.6	372.6	726.2
	Mejor	358	1720.03	1807.02
	Tiempo promedio	52.6	31.2	11.4
CHC2	*Promedio*	323.0	815.98	1435.82
	Desv. Est.	15.1	21.2	43.2
	Mejor	291	773.09	1353.49
	Tiempo promedio	59.4	275.4	51.0
GA	*Promedio*	455.5	986.71	1633.16
	Desv. Est.	28.2	50.2	85.8
	Mejor	394	888.27	1493.95
	Tiempo promedio	122.8	310.2	125.4
SA	*Promedio*	739.0	1123.16	1772.81
	Desv. Est.	48.6	51.9	114.6
	Mejor	623	1045.95	1628.10
	Tiempo promedio	18.6	31.8	9.0

CHC2 obtiene sistemáticamente mejores resultados en todas las instancias, utilizando un menor tiempo de ejecución que los algoritmos que logran alcanzar resultados aceptables. El algoritmo híbrido GA+SA1 presenta resultados inferiores a CHC2 pero levemente superiores a los obtenidos por GA y GA+SA2. Sin embargo, el tiempo de ejecución de GA+SA1 es más del triple que el utilizado por GA y GA+SA2. SA y CHC1 presentan resultados muy pobres con respecto a los otros algoritmos.

7.2. Pruebas de los algoritmos secuenciales

En esta sección se presentan los experimentos realizados con los algoritmos secuenciales propuestos en este trabajo. Primeramente se realizaron pruebas de calibración para poder definir la mejor configuración de parámetros de las variantes propuestas. A partir de las configuraciones determinadas en la calibración, se realizó la evaluación de las variantes sobre los grafos medianos. La etapa de evaluación permitió determinar la variante más promisoria para profundizar su análisis, estudiando el efecto del tamaño

de la población y la incorporación de operadores de búsqueda local sobre la calidad de las soluciones y el tiempo de ejecución.

Las versiones de ACO con un enfoque basado en aristas consideradas son: *Puro*, *Term*, *Alternativo*, *Alt. Norm.*, *Privilegiada2*, *Privilegiada5* y *Privilegiada10* (presentadas en la sección 6.2.1). Las versiones de ACO con un enfoque basado en caminos consideradas son: *Caminos* y *HCF-$\mathcal{MMAS}$* (presentadas en la sección 6.2.2). Todas las versiones fueron implementadas en el lenguaje C++.

Para las pruebas de algoritmos secuenciales se utilizaron tres plataformas de ejecución distintas que se detallan a continuación:

1. computadoras Intel Pentium 4 de 2.4 Ghz con 512 MB RAM y con sistema operativo SuSE Linux 8.1.

2. computadoras AMD Athlon 64 Processor 3000+ de 2.0 Ghz con 1024 MB RAM y con sistema operativo SuSE Linux 10.0.

3. computadoras no dedicadas Pentium 4 de 2.4 Ghz con 512 Mb RAM con sistema operativo SuSE Linux 8.1 y AMD Athlon 64 Processor 3000+ de 2.0 Ghz con 1024 MB RAM y con sistema operativo SuSE Linux 10.0.

7.2.1. Calibración

Las instancias de pruebas usadas en la etapa de calibración fueron generadas mediante la replicación de grafos y tienen una complejidad mediana y costo óptimo conocido. Los algoritmos constructivos, como la metaheurística ACO, explotarán la desconexión existente entre las replicas al aplicarse sobre estas instancias. Por este motivo, la comparación entre las variantes ACO propuestas en este trabajo y las planteadas por Nesmachnow et al. sobre estas instancias puede resultar injusto. Sin embargo, cada hormiga debe construir una mayor cantidad de caminos que en los grafos originales, constituyendo un incremento real en la complejidad del problema a resolver.

Enfoque basado en aristas

El enfoque basado en aristas tiene como parámetros la influencia relativa del rastro de feromona (α), la influencia relativa del componente heurístico (β), la tasa de evaporación del rastro de feromona (ρ), el tamaño de la población de hormigas m, la cantidad inicial del rastro de feromona en las componentes (τ_{init}) y la cantidad de feromona depositada Q. La cantidad de configuraciones posibles es muy alta debido a la gran cantidad de parámetros. El criterio adoptado fue reducir la cantidad de parámetros a tres, de forma de volver manejable la cantidad de configuraciones.

Trabajos previos permiten concluir que los parámetros Q y τ_{init} suelen ser poco sensibles a las particularidades del problema. Se ha comprobado que el valor del parámetro Q no modifica significativamente los resultados, por lo cual se lo suele considerar como uno [67, 68]. Por otro lado, la experiencia indica que conviene inicializar el rastro de feromona en una cantidad similar a la que será depositada en una iteración por la colonia de hormigas [68]. Por lo cual, se suele considerar τ_{init} como $\frac{m}{Costo(SolucionAuxiliar)}$. En este caso se consideró como $Costo(SolucionAuxiliar)$ el costo de la mejor solución conocida.

Nesmachnow [128] trabajó con algoritmos poblacionales de 120 individuos sobre las mismas instancias, considerándose apropiado utilizar una población de hormigas del mismo tamaño.

Para los parámetros restantes se consideraron como valores $\alpha, \beta \in \{0.5, 1, 2, 5\}$ y $\rho \in \{0.1, 0.3, 0.5, 0.7\}$ a partir de los valores utilizados por Robledo [144] y por Dorigo et al. [67]. Las configuraciones para cada una de las siete variantes del enfoque basado en aristas eran 64, limitando significativamente la cantidad de ejecuciones realizables. Para cada configuración se realizaron solamente cinco ejecuciones independientes, utilizándose un criterio de parada de esfuerzo prefijado de 200 iteraciones. Debido a la gran cantidad de ejecuciones que se necesitaba en esta etapa, se utilizó la plataforma de ejecución 3 que es heterogénea y no dedicada, no reportándose los tiempos de ejecución.

No se realizaron ejecuciones para la familia de versiones *Privilegiada* sobre la instancia Grafo 2 debido a que solamente tiene dos terminales, por lo cual nunca se resuelven requisitos a partir de una solución parcial. Los resultados que se obtendrían para esa instancia deben coincidir con los resultados obtenidos por la versión *Puro*.

En la tabla 7.5 se presentan las configuraciones que obtienen el mejor costo promedio para cada una de las versiones del enfoque basado en aristas sobre cada una de las instancias del conjunto de calibración. Para cada una de las versiones se aprecia una gran variabilidad en las configuraciones que obtienen el mejor promedio para cada instancia.

Tabla 7.5: Configuraciones con el mejor costo promedio del enfoque basado en aristas

Variante		**Grafo 1**	**Grafo 2**	**Grafo 3**	**Grafo 4**
Puro	(α,β,ρ)	(5, 1, 0.5)	(5, 0.5, 0.5)	(1, 5, 0.7)	empate
	Promedio	201.0	286.6	427.8	381.8
Term	(α,β,ρ)	(5, 1, 0.3)	(5, 0.5, 0.3)	(0.5, 5, 0.1)	(5, 0.5, 0.7)
	Promedio	199.6	281.6	437.8	377.6
Alternativo	(α,β,ρ)	(2, 2, 0.3)	(5, 0.5, 0.5)	(1, 5, 0.1)	(5, 0.5, 0.5)
	Promedio	202.0	286.0	437.2	387.6
Alt. Norm.	(α,β,ρ)	(5, 2, 0.1)	(1, 0.5, 0.1)	(1, 2, 0.1)	empate
	Promedio	210.0	350.6	443.2	394.8
Privilegiada2	(α,β,ρ)	(5, 1, 0.3)	N/A	(1, 5, 0.1)	(5, 2, 0.5)
	Promedio	201.6	N/A	429.6	380.2
Privilegiada5	(α,β,ρ)	(2, 1, 0.5)	N/A	(0.5, 2, 0.3)	(5, 0.5, 0.5)
	Promedio	201.6	N/A	433.0	377.6
Privilegiada10	(α,β,ρ)	(2, 2, 0.5)	N/A	(0.5, 2, 0.1)	(5, 0.5, 0.7)
	Promedio	202.4	N/A	431.0	375.8

En la tabla 7.6 se presentan las configuraciones que obtienen la mejor solución para cada una de las versiones del enfoque basado en aristas sobre cada una de las instancias del conjunto de calibración. Se observa que se mantiene una gran variabilidad de las configuraciones para cada una de las versiones en cada instancia.

En la tabla 7.7 se presentan las mejores configuraciones para cada una de las versiones del enfoque basado en aristas sobre cada una de las instancias del conjunto de calibración. La mejor configuración para cada versión fue elegida utilizando como criterio la minimización del promedio del porcentaje de distancia al óptimo de cada instancia.

Term presenta los mejores resultados pero solamente es competitivo con los algorit-

Tabla 7.6: Configuraciones con la mejor solución para el enfoque basado en aristas

Variante		**Grafo 1**	**Grafo 2**	**Grafo 3**	**Grafo 4**
Puro	**(α,β,ρ)**	(5, 0.5, 0.5)	(5, 0.5, 0.5)	(1, 5, 0.7)	(5, 1, 0.1)
	Mejor	198	279	421	379
Term	**(α,β,ρ)**	(5, 1, 0.3)	(5, 1, 0.5)	(5, 5, 0.3)	(5, 2, 0.3)
	Mejor	196	270	432	374
Alternativo	**(α,β,ρ)**	(5, 2, 0.1)	(5, 0.5, 0.5)	(0.5, 5, 0.3)	(2, 2, 0.3)
	Mejor	198	273	430	383
Alt. Norm.	**(α,β,ρ)**	(5, 5, 0.5)	(0.5, 0.5, 0.5)	(2, 2, 0.7)	(0.5, 1, 0.1)
	Mejor	206	346	436	390
Privilegiada2	**(α,β,ρ)**	empate	N/A	(1, 0.5, 0.3)	(5, 0.5, 0.3)
	Mejor	200	N/A	420	377
Privilegiada5	**(α,β,ρ)**	(1, 2, 0.1)	N/A	(0.5, 2, 0.1)	(5, 0.5, 0.5)
	Mejor	199	N/A	428	374
Privilegiada10	**(α,β,ρ)**	(2, 1, 0.3)	N/A	(0.5, 2, 0.1)	(5, 0.5, 0.7)
	Mejor	200	N/A	427	373

Tabla 7.7: Resultados de la mejor configuración para el enfoque basado en aristas

Variante	**(α,β,ρ)**		**Grafo 1**	**Grafo 2**	**Grafo 3**	**Grafo 4**
Puro	(5, 0.5, 0.1)	*Prom.*	202.2	288.6	461.8	385.6
		Mejor	201	282	452	382
Term	(5, 1, 0.5)	*Prom.*	200.0	281.8	452.8	380.6
		Mejor	199	270	451	378
Alternativo	(5, 0.5, 0.5)	*Prom.*	220.0	286.0	460.0	387.6
		Mejor	220	273	460	385
Alt. Norm.	(1, 0.5, 0.1)	*Prom.*	218.8	350.6	472.6	396.2
		Mejor	217	348	470	390
Privilegiada2	(5, 2, 0.1)	*Prom.*	202.8	N/A	437.6	383.2
		Mejor	200	N/A	435	381
Privilegiada5	(5, 2, 0.1)	*Prom.*	203.4	N/A	444.4	382.8
		Mejor	202	N/A	443	381
Privilegiada10	(5, 1, 0.7)	*Prom.*	204.8	N/A	447.0	382.2
		Mejor	202	N/A	432	378

mos de Nesmachnow para el Grafo 4. Los resultados obtenidos por *Term* están aproximadamente al 11 % del óptimo del Grafo 1, al 14 % del óptimo del Grafo 2, al 16 % del óptimo del Grafo 3 y al 6 % del óptimo del Grafo 4. Las versiones *Puro* y *Privilegiada2* presentan resultados de calidad similar a los obtenidos por *Term*.

En lo que concierne a la utilización de mecanismos alternativos para determinar la cantidad de feromona a depositar, *Alternativa* presenta resultados de calidad similar a *Puro* excepto para el Grafo 1. Por otro lado, *Alt. Norm.* presenta resultados de muy baja calidad, siendo la versión que obtiene los peores resultados en los cuatro grafos. Mientras que para la familia de versiones que privilegian la reutilización de componentes, en general los mejores resultados se obtienen cuando γ es 2, degradándose la calidad de

los resultados al utilizar valores mayores.

El Grafo 3 tiene como particularidad que los algoritmos obtienen los resultados más lejanos al óptimo y que es el único en que los mejores resultados los obtiene una versión de la familia *Privilegiada*.

Para varias instancias se observa que valores de $\alpha > 1$ obtienen los mejores resultados. Sin embargo, la experiencia que existe sobre la utilización de ACO sobre otros problemas indica que el valor recomendable para α es uno [68]. La explicación detrás de esta diferencia puede atribuirse a que el enfoque del algoritmo ACO propuesto es poco convencional, ya que las componentes se utilizan varias veces en la construcción de la solución. Ese no es el caso típico de los problemas abordados por otros autores, como por ejemplo el TSP y el QAP.

Enfoque basado en caminos

El enfoque basado en caminos tiene como parámetros la influencia relativa del rastro de feromona (α), la influencia relativa del componente heurístico (β), la tasa de evaporación del rastro de feromona (ρ), el tamaño de la población de hormigas m, la cantidad inicial del rastro de feromona en las componentes (τ_{init}), la cantidad de feromona depositada Q y la cantidad de caminos considerados para satisfacer un requerimiento de conexión. Como la cantidad de configuraciones puede ser muy alta debido a la gran cantidad de parámetros, se adoptó como criterio reducir al máximo la cantidad de parámetros estudiados.

La cantidad de caminos considerados para satisfacer un requerimiento de conexión se fijó en 10, estudiándose en mayor profundidad el efecto de este parámetro en una etapa posterior.

Para la determinación de los parámetros a calibrar para la versión *Caminos* se siguieron las mismas pautas que para el enfoque basado en aristas, utilizando los mismos valores para m, τ_{init} y Q. Por consiguiente los parámetros calibrados fueron $\alpha, \beta \in \{0.5, 1, 2, 5\}$ y $\rho \in \{0.1, 0.3, 0.5, 0.7\}$.

Para la versión *HCF-$\mathcal{MM}AS$* se utilizó para m el mismo valor que para el enfoque basado en aristas. La variante HCF-$\mathcal{MM}$AS de ACO posee la particularidad de que tiene definida la cantidad de feromona depositada (como se detalla en la sección 3.6), por lo cual no se debe calibrar ese parámetro. El valor de los rastros de feromona está acotado a $[0, 1]$ por lo que se recomienda considerar τ_{init} como 0.5 [20]. La estrategia de depósito de feromona es fuertemente elitista ya que solamente una hormiga por iteración puede depositar feromona, por este motivo se suelen usar tasas de evaporación bajas [20, 68]. Para mantener uniformidad entre los valores utilizados para los parámetros entre los distintos enfoques, se utilizó 0.1 como valor de ρ. Por consiguiente los parámetros calibrados fueron $\alpha, \beta \in \{0.5, 1, 2, 5\}$.

De este modo resultaron 64 configuraciones para *Caminos* y 16 para *HCF-$\mathcal{MM}AS$*, realizándose solamente 5 ejecuciones independientes para cada configuración. Para las ejecuciones se utilizó un criterio de parada de esfuerzo prefijado de 200 iteraciones. Debido a la gran cantidad de ejecuciones que se necesitaba en esta etapa, se utilizó la plataforma de ejecución 3 que es heterogénea y no dedicada, no reportándose los tiempos de ejecución.

En la tabla 7.8 se presentan las configuraciones que obtienen el mejor costo promedio para cada una de las versiones del enfoque basado en caminos sobre cada una de las instancias del conjunto de calibración. Para ambas variantes se aprecia que el mejor

promedio para cada instancia se obtiene con la misma configuración (inclusive en los casos de empate).

Tabla 7.8: Configuraciones con el mejor costo promedio del enfoque basado en caminos

Variante		**Grafo 1**	**Grafo 2**	**Grafo 3**	**Grafo 4**
Caminos	**(α,β,ρ)**	(5, 5, 0.5)	empate	(5, 5, 0.5)	(5, 5, 0.5)
	Promedio	180.8	247.0	406.6	377.8
HCF-$\mathcal{MM}$AS	**(α,β,ρ)**	(5, 5, 0.1)	empate	(5, 5, 0.1)	(5, 5, 0.1)
	Promedio	180.0	247.0	400.0	370.0

En la tabla 7.9 se presentan las configuraciones que obtienen la mejor solución para cada una de las versiones del enfoque basado en caminos sobre cada una de las instancias del conjunto de calibración. Se observa que las mejores soluciones se obtienen consistentemente con la misma configuración para cada instancia de cada una de las variantes (inclusive en los casos de empate).

Tabla 7.9: Configuraciones con la mejor solución para el enfoque basado en caminos

Variante		**Grafo 1**	**Grafo 2**	**Grafo 3**	**Grafo 4**
Caminos	**(α,β,ρ)**	empate	empate	empate	(5, 5, 0.5)
	Mejor	180	247	405	375
HCF-$\mathcal{MM}$AS	**(α,β,ρ)**	empate	empate	(5, 5, 0.1)	(5, 5, 0.1)
	Mejor	180	247	397	367

En la tabla 7.10 se presentan las mejores configuraciones para cada una de las variantes del enfoque basado en caminos sobre cada una de las instancias del conjunto de calibración. Las mejores configuraciones para cada variante fueron elegidas utilizando como criterio la minimización del promedio del porcentaje de distancia al óptimo de cada instancia.

Tabla 7.10: Resultados de la mejor configuración para el enfoque basado en caminos

Variante	**(α,β,ρ)**		**Grafo 1**	**Grafo 2**	**Grafo 3**	**Grafo 4**
Caminos	(5, 5, 0.5)	*Prom.*	180.8	247.0	406.6	377.8
		Mejor	180	247	405	375
HCF-$\mathcal{MM}$AS	(5, 5, 0.1)	*Prom.*	180.0	247.0	400.0	370.0
		Mejor	180	247	397	367

Ambas versiones presentan muy buenos resultados siendo superiores a los obtenidos por Nesmachnow excepto para el Grafo 3. *HCF-$\mathcal{MM}$AS* presenta resultados levemente superiores a *Caminos*, alcanzando el óptimo para el Grafo 1 y 2 y estando aproximadamente al 2.5 % del óptimo para el Grafo 3 y al 3 % del óptimo para el Grafo 4. Los resultados obtenidos mediante el enfoque con caminos son altamente satisfactorios aventajando en forma clara al enfoque por aristas.

Se aprecia que las configuraciones que obtienen los mejores resultados tienen un valor alto para el parámetro α, del mismo modo que en algunas de las versiones del

enfoque basado en aristas. Por esa particularidad se consideró interesante analizar en profundidad la relación existente entre la calidad de los resultados obtenidos y cada uno de los parámetros calibrados en forma independiente. El análisis se centró en la versión *Caminos* debido a que tiene una mayor cantidad de configuraciones dentro del enfoque basado en caminos.

En la figura 7.1 se presentan gráficas del costo promedio para cada una de las instancias por configuración agrupando según el valor del parámetro α. Se observa que la calidad promedio de las soluciones mejora al utilizar valores crecientes de α, obteniéndose los mejores resultados con $\alpha = 5$. El parámetro α incide en forma similar sobre la calidad de las soluciones obtenidas por la versión *HCF-MMAS*.

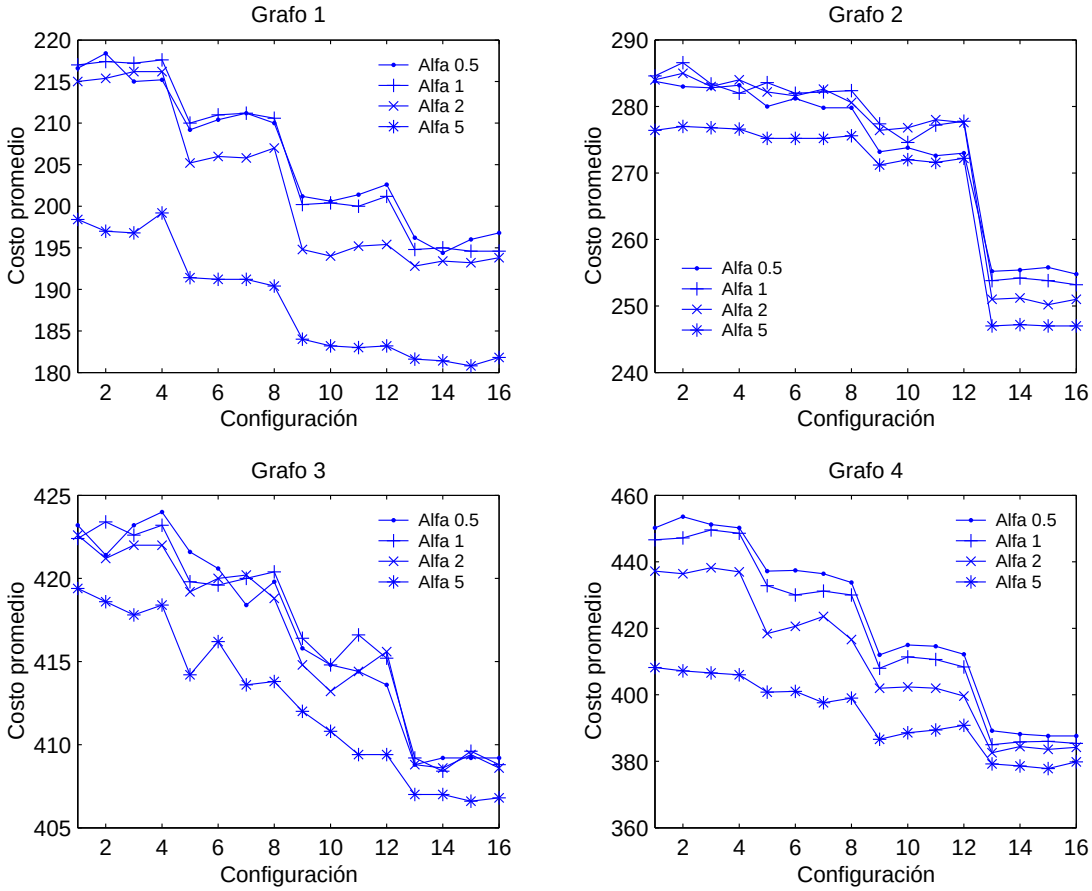

Figura 7.1: Resultado promedio por configuración agrupados por α

En la figura 7.2 se presentan gráficas del costo promedio para cada una de las instancias por configuración agrupando según el valor del parámetro β. Se observa que la calidad promedio de las soluciones mejora al utilizar valores crecientes de β, obteniéndose los mejores resultados con $\beta = 5$. El parámetro β incide en forma similar sobre la calidad de las soluciones obtenidas por la versión *HCF-MMAS*.

En la figura 7.3 se presentan gráficas del costo promedio para cada una de las instancias por configuración agrupando según el valor del parámetro ρ. Se observa que la incidencia del parámetro ρ en la calidad promedio de las soluciones obtenidas es menos clara que en los otros parámetros. Los mejores resultados se obtienen con $\rho = 0.5$, pero

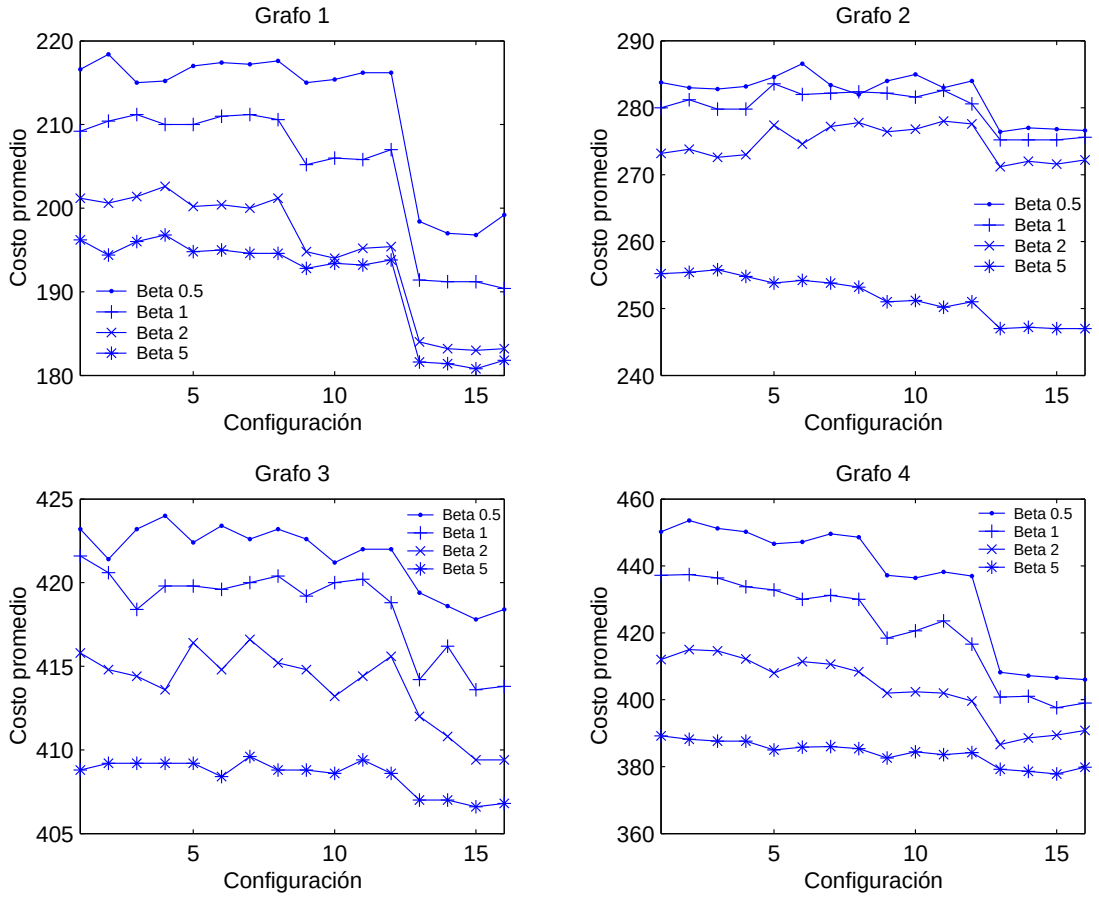

Figura 7.2: Resultado promedio por configuración agrupados por β

utilizar otro valor tiene un efecto leve en la calidad de los resultados obtenidos.

Como se producían mejoras en las soluciones obtenidas al trabajar con valores crecientes de α y β, se realizaron pruebas no formalizadas para evaluar el impacto de utilizar valores mayores a cinco. Se constataron que existen mejoras leves hasta valores cercanos a diez, empeorando al utilizar valores mayores a diez. El comportamiento detectado de los parámetros α y β es una particularidad de la aplicación al GSP siguiendo un enfoque basado en caminos. El aumento de los valores de ambos parámetros puede ser interpretado como acrecentar la probabilidad de elegir los caminos para los cuales la multiplicación entre los valores de los rastros de feromona y la visibilidad de la regla de transición de estados es mayor. Por este motivo, se realizaron pruebas no formalizadas en las que las hormigas seleccionaban en forma determinística el camino con menor costo y el camino que maximiza la multiplicación entre los valores de los rastros de feromona y la visibilidad, constatándose en ambos casos que los resultados obtenidos eran peores.

A pesar de que existen leves mejoras al aumentar el valor de los parámetros α y β, se estableció como mejor configuración la que fija ambos parámetros en cinco. Esta decisión se basó en mantener la homogeneidad entre los valores usados para los enfoques basados en aristas y en caminos, y en evitar el privilegiar sobremanera a estrategias excesivamente greedy porque se podría estar sesgando la calibración por trabajar sobre instancias desconexas.

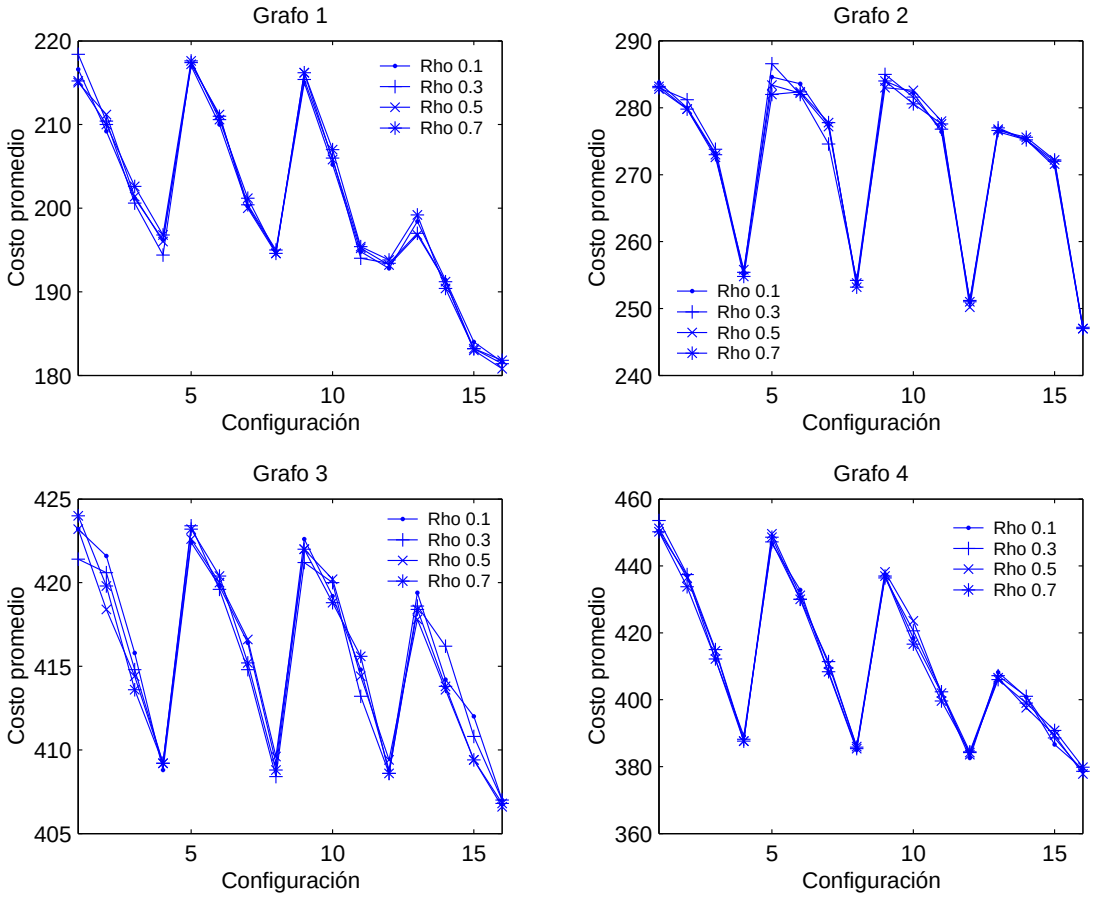

Figura 7.3: Resultado promedio por configuración agrupados por ρ

7.2.2. Evaluación

La evaluación de las versiones propuestas fue realizada sobre los grafos presentados en la sección 7.1.2. Las instancias del conjunto de grafos de evaluación no presentan desconexiones, por esta razón la comparación entre los resultados obtenidos por Nesmachnow et al. [129] y los algoritmos propuestos en este trabajo es más justa que la comparación realizada en la calibración.

Enfoque basado en aristas

Se realizaron 15 ejecuciones independientes para cada versión del enfoque basado en aristas sobre el conjunto de instancias de evaluación, utilizando las mejores configuraciones de parámetros determinadas en la etapa de calibración. Se utilizó un criterio de parada de esfuerzo prefijado de 500 iteraciones. Las pruebas se realizaron sobre la plataforma de ejecución 1 (Intel P4). En la tabla 7.11 se presentan los resultados obtenidos por las versiones del enfoque basado en aristas, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. Se señalan en negrita los mejores resultados obtenidos por estas versiones para cada instancia. La tabla incluye, a modo de referencia, la mejor solución conocida hasta el momento.

Tabla 7.11: Resultados obtenidos mediante el enfoque basado en aristas

		Grafo 100-10	**Grafo 75-25**	**Grafo 50-15**
Puro	*Promedio*	720.88	1158.17	1941.05
	Desv. Est.	32.78	23.33	95.03
	Mejor	665	1094.25	1729.71
	Tiempo Prom.	97.12	254.66	84.26
Term	*Promedio*	568.60	1086.76	1832.55
	Desv. Est.	22.69	27.15	65.45
	Mejor	523	1032.02	**1694.22**
	Tiempo Prom.	92.47	250.37	80.42
Alternativo	*Promedio*	**421.20**	**1068.03**	**1814.23**
	Desv. Est.	14.80	41.04	42.31
	Mejor	**398**	**1012.33**	1758.35
	Tiempo Prom.	70.35	201.65	64.89
Alt. Norm.	*Promedio*	850.15	1321.16	2046.24
	Desv. Est.	32.14	27.45	50.25
	Mejor	783	1256.40	1986.25
	Tiempo Prom.	78.27	220.15	69.55
Privilegiada2	*Promedio*	715.33	1249.35	2015.01
	Desv. Est.	25.91	34.83	48.70
	Mejor	650	1195.60	1953.10
	Tiempo Prom.	103.33	263.52	88.77
Privilegiada5	*Promedio*	779.73	1221.02	2014.65
	Desv. Est.	34.59	27.44	60.12
	Mejor	708	1173.18	1919.74
	Tiempo Prom.	106.18	280.60	90.30
Privilegiada10	*Promedio*	824.27	1250.98	2007.45
	Desv. Est.	31.04	25.44	47.19
	Mejor	767	1213.40	1945.03
	Tiempo Prom.	110.70	256.88	93.35
Mejor solución conocida		291	773.09	1353.49

Se aprecia que los mejores resultados los obtiene *Alternativo*, mostrando que la utilización de un esquema distinto al tradicional para la cantidad de feromona a depositar es provechoso. *Term* también aventaja a *Puro* en la calidad de las soluciones obtenidas, confirmando que incorporar conocimiento en la visibilidad sobre si la arista considerada es incidente a un terminal mejora la calidad de las soluciones obtenidas. La familia de versiones *Privilegiada* muestra resultados pobres. La utilización de aristas privilegiadas en las instancias del conjunto de evaluación suele conducir a explorar caminos que no permiten encontrar soluciones factibles, existiendo un porcentaje alto de hormigas que no contribuyen en la búsqueda. *Alt. Norm.* presenta los peores resultados, indicando que la normalización de la función de calidad no es una buena idea.

Alternativo tiene el menor tiempo promedio de ejecución, seguido por *Alt. Norm.*, *Term* y *Puro*. La familia de versiones *Privilegiada* tiene los peores tiempos de ejecución para todas las instancias debido a que ante la imposibilidad de construir caminos entre

pares de terminales, el mecanismo de reintentos acrecienta el tiempo de ejecución.

La comparación con los resultados obtenidos por Nesmachnow et al. muestra que *Alternativo* presenta resultados competitivos sobre dos de las tres instancias. Los resultados obtenidos por el algoritmo ACO propuesto para el *Grafo 100-10* son superados solamente por CHC2, aunque el costo promedio de las soluciones está a un 44.75 % del mejor valor conocido. En el *Grafo 75-25*, los resultados son levemente inferiores a los obtenidos por el GA, estando el costo promedio a un 38.15 % del mejor valor conocido. Finalmente, los resultados obtenidos para el *Grafo 50-15* son pobres, siendo aventajado inclusive por SA que no es un algoritmo poblacional. Sin embargo, el costo promedio de las soluciones obtenidas por *Alternativo* está a un 34.04 % del mejor valor conocido, siendo la instancia en la que la distancia a la mejor solución conocida es mínima. La mejor de las versiones del enfoque basado en aristas (*Alternativo*) presenta menores tiempos de ejecución promedio que todas las técnicas poblacionales propuestas por Nesmachnow et al. excepto los algoritmos CHC.

A partir de la evaluación experimental realizada es posible concluir que uno de los algoritmos ACO con enfoque basado en aristas propuesto para la resolución del GSP (*Alternativo*) muestra resultados aceptables con tiempos de ejecución bajos. Sin embargo, no es competitivo con el mejor algoritmo propuesto hasta el momento (CHC2).

Enfoque basado en caminos

Se realizaron 15 ejecuciones independientes para cada versión del enfoque basado en caminos sobre el conjunto de instancias de evaluación, utilizando las mejores configuraciones de parámetros determinadas en la etapa de calibración. Se utilizó un criterio de parada de esfuerzo prefijado de 500 iteraciones. Las pruebas se realizaron sobre la plataforma de ejecución 2 (AMD Athlon 64). En la tabla 7.12 se presentan los resultados obtenidos por las versiones del enfoque basado en caminos, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. Se señalan en negrita los mejores resultados obtenidos por estas versiones para cada instancia. La tabla incluye, a modo de referencia, la mejor solución conocida hasta el momento.

Tabla 7.12: Resultados obtenidos mediante el enfoque basado en caminos

		Grafo 100-10	**Grafo 75-25**	**Grafo 50-15**
Caminos	*Promedio*	306.00	1001.63	1672.28
	Desv. Est.	2.95	11.66	19.27
	Mejor	299	981.23	1645.41
	Tiempo Prom.	1177.67	1071.12	413.95
HCF-$\mathcal{MM}$AS	*Promedio*	**298.13**	**991.32**	**1631.66**
	Desv. Est.	2.36	9.86	24.75
	Mejor	**293**	**976.76**	**1581.18**
	Tiempo Prom.	1243.91	1106.23	431.12
Mejor solución conocida		291	773.09	1353.49

Se aprecia que ambas versiones obtienen buenos resultados, mostrando que la utilización de un enfoque basado en caminos es promisoria. La versión *HCF-$\mathcal{MM}AS$* presenta resultados levemente superiores, aunque las diferencias están por debajo de la

desviación estándar de las ejecuciones realizadas, por lo que no se puede formular conclusiones terminantes. El tiempo de ejecución promedio de *Caminos* es levemente inferior al de *HCF-$\mathcal{MM}AS$*.

La comparación con los resultados obtenidos por Nesmachnow et al. muestra que ambas versiones presentan resultados competitivos en las tres instancias. El análisis de los resultados obtenidos se centra en la versión *HCF-$\mathcal{MM}AS$*, pero es extrapolable a *Caminos*. El costo promedio de las soluciones obtenidas por *HCF-$\mathcal{MM}AS$* para el *Grafo 100-10* es inferior al obtenido por todas los algoritmos propuestos por Nesmachnow et al., estando solamente a un 2.45 % del mejor valor conocido. En el *Grafo 75-25*, los resultados son similares a los obtenidos por el GA, estando el costo promedio a un 28.23 % del mejor valor conocido. Finalmente, los resultados obtenidos para el *Grafo 50-15* son similares a los del GA, estando el costo promedio de las soluciones obtenidas a 20.55 % del mejor valor conocido. Aunque las plataformas de ejecución utilizadas en esta prueba y por Nesmachnow et al. no coinciden exactamente, el tiempo de ejecución promedio de las versiones del enfoque orientado a caminos es considerablemente superior.

La comparación entre la calidad de las soluciones obtenidas por los dos enfoques propuestos posiciona a las versiones que tienen un enfoque basado en caminos como más promisorias a pesar de ser más costosos computacionalmente (el tiempo de ejecución en algunas instancias es 15 veces mayor).

Debido a la gran cantidad de versiones propuestas es necesario centrar el análisis en una única versión de modo de realizar un estudio más profundo de ella. Las versiones con un enfoque basado en caminos obtienen muy buenos resultados pero a costa de un mayor tiempo de cómputo. En la versión *HCF-$\mathcal{MM}AS$* solamente una hormiga actualiza el rastro de feromona, por lo que el efecto de la cantidad de individuos de la población sobre la calidad de las soluciones obtenidas debería ser menor. Esto podría permitir reducir el tiempo de ejecución perdiendo poca calidad en las soluciones. Por este motivo, se decidió centrar el análisis en la versión *HCF-$\mathcal{MM}AS$*.

7.2.3. Efecto del tamaño de la población

En esta subsección se estudia el efecto del tamaño de la población en las soluciones obtenidas y en el tiempo de ejecución. El estudio se centra en la versión *HCF-$\mathcal{MM}AS$*.

Se realizaron 15 ejecuciones independientes de *HCF-$\mathcal{MM}AS$* sobre el conjunto de instancias de evaluación, utilizando para el parámetro tamaño de la población los valores 10, 20, 30, 60, 90 y 120. El resto de los parámetros fueron tomados de la mejor configuración determinada en la etapa de calibración. Se utilizó un criterio de parada de esfuerzo prefijado de 500 iteraciones. Las pruebas se realizaron sobre la plataforma de ejecución 2 (AMD Athlon 64). En la tabla 7.13 se presentan los resultados obtenidos por *HCF-$\mathcal{MM}AS$* en función del tamaño de la población, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. Se señalan en negrita los mejores resultados obtenidos al estudiar el efecto del tamaño de la población. La tabla incluye, a modo de referencia, la mejor solución conocida hasta el momento.

En las figuras 7.4, 7.5 y 7.6 se presentan en forma gráfica los resultados obtenidos para las instancias *Grafo 100-10*, *Grafo 75-25* y *Grafo 50-15*, respectivamente. Cada figura se compone de dos gráficas. La gráfica de la izquierda muestra el mejor costo, el costo promedio y el peor costo para los tamaños de población estudiados. La gráfica de la derecha muestra el tiempo de ejecución promedio, en punteado se indica la proyección

Tabla 7.13: Resultados de *HCF-$\mathcal{MMAS}$* según el tamaño de la población

Individuos		**Grafo 100-10**	**Grafo 75-25**	**Grafo 50-15**
10	*Promedio*	307.20	1022.90	1712.04
	Desv. Est.	4.80	15.83	37.66
	Mejor	299	985.67	1643.61
	Tiempo Prom.	97.77	85.91	33.02
20	*Promedio*	304.33	1009.24	1701.10
	Desv. Est.	3.39	15.49	44.41
	Mejor	297	975.1	**1572.57**
	Tiempo Prom.	194.3	171.48	66.15
30	*Promedio*	303.13	1007.09	1691.63
	Desv. Est.	2.17	14.39	25.79
	Mejor	300	970.65	1652.14
	Tiempo Prom.	321.91	259.43	108.31
60	*Promedio*	300.13	998.94	1671.73
	Desv. Est.	2.29	10.38	19.93
	Mejor	296	981.84	1626.87
	Tiempo Prom.	640.39	518.78	216.02
90	*Promedio*	299.33	993.67	1663.89
	Desv. Est.	3.35	13.93	25.06
	Mejor	**292**	**948.99**	1622.26
	Tiempo Prom.	957.94	777.09	323.93
120	*Promedio*	**298.13**	**991.32**	**1631.66**
	Desv. Est.	2.36	9.86	24.75
	Mejor	293	976.76	1581.18
	Tiempo Prom.	1243.91	1106.23	431.12
Mejor solución conocida		291	773.09	1353.49

lineal del tiempo de ejecución promedio con una población de 10 individuos.

Las diferencias en la calidad de las soluciones obtenidas por poblaciones de distinto tamaño están por debajo de la desviación estándar de las ejecuciones realizadas, por lo que no pueden sacarse conclusiones terminantes. Sin embargo, se puede apreciar una tendencia a disminuir el costo promedio de las soluciones obtenidas con tamaños crecientes de la población. El incremento en el tiempo de ejecución promedio al aumentar el tamaño de la población puede considerarse lineal dentro de los márgenes de error de las pruebas. El estudio permite concluir que es posible trabajar con un tamaño de población menor al original, logrando una reducción significativa en el tiempo de ejecución sin comprometer la calidad de los resultados obtenidos. Por ejemplo, si se reduce el tamaño de la población de 120 a 10 hormigas, el costo promedio de las soluciones empeora entre un 3 y 5 % dependiendo de la instancia, mientras que el tiempo de ejecución promedio se reduce a la duodécima parte.

El estudio de la incidencia de la cantidad de caminos considerados para satisfacer un requerimiento de conexión entre dos terminales y la incorporación de un operador de búsqueda local en la calidad de las soluciones obtenidas, se realiza sobre poblaciones con 10 y 60 hormigas. La elección de la población con 10 hormigas se basa en que presenta

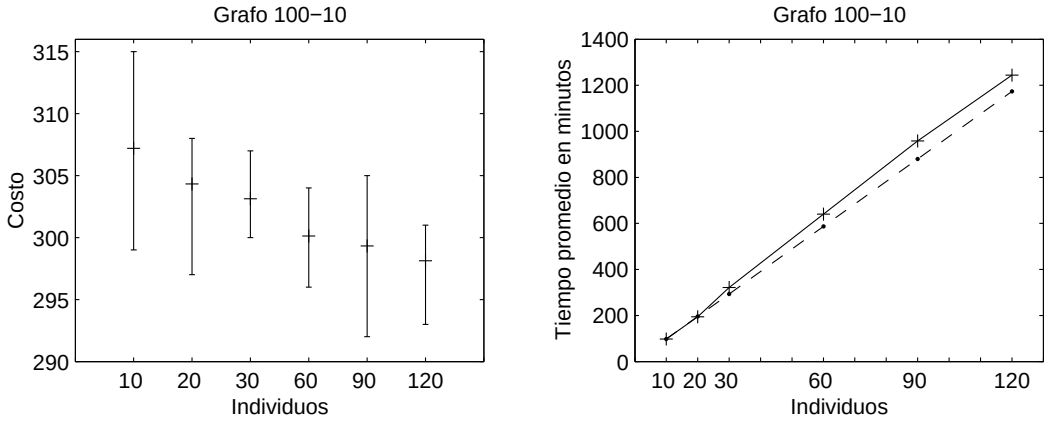

Figura 7.4: Efecto del tamaño de la población para el *Grafo 100-10*

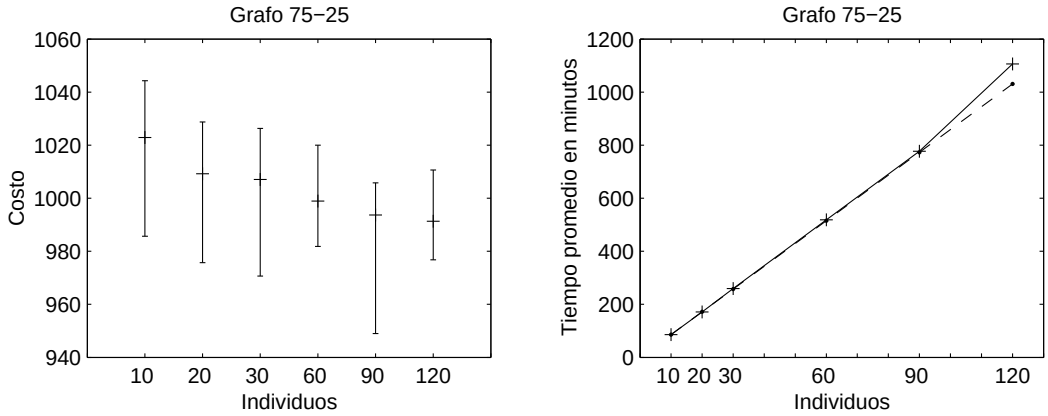

Figura 7.5: Efecto del tamaño de la población para el *Grafo 75-25*

el menor tiempo de ejecución dentro de los tamaños de población considerados. Por otro lado, es necesario trabajar con un tamaño de población que obtenga soluciones de muy buena calidad, pero que requiera un tiempo de ejecución menor que la población de 120 hormigas, ya que los factores que se estudian aumentarán la complejidad computacional y por consiguiente el tiempo de ejecución. Un tamaño de población de 60 hormigas se considera un buen balance ya que permite una reducción sensible en el tiempo de ejecución con poca pérdida en la calidad de las soluciones obtenidas.

7.2.4. Efecto de la cantidad de caminos

En esta subsección se estudia el efecto de la cantidad de caminos considerados para satisfacer un requerimiento de conexión entre dos terminales en las soluciones obtenidas y en el tiempo de ejecución. Se realizaron 15 ejecuciones independientes de *HCF-$\mathcal{MMAS}$* sobre el conjunto de instancias de evaluación, utilizando para el parámetro cantidad de caminos los valores 5, 10 y 15. El resto de los parámetros fueron tomados de la mejor configuración determinada en la etapa de calibración. Se utilizó un criterio de parada de esfuerzo prefijado de 500 iteraciones. Las pruebas se realizaron sobre la plataforma de ejecución 2 (AMD Athlon 64).

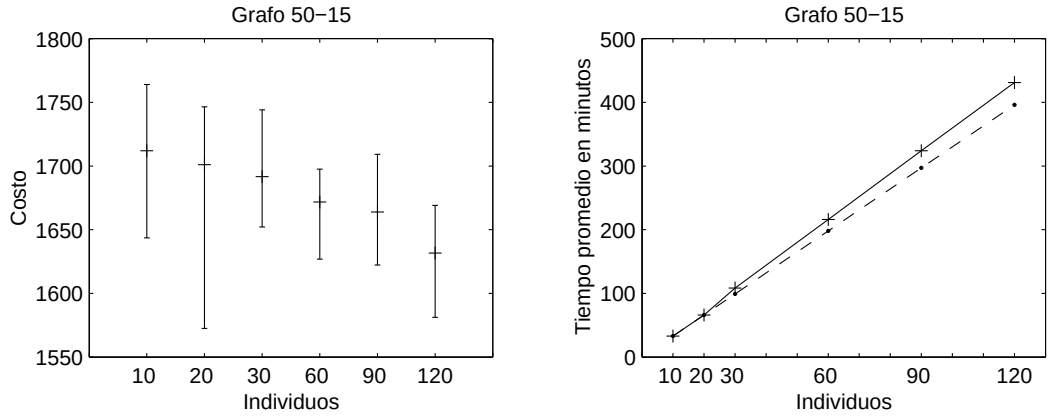

Figura 7.6: Efecto del tamaño de la población para el *Grafo 50-15*

En la tabla 7.14 se presentan los resultados obtenidos por *HCF-$\mathcal{MMAS}$* con una población de 60 hormigas según la cantidad de caminos considerados, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. Se señalan en negrita los mejores resultados obtenidos al estudiar el efecto de la cantidad de caminos considerados. La tabla incluye, a modo de referencia, la mejor solución conocida hasta el momento.

Tabla 7.14: Resultados de *HCF-$\mathcal{MMAS}$* con 60 hormigas

Caminos		**Grafo 100-10**	**Grafo 75-25**	**Grafo 50-15**
5	*Promedio*	308.33	1027.43	1726.42
	Desv. Est.	3.79	10.97	26.71
	Mejor	301	1009.76	1682.37
	Tiempo Prom.	278.24	278.11	100.71
10	*Promedio*	300.13	998.94	1671.73
	Desv. Est.	2.29	10.38	19.93
	Mejor	296	981.84	1626.87
	Tiempo Prom.	640.39	518.78	216.02
15	*Promedio*	**298.67**	**979.72**	**1641.86**
	Desv. Est.	3.27	15.51	25.96
	Mejor	**293**	**939.43**	**1598.78**
	Tiempo Prom.	957.3	869.90	330.27
Mejor solución conocida		291	773.09	1353.49

Si la cantidad de caminos considerados se incrementa a 15, se produce una mejora leve en la calidad de las soluciones de entre un 0.5 y 2 % dependiendo de la instancia, aunque las diferencias están por debajo de la desviación estándar de las ejecuciones, por esta razón no es posible formular conclusiones terminantes. El incremento provoca un aumento en el tiempo de ejecución de entre un 50 y 67 % dependiendo de la instancia. Si la cantidad de caminos considerados se reduce a 5, se produce una reducción en la calidad de las soluciones de entre un 2.7 y 3.3 % dependiendo de la instancia, aunque las diferencias son levemente superiores a la desviación estándar, por lo que no permiten

sacar conclusiones definitivas. La reducción provoca una disminución en el tiempo de ejecución a entre un 43 y 53 % del tiempo de ejecución con 10 caminos.

En la tabla 7.15 se presentan los resultados obtenidos por *HCF-$\mathcal{MMAS}$* con una población de 10 hormigas según la cantidad de caminos considerados, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. Se señalan en negrita los mejores resultados obtenidos al estudiar el efecto de la cantidad de caminos considerados. La tabla incluye, a modo de referencia, la mejor solución conocida hasta el momento.

Tabla 7.15: Resultados de *HCF-$\mathcal{MMAS}$* con 10 hormigas

Caminos		**Grafo 100-10**	**Grafo 75-25**	**Grafo 50-15**
5	*Promedio*	316.27	1061.78	1782.74
	Desv. Est.	2.43	12.46	34.83
	Mejor	312	1043.87	1717.98
	Tiempo Prom.	46.13	42.77	16.4
10	*Promedio*	**307.20**	1022.90	1712.04
	Desv. Est.	4.80	15.83	37.66
	Mejor	**299**	985.67	1643.61
	Tiempo Prom.	97.77	85.91	33.02
15	*Promedio*	307.27	**1004.02**	**1701.00**
	Desv. Est.	2.40	10.05	30.61
	Mejor	305	**979.35**	**1634.64**
	Tiempo Prom.	163.26	134.33	53.07
Mejor solución conocida		291	773.09	1353.49

Si la cantidad de caminos considerados se incrementa a 15, se provoca una mejora leve (menos de 1.8 %) en dos de las tres instancias incluso por debajo de la desviación estándar de las ejecuciones, mientras que no produce efectos en el *Grafo 100-10*. El incremento provoca un aumento en el tiempo de ejecución de entre un 56 y 66 % dependiendo de la instancia. Por otro lado, si la cantidad de caminos considerados se reduce a 5, se produce una reducción en la calidad de las soluciones de entre un 3.0 y 4.1 % dependiendo de la instancia, aunque las diferencias son levemente superiores a la desviación estándar, imposibilitando la formulación de conclusiones definitivas. La reducción provoca una disminución en el tiempo de ejecución a entre un 47 y 50 % del tiempo de ejecución con 10 caminos.

En términos generales, el incremento en la cantidad de caminos considerados para satisfacer un requerimiento de conexión entre dos terminales provoca pequeñas mejoras en la calidad de las soluciones obtenidas con la excepción de la utilización de una población de 10 hormigas para resolver el *Grafo 100-10*. Inclusive con estas mejoras en las soluciones, en dos instancias los resultados no son competitivos con CHC2, estando lejos de la mejor solución conocida (26.73 % para el *Grafo 75-25* y 21.31 % para el *Grafo 50-15*). La relación existente entre la cantidad de caminos y el tiempo promedio de ejecución es lineal dentro de los márgenes de error de las pruebas.

7.2.5. Búsqueda local

En esta subsección se estudia el efecto de incorporar un operador de búsqueda local sobre la calidad de las soluciones obtenidas y el tiempo de ejecución. Se realizaron 15 ejecuciones independientes de *HCF-$\mathcal{MM}AS$* sobre el conjunto de instancias de evaluación. Los parámetros fueron tomados de la mejor configuración determinada en la etapa de calibración. Se utilizó un criterio de parada de esfuerzo prefijado de 500 iteraciones. Las pruebas se realizaron sobre la plataforma de ejecución 2 (AMD Athlon 64). A pesar de que la plataforma de ejecución de esta prueba y la usada por Nesmachnow et al. no coinciden, los procesadores utilizados guardan cierta similitud en su desempeño, por lo que se tomarán los tiempos reportados por Nesmachnow et al. como referencia para comparar.

En la tabla 7.16 se presentan los resultados obtenidos por *HCF-$\mathcal{MM}AS$* con una población de 60 hormigas, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. Se señalan en negrita los mejores resultados obtenidos al evaluar la incorporación de un operador de búsqueda local. La tabla incluye, a modo de referencia, la mejor solución conocida hasta el momento.

Tabla 7.16: Resultados de *HCF-$\mathcal{MM}AS$* con 60 hormigas

Caminos		**Grafo 100-10**	**Grafo 75-25**	**Grafo 50-15**
Sin BL	*Promedio*	300.13	998.94	1671.73
	Desv. Est.	2.29	10.38	19.93
	Mejor	296	981.84	1626.87
	Tiempo Prom.	640.39	518.78	216.02
BL	*Promedio*	284.75	863.74	1503.62
	Desv. Est.	0.79	10.94	16.66
	Mejor	283	841.58	1476.21
	Tiempo Prom.	837.11	947.38	342.11
BL iterada	*Promedio*	**281.40**	**796.36**	**1389.24**
	Desv. Est.	0.63	7.22	5.74
	Mejor	**280**	**770.53**	**1383.49**
	Tiempo Prom.	1216.15	1763.25	647.99
Mejor solución conocida		291	773.09	1353.49

Se aprecia que las versiones *BL* y *BL iterada* mejoran considerablemente los resultados obtenidos por la versión *Sin BL*, aunque a costa de un incremento importante en el tiempo de ejecución promedio. Al incorporarse los operadores de búsqueda local planteados en este trabajo, el tiempo de ejecución promedio para la instancia *Grafo 75-25* es superior a la del *Grafo 100-10*. Este hecho puede deberse a que la complejidad computacional de los operadores de búsqueda local está estrechamente vinculada a la cantidad de terminales y requisitos de la instancia.

En la instancia *Grafo 100-10*, *BL* supera la mejor solución conocida previamente y el costo promedio de las soluciones obtenidas es inferior al que obtiene CHC2, aunque el tiempo de ejecución es catorce veces mayor. *BL* no alcanza los resultados obtenidos por CHC2 en el *Grafo 75-25*, pero supera al resto de las propuestas formuladas por Nesmachnow et al., siendo el tiempo de ejecución más de tres veces el de CHC2. Por

último en la instancia *Grafo 50-15* tampoco alcanza los resultados obtenidos por CHC2, pero supera al resto de las propuestas formuladas por Nesmachnow et al., siendo el tiempo de ejecución casi siete veces el de CHC2.

En la instancia *Grafo 100-10*, *BL iterada* supera la mejor solución conocida previamente (la solución obtenida es actualmente la mejor solución conocida para esta instancia y se presenta en el apéndice B) y el costo promedio de las soluciones obtenidas es inferior al que obtiene CHC2, aunque el tiempo de ejecución es más de veinte veces mayor. El costo promedio de las soluciones obtenidas está a un 0.5 % de la mejor solución conocida. En el *Grafo 75-25*, *BL iterada* también supera la mejor solución conocida previamente (la solución obtenida es actualmente la mejor solución conocida para esta instancia y se presenta en el apéndice B) y el costo promedio de las soluciones obtenidas es inferior al que obtiene CHC2, aunque el tiempo de ejecución es más de seis veces mayor. El costo promedio de las soluciones obtenidas está a un 3.35 % de la mejor solución conocida. Finalmente, *BL iterada* obtiene soluciones con costo promedio inferior a CHC2 en la instancia *Grafo 50-15*, aunque con un tiempo de ejecución más de doce veces el de CHC2. El costo promedio de las soluciones obtenidas está solamente a un 2.64 % de la mejor solución conocida.

En la tabla 7.17 se presentan los resultados obtenidos por *HCF-$\mathcal{MMAS}$* con una población de 10 hormigas, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. Se señalan en negrita los mejores resultados obtenidos al evaluar la incorporación de un operador de búsqueda local. La tabla incluye, a modo de referencia, la mejor solución conocida hasta el momento.

Tabla 7.17: Resultados de *HCF-$\mathcal{MMAS}$* con 10 hormigas

Caminos		**Grafo 100-10**	**Grafo 75-25**	**Grafo 50-15**
Sin BL	*Promedio*	307.20	1022.90	1712.04
	Desv. Est.	4.80	15.83	37.66
	Mejor	299	985.67	1643.61
	Tiempo Prom.	97.77	85.91	33.02
BL	*Promedio*	287.60	885.92	1541.04
	Desv. Est.	1.68	6.68	22.10
	Mejor	285	874.19	1495.89
	Tiempo Prom.	141.66	154.91	57.47
BL iterada	*Promedio*	**282.47**	**808.24**	**1430.16**
	Desv. Est.	0.64	6.08	15.72
	Mejor	**282**	**796.51**	**1394.97**
	Tiempo Prom.	204.25	301.30	109.32
Mejor solución conocida		291	773.09	1353.49

Se comprueba que también al trabajar con una población de 10 hormigas, las versiones *BL* y *BL iterada* mejoran considerablemente los resultados obtenidos por la versión *Sin BL*, aunque a costa de un incremento en el tiempo de ejecución.

En la instancia *Grafo 100-10*, *BL* supera la mejor solución conocida previamente y el costo promedio de las soluciones obtenidas es inferior al que obtiene CHC2, aunque el tiempo de ejecución es más de dos veces mayor. *BL* no alcanza los resultados obtenidos

por CHC2 en el *Grafo 75-25*, pero supera al resto de las propuestas formuladas por Nesmachnow et al., siendo el tiempo de ejecución inferior al de CHC2. Finalmente, en la instancia *Grafo 50-15* tampoco alcanza los resultados obtenidos por CHC2, pero supera al resto de las propuestas formuladas por Nesmachnow et al., siendo el tiempo de ejecución similar al de CHC2.

En la instancia *Grafo 100-10*, *BL iterada* supera la mejor solución conocida previamente y el costo promedio de las soluciones obtenidas es inferior al que obtiene CHC2, aunque el tiempo de ejecución es casi cuatro veces mayor. El costo promedio de las soluciones obtenidas está a un 0.88 % de la mejor solución conocida. En el *Grafo 75-25*, el costo promedio de las soluciones obtenidas por *BL iterada* es inferior al de CHC2, con un tiempo de ejecución similar. El costo promedio de las soluciones obtenidas está a un 4.89 % de la mejor solución conocida. Finalmente, *BL iterada* obtiene soluciones con costo promedio inferior a CHC2 en la instancia *Grafo 50-15*, siendo el tiempo de ejecución el doble del de CHC2. El costo promedio de las soluciones obtenidas está a un 5.66 % de la mejor solución conocida.

Con el objetivo de estudiar la evolución en la calidad de las soluciones, se presenta un estudio de la distribución del tiempo de ejecución (RTD, por sus siglas en inglés). La RTD muestra la probabilidad que un algoritmo obtenga una solución para una instancia a una cierta distancia del óptimo en función del tiempo de ejecución. En el caso particular de este estudio, se utiliza la distancia a la mejor solución conocida, ya que no se conoce la solución óptima de cada instancia y el número de iteración, ya que no estaba disponible el tiempo de ejecución en cada iteración. Las RTDs que se muestran a continuación fueron elaboradas utilizando las mismas ejecuciones que en la presentación de resultados previa.

En las figuras 7.7 y 7.8 se muestran las RTDs para el *Grafo 100-10* con poblaciones de 60 y 10 hormigas, respectivamente. La gráfica de la izquierda muestra la probabilidad de obtener soluciones hasta a 0.1 de la mejor solución conocida entre las iteraciones 1 y 100. Mientras que la gráfica de la derecha se concentra en la probabilidad de obtener soluciones hasta a 0.05 de la mejor solución conocida entre las iteraciones 1 y 25. En las figuras 7.9 y 7.10 se muestran RTDs para el mismo grafo, pero con la probabilidad en función de la cantidad de soluciones construidas. Las gráficas de la izquierda muestran la probabilidad de obtener soluciones hasta a 0.05 de la mejor solución conocida construyendo entre 0 y 1000 soluciones[1]. Mientras que las gráficas de la derecha se concentran en la probabilidad de obtener soluciones hasta a 0.05 de la mejor solución conocida construyendo entre 1000 y 5000 soluciones (la cantidad máxima de soluciones que se construye con una población de 10 hormigas es 5000).

En el *Grafo 100-10* en pocas iteraciones se pueden obtener soluciones de excelente calidad con alta probabilidad. Al utilizar una población de 60 hormigas, la probabilidad de estar al 0.05 de la mejor solución conocida es mayor que 0.5 en la iteración 8 e igual a uno en la iteración 15, y la probabilidad de estar al 0.01 de la mejor solución conocida es mayor que 0.5 en la iteración 58 e igual a uno en la iteración 92. Al utilizar una población de 10 hormigas, la probabilidad de estar al 0.05 de la mejor solución conocida es mayor que 0.5 en la iteración 15 e igual a uno en la iteración 25, y la probabilidad de estar al 0.01 de la mejor solución conocida es mayor que 0.5 en la iteración 320. Sin embargo, en las 500 iteraciones no se alcanza la distancia 0.01 con probabilidad uno, la probabilidad de estar a 0.01 de la mejor solución conocida es 0.6.

[1]Se consideran como soluciones construidas las que involucran la construcción inicial y la aplicación del operador de búsqueda local.

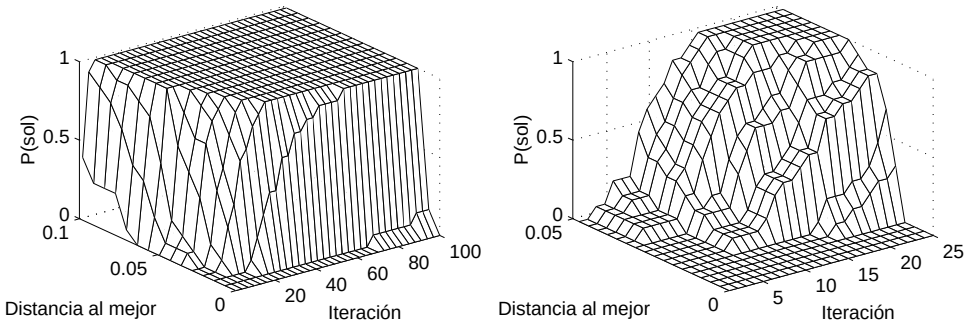

Figura 7.7: RTD de *HCF-$\mathcal{MM}AS$* con 60 hormigas para el *Grafo 100-10*

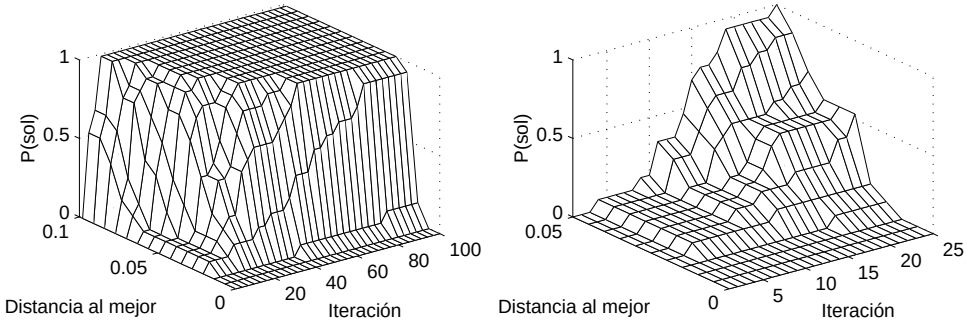

Figura 7.8: RTD de *HCF-$\mathcal{MM}AS$* con 10 hormigas para el *Grafo 100-10*

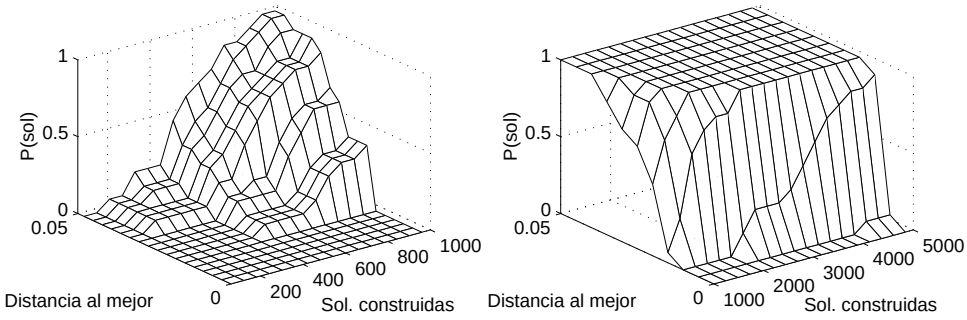

Figura 7.9: RTD de *HCF-$\mathcal{MM}AS$* con 60 hormigas para el *Grafo 100-10*

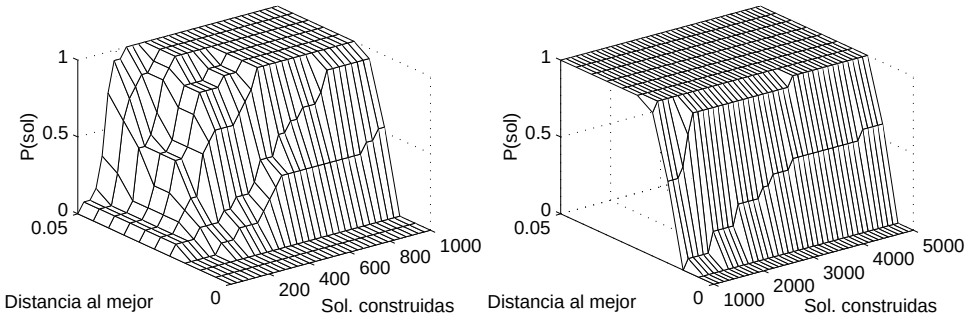

Figura 7.10: RTD de *HCF-$\mathcal{MM}AS$* con 10 hormigas para el *Grafo 100-10*

Al comparar la evolución en la calidad de las soluciones en función de la cantidad de las soluciones construidas, es posible apreciar que en el primer intervalo considerado el algoritmo con tamaño de población 10 tiene probabilidades más altas de estar más cerca de la mejor solución conocida. Sin embargo, en etapas avanzadas de la búsqueda, la situación se empareja.

En las figuras 7.11 y 7.12 se presentan las RTDs para el *Grafo 75-25* con poblaciones de 60 y 10 hormigas, respectivamente. Cada figura se compone de dos gráficas. La gráfica de la izquierda muestra la probabilidad de obtener soluciones hasta a 0.1 de la mejor solución conocida entre las iteraciones 1 y 100, mientras que la de la derecha muestra la probabilidad de obtener soluciones hasta a 0.05 de la mejor solución conocida entre las iteraciones 101 y 500. En las figuras 7.13 y 7.14 se muestran RTDs para el mismo grafo, pero con la probabilidad en función de la cantidad de soluciones construidas. Las gráficas de la izquierda muestran la probabilidad de obtener soluciones hasta a 0.01 de la mejor solución conocida construyendo entre 0 y 1000 soluciones. Mientras que las gráficas de la derecha se concentran en la probabilidad de obtener soluciones hasta a 0.05 de la mejor solución conocida construyendo entre 1000 y 5000 soluciones.

En el *Grafo 75-25* en pocas iteraciones se pueden obtener soluciones de muy buena calidad con alta probabilidad. Sin embargo, la mejora en la calidad de las soluciones con las sucesivas iteraciones es lenta. Al utilizar una población de 60 hormigas, la probabilidad de estar al 0.05 de la mejor solución conocida es mayor que 0.5 en la iteración 65 e igual a uno en la iteración 156, mientras que en la iteración 500 la probabilidad de estar a 0.033 es mayor que 0.5. Al utilizar una población de 10 hormigas, la probabilidad de estar al 0.05 de la mejor solución conocida es mayor que 0.5 en la iteración 473, sin llegar a tener probabilidad uno en las 500 iteraciones.

Al igual que para el *Grafo 100-10*, en el primer intervalo considerado el algoritmo con una población de 10 hormigas tiene probabilidades más altas de estar más cerca de la mejor solución conocida. Del mismo modo, la tendencia se revierte en etapas avanzadas de la búsqueda, logrando el algoritmo con una población de 60 hormigas cuotas levemente superiores de confianza para una misma distancia a la mejor solución conocida.

En las figuras 7.15 y 7.16 se presentan las RTDs para el *Grafo 50-15* con poblaciones de 60 y 10 hormigas, respectivamente. Cada figura se compone de dos gráficas. La gráfica de la izquierda muestra la probabilidad de obtener soluciones hasta a 0.1 de la mejor solución conocida entre las iteraciones 1 y 100, mientras que la de la derecha muestra la probabilidad de obtener soluciones hasta a 0.05 de la mejor solución conocida entre las iteraciones 101 y 500. En las figuras 7.17 y 7.18 se muestran RTDs para el mismo grafo, pero con la probabilidad en función de la cantidad de soluciones construidas. Las gráficas de la izquierda muestran la probabilidad de obtener soluciones hasta a 0.1 de la mejor solución conocida construyendo entre 0 y 1000 soluciones. Mientras que las gráficas de la derecha se concentra en la probabilidad de obtener soluciones hasta a 0.05 de la mejor solución conocida construyendo entre 1000 y 5000 soluciones.

En el *Grafo 50-15* la evolución es similar al del *Grafo 75-25*, obteniéndose en pocas iteraciones soluciones de buena calidad con alta probabilidad, pero con lentas mejoras en la calidad de las soluciones con las sucesivas iteraciones. Al utilizar una población de 60 hormigas, la probabilidad de estar al 0.05 de la mejor solución conocida es mayor que 0.5 en la iteración 85 e igual a uno en la iteración 421, mientras que en la iteración 500 la probabilidad de estar a 0.029 es mayor que 0.5. Al utilizar una población de 10

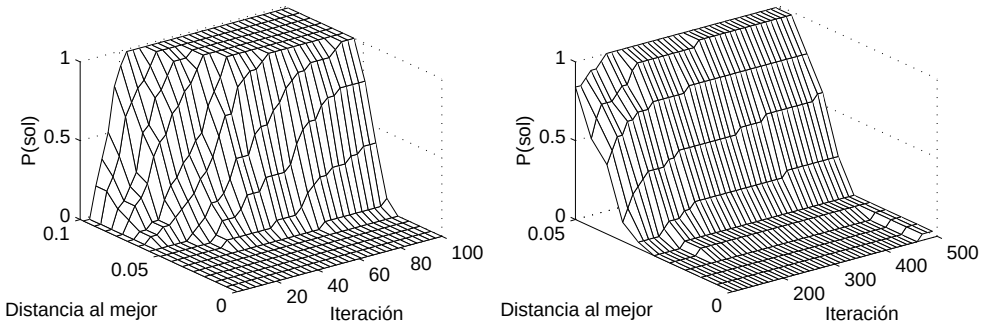

Figura 7.11: RTD de *HCF-$\mathcal{MM}AS$* con 60 hormigas para el *Grafo 75-25*

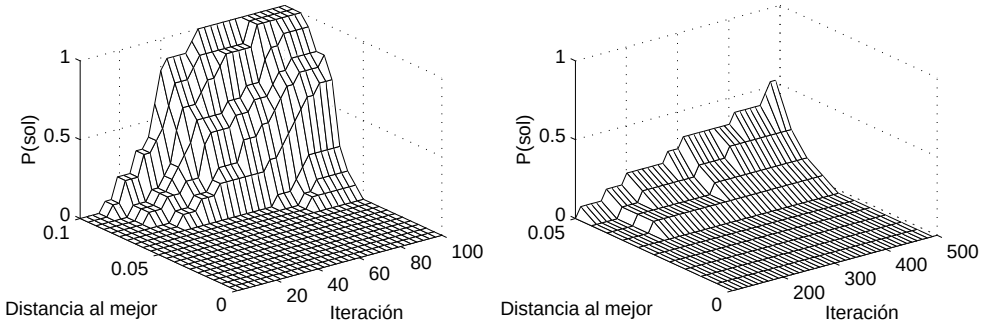

Figura 7.12: RTD de *HCF-$\mathcal{MM}AS$* con 10 hormigas para el *Grafo 75-25*

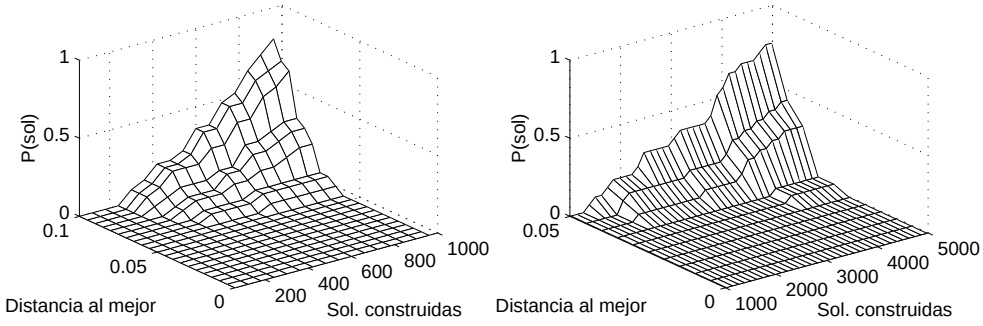

Figura 7.13: RTD de *HCF-$\mathcal{MM}AS$* con 60 hormigas para el *Grafo 75-25*

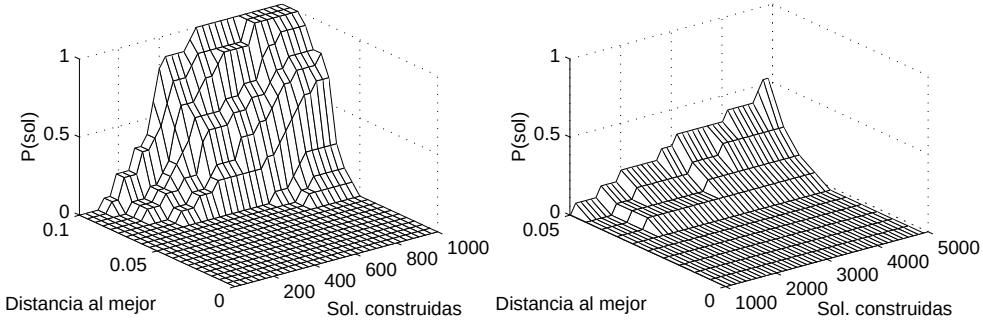

Figura 7.14: RTD de *HCF-$\mathcal{MM}AS$* con 10 hormigas para el *Grafo 75-25*

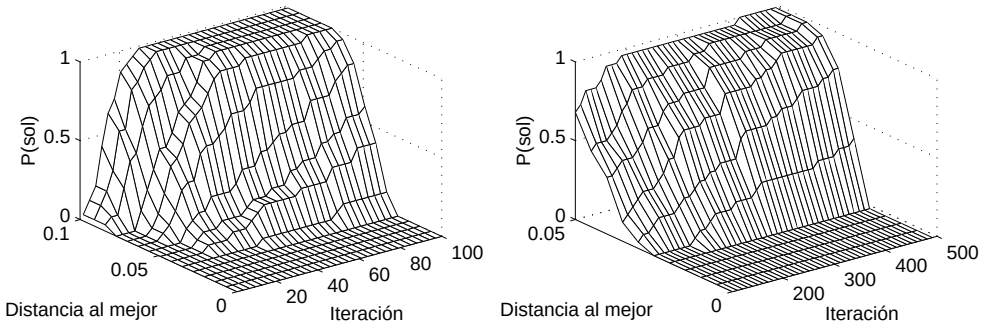

Figura 7.15: RTD de *HCF-$\mathcal{MM}AS$* con 60 hormigas para el *Grafo 50-15*

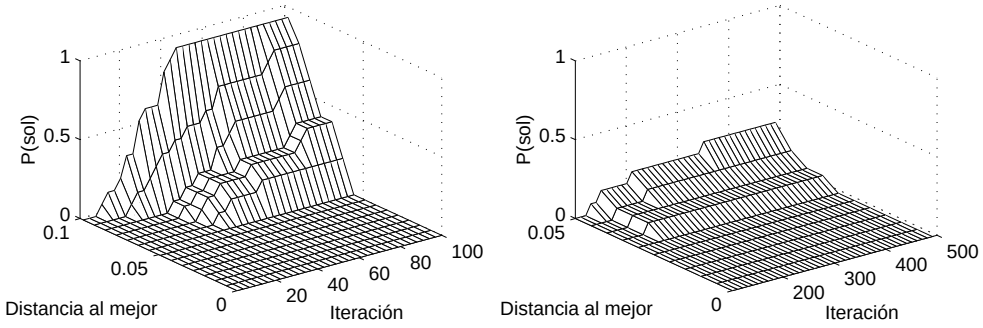

Figura 7.16: RTD de *HCF-$\mathcal{MM}AS$* con 10 hormigas para el *Grafo 50-15*

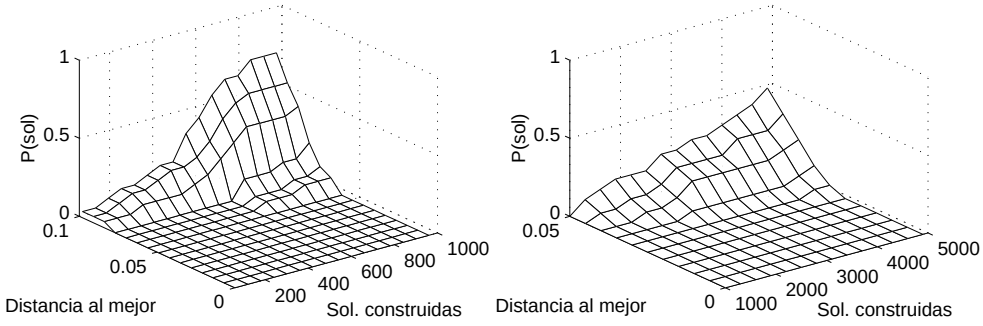

Figura 7.17: RTD de *HCF-$\mathcal{MM}AS$* con 60 hormigas para el *Grafo 50-15*

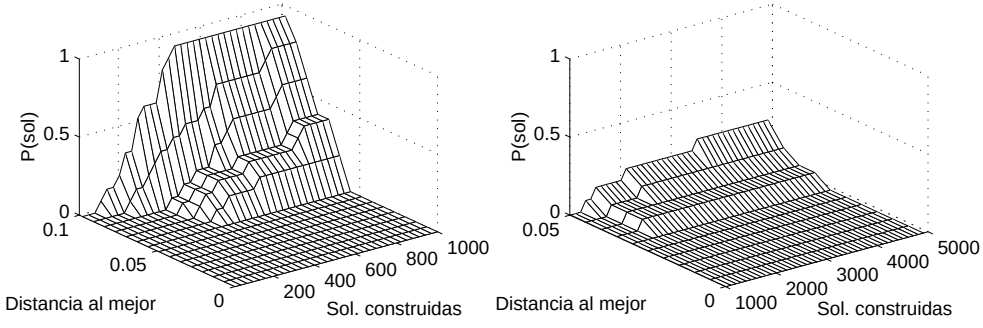

Figura 7.18: RTD de *HCF-$\mathcal{MM}AS$* con 10 hormigas para el *Grafo 50-15*

hormigas, la probabilidad de estar al 0.05 de la mejor solución conocida no alcanza el valor 0.5 en las 500 iteraciones. En la iteración 500, la probabilidad de estar a 0.061 de la mejor solución conocida es mayor que 0.5.

La evolución en la calidad de las soluciones en función de la cantidad de las soluciones construidas para el *Grafo 50-15* presenta idénticas características a los grafos anteriores. En un primer intervalo, el algoritmo con población de tamaño 10 tiene probabilidades más altas de estar más cerca de la mejor solución conocida. En etapas avanzadas de la búsqueda, la tendencia se revierte logrando el algoritmo con la población más numerosa una probabilidad levemente mayor de estar a una cierta distancia de la mejor solución conocida.

Los operadores de búsqueda local estudiados provocaron importantes mejoras en la calidad de las soluciones obtenidas por el algoritmo ACO, a costa de un incremento sustancial en el tiempo promedio de ejecución. En particular, la incorporación del operador de búsqueda local iterado permitió obtener resultados excelentes, incluso superando a la mejor solución conocida sobre dos de las instancias al trabajar con una población de 60 hormigas. Como contrapartida el tiempo de ejecución puede considerarse excesivo en comparación con los algoritmos propuestos por otros autores. La utilización de una población de 10 hormigas permite obtener resultados competitivos con CHC2 (en el *Grafo 100-10* supera claramente a CHC2, mientras que el costo promedio es inferior en las otras dos instancias aunque las diferencias quedan comprendidas dentro de las desviaciones estándares), utilizando un tiempo de ejecución que si bien es mayor, se considera aceptable. Los operadores de búsqueda local propuestos pueden ser fácilmente adaptados para ser utilizado por otras metaheurísticas para la resolución del GSP.

A partir del análisis de las RTDs realizado, es posible concluir que en algunas instancias se produce una evolución lenta y cierto nivel de estancamiento del algoritmo. Es posible realizar mejoras sobre el algoritmo ACO propuesto para intentar solucionar estos inconvenientes. El efecto del depósito y evaporación de los rastros de feromona a lo largo de las iteraciones produce la desaparición y la saturación de feromona en las componentes, que provoca el estancamiento del algoritmo. Por este motivo puede resultar útil incorporar algún mecanismo de reinicialización de los rastros al detectar el estado de estancamiento. Otra alternativa puede ser aumentar la cantidad de caminos considerados para satisfacer un requerimiento de conexión entre dos terminales en forma dinámica durante la ejecución del algoritmo, de forma de generar diversidad en la exploración de posibles soluciones.

Los RTDs también permitieron comprobar que la evolución en la calidad de las soluciones en función de la cantidad de las soluciones construidas tiene un patrón bien definido de comportamiento. Cuando la cantidad de soluciones construidas es baja, o equivalentemente el tiempo de ejecución, la utilización de una población de menor tamaño es preferible. Por el contrario, si la cantidad de soluciones construidas es alta, es recomendable trabajar con una población con una mayor cantidad de hormigas.

El tiempo de ejecución es superior al utilizado por el algoritmo CHC2. Una alternativa para reducir el tiempo de ejecución es volver a considerar los caminos determinados al resolver más de un requisito para un par de terminales. Actualmente, para cada requisito a partir de las aristas disponibles, se determinan los K caminos más cortos entre los que la hormiga realiza la selección. Si se debe resolver más de un requisito para un par de terminales, es posible identificar los caminos que deben ser descartados (porque comparten aristas con el camino seleccionado para resolver el requisito anterior) y realizar

la selección sobre los caminos disponibles, evitando volver a ejecutar el algoritmo para obtener los K caminos más cortos. Otra alternativa es aplicar el operador de búsqueda local iterada solamente sobre una de las hormigas de la población, determinándose el resto de las soluciones sin aplicar una búsqueda local o aplicando la búsqueda local sin iteraciones. Ambas alternativas propuestas pueden tener efectos sobre la calidad de las soluciones obtenidas por lo cual se debe realizar un estudio profundo para evaluar la validez de las alternativas propuestas.

7.3. Pruebas del modelo celular

En esta sección se presentan los experimentos realizados con el modelo celular para ACO propuesto en este trabajo para la resolución del GSP. El procedimiento de construcción de las soluciones de cada hormiga coincide exactamente con el utilizado por *HCF-$\mathcal{MM}AS$*, por lo que no se consideró necesario realizar una etapa de calibración para obtener la mejor configuración de parámetros, directamente se evaluaron las versiones sobre los grafos medianos.

Las versiones de ACO con un modelo celular consideradas son: *Von*, *Moore*, *Von+BL iterada* y *Moore+BL iterada* (presentadas en la sección 6.3). Todas las versiones fueron implementadas en el lenguaje C++, utilizando para las implementaciones paralelas la implementación MPICH de la biblioteca de desarrollo de programas paralelos y distribuidos MPI [90].

Para las pruebas del modelo celular se utilizó como plataforma de ejecución un cluster de cuatro computadoras AMD Athlon 64 Processor 3000+ de 2.0 Ghz con 1024 MB RAM y con sistema operativo SuSE Linux 10.0.

Se realizaron 15 ejecuciones independientes de las versiones del modelo celular sobre el conjunto de instancias de evaluación. Los parámetros fueron tomados de la mejor configuración determinada para *HCF-$\mathcal{MM}AS$* en la etapa de calibración.

El tamaño de la población se fijó en 60 hormigas de forma de poder comparar sus resultados con los del algoritmo secuencial *HCF-$\mathcal{MM}AS$* y por tener un tiempo de ejecución considerable volviendo atractiva su paralelización. Se utilizó un criterio de parada de esfuerzo prefijado de 500 iteraciones.

En la tabla 7.18 se presentan los resultados obtenidos por las versiones del modelo celular, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. Se señalan en negrita los mejores resultados obtenidos para el modelo celular. En dicha tabla se incluye, a modo de referencia, la mejor solución conocida hasta el momento.

Las diferencias en la calidad de las soluciones obtenidas entre las dos versiones con vecindad de Von Neumann y las dos con vecindad de Moore están por debajo de la desviación estándar, por lo cual no pueden considerarse significativas. Al no existir evidencia que determine que una vecindad es superior a la otra, se optó por concentrarse en estudiar las versiones del modelo celular con vecindad de Von Neumann por ser más simple e implicar menores tiempos de ejecución.

Von tiene perdida de calidad en las soluciones obtenidas y un incremento en el tiempo promedio de ejecución con respecto a la versión *HCF-$\mathcal{MM}AS$* de similares características. El promedio de las soluciones obtenidas por *Von* es entre 6.25 y 7.42 % peores que las de *HCF-$\mathcal{MM}AS$* dependiendo de la instancia, mientras que el incremento en el tiempo de ejecución varía entre 3.45 y 8.02 %. Una situación similar sucede al

Tabla 7.18: Resultados del modelo celular

		Grafo 100-10	**Grafo 75-25**	**Grafo 50-15**
Von	*Promedio*	320.26	1061.42	1795.85
	Desv. Est.	3.76	15.48	28.65
	Mejor	309	1016.59	1739.61
	Tiempo Prom.	662.50	560.40	226.91
Moore	*Promedio*	323.27	1065.18	1796.59
	Desv. Est.	2.49	9.08	19.52
	Mejor	318	1047.83	1746.56
	Tiempo Prom.	678.83	571.68	236.66
Von+BL it.	*Promedio*	**291.67**	**850.14**	**1472.41**
	Desv. Est.	2.23	8.90	14.77
	Mejor	288	**827.37**	1449.58
	Tiempo Prom.	1515.65	2357.96	791.27
Moore+BL it.	*Promedio*	**291.67**	853.57	1473.18
	Desv. Est.	2.46	8.63	16.34
	Mejor	**286**	838.34	**1443.14**
	Tiempo Prom.	1532.13	2419.72	808.50
Mejor solución conocida		280	770.53	1353.49

comparar los resultados obtenidos por *Von+BL iterada* con la versión tradicional de testigo. Las soluciones obtenidas por *Von+BL iterada* son entre 3.65 y 6.75 % peores dependiendo de la instancia, con un incremento de entre 22.11 y 33.73 % en el tiempo de ejecución.

El incremento en el tiempo de ejecución es esperable, ya que el modelo propuesto considera una matriz de feromona por hormiga en lugar de una única para toda la colonia. En cuanto al decremento en la calidad de las soluciones obtenidas, el mecanismo de actualización de cada una de las matrices de feromona solamente considera una solución perteneciente a su vecindario. Posiblemente la utilización de algún mecanismo de actualización para las matrices más sofisticado podría mejorar los resultados obtenidos. Por ejemplo, el esquema de depósito de feromona de la variante P-ACO (presentada en la sección 3.8.1) que se caracteriza por calcular el rastro a partir de las mejores soluciones de las últimas iteraciones. Si bien la utilización de un esquema de depósito con esas características podría mejorar la calidad de las soluciones obtenidas, aumentará la complejidad del algoritmo, impactando en el tiempo de ejecución. Otra posibilidad es que las soluciones enviadas por las hormigas puedan sustituir a las mejores soluciones conocidas por las hormigas receptoras para provocar que las mejores soluciones se propaguen directamente entre las vecindades.

En las tablas 7.19 y 7.20 se presentan los resultados obtenidos por la implementación secuencial y paralela de las versiones del modelo celular con vecindad de Von Neumann, detallándose el costo promedio, la desviación estándar, la mejor solución obtenida y el tiempo promedio de las ejecuciones en minutos. En ambas tablas se incluye, a modo de referencia, la mejor solución conocida hasta el momento. La interacción entre las hormigas en la implementación paralela es exactamente la misma que en la implementación secuencial. Los resultados presentados en las tablas 7.19 y 7.20 permiten constatar que no hay variaciones en la calidad de las soluciones.

Tabla 7.19: Resultados de *Von*

# procesos		Grafo 100-10	Grafo 75-25	Grafo 50-15
1	*Promedio*	320.26	1061.42	1795.85
	Desv. Est.	3.76	15.48	28.65
	Mejor	309	1016.59	1739.61
2	*Promedio*	321.60	1064.83	1792.78
	Desv. Est.	4.09	14.75	30.69
	Mejor	310	1036.30	1729.15
3	*Promedio*	321.17	1062.46	1795.00
	Desv. Est.	3.62	12.67	30.87
	Mejor	313	1024.60	1721.52
4	*Promedio*	323.00	1062.83	1798.54
	Desv. Est.	4.17	13.05	22.36
	Mejor	308	1039.32	1752.52
Mejor solución conocida		280	770.53	1353.49

Tabla 7.20: Resultados de *Von+BL iterada*

# procesos		Grafo 100-10	Grafo 75-25	Grafo 50-15
1	*Promedio*	291.67	850.14	1472.41
	Desv. Est.	2.23	8.90	14.77
	Mejor	288	827.37	1449.58
2	*Promedio*	292.44	852.86	1470.06
	Desv. Est.	2.36	8.10	24.15
	Mejor	286	830.75	1393.64
3	*Promedio*	292.44	850.03	1471.49
	Desv. Est.	3.08	10.06	12.4
	Mejor	286	823.00	1443.42
4	*Promedio*	292.00	850.22	1475.61
	Desv. Est.	2.17	9.27	24.27
	Mejor	286	830.29	1413.51
Mejor solución conocida		280	770.53	1353.49

En la tabla 7.21 se presentan medidas del desempeño computacional de la implementación paralela de *Von* en 2, 3 y 4 procesadores, detallándose el tiempo promedio de las ejecuciones en minutos, la desviación estándar, el speedup algorítmico y la eficiencia. La implementación paralela presenta valores de eficiencia muy próximos a uno, incluso siendo mayor que uno en algunos casos. Los valores de speedup y eficiencia de la implementación paralela corresponden a una reducción lineal en el tiempo de ejecución con el aumento de la cantidad de procesadores.

En la tabla 7.22 se presentan medidas del desempeño computacional de la implementación paralela de *Von+BL iterada* en 2, 3 y 4 procesadores, detallándose el tiempo promedio de las ejecuciones en minutos, la desviación estándar, el speedup algorítmico y la eficiencia. La implementación paralela presenta valores de eficiencia cercanos a uno lo que corresponde a un speedup casi lineal, aunque se aprecia que comienza a degradarse

Tabla 7.21: Desempeño computacional de *Von*

# procesos		Grafo 100-10	Grafo 75-25	Grafo 50-15
1	*Tiempo Prom.*	662.50	560.40	226.91
	Desv. Est.	1.95	1.28	1.49
2	*Tiempo Prom.*	331.61	281.69	113.23
	Desv. Est.	3.21	2.27	1.61
	Speedup	1.99	1.99	2.00
	Eficiencia	0.99	0.99	1.00
3	*Tiempo Prom.*	214.42	182.78	77.77
	Desv. Est.	1.57	1.51	0.96
	Speedup	3.08	3.06	2.92
	Eficiencia	1.02	1.02	0.97
4	*Tiempo Prom.*	166.66	141.60	55.67
	Desv. Est.	1.44	1.33	0.35
	Speedup	3.97	3.96	4.08
	Eficiencia	0.99	0.99	1.02

Tabla 7.22: Desempeño computacional de *Von+BL iterada*

# procesos		Grafo 100-10	Grafo 75-25	Grafo 50-15
1	*Tiempo Prom.*	1515.65	2357.96	791.27
	Desv. Est.	9.19	14.22	7.58
2	*Tiempo Prom.*	790.85	1234.95	421.97
	Desv. Est.	11.70	15.34	6.16
	Speedup	1.92	1.91	1.88
	Eficiencia	0.96	0.95	0.94
3	*Tiempo Prom.*	520.88	816.64	299.70
	Desv. Est.	4.22	7.20	2.41
	Speedup	2.91	2.89	2.64
	Eficiencia	0.97	0.96	0.88
4	*Tiempo Prom.*	433.78	692.65	230.34
	Desv. Est.	1.52	4.54	2.05
	Speedup	3.49	3.40	3.44
	Eficiencia	0.87	0.85	0.86

ligeramente con 4 procesadores. Si bien el desempeño de la implementación paralela de *Von+BL iterada* es levemente inferior al de *Von*, es considerado como muy satisfactorio. A pesar del buen desempeño obtenido, la utilización del operador de búsqueda local iterado puede provocar desbalances en el tiempo que le lleva a cada hormiga construir su solución. Cuando estos desbalances se producen entre hormigas que forman parte de una misma vecindad pero que están en distintos procesadores, la implementación fuertemente sincrónica puede provocar que algunas hormigas queden esperando el envío de soluciones de otras hormigas.

La aplicación del modelo celular propuesto a la resolución del GSP no obtuvo los resultados esperados, provocándose una pérdida en la calidad de las soluciones obtenidas

en relación a estructurar la población en la forma tradicional. Es necesario estudiar alternativas que permitan evitar el decremento en la calidad de las soluciones obtenidas. Una alternativa es que la mejor solución construida por una hormiga y enviada a sus vecinas, se comporte como un migrante, es decir que sustituya a la mejor solución de las hormigas receptoras cuando su costo sea inferior. De esta forma se busca que el intercambio de soluciones tenga un efecto directo, complementario al efecto indirecto que se produce en la actualización del rastro de feromona. Otra posible alternativa que puede resultar promisoria es la utilización de un mecanismo de actualización para las matrices de feromona que considere mayor información como el de la variante P-ACO.

El algoritmo ACO propuesto para la resolución del GSP tiene como particularidad que reutiliza componentes. Esta característica no está presente en los problemas que se suelen utilizar para validar las propuestas de algoritmos ACO. Por este motivo, podría resultar más justo evaluar el modelo celular propuesto aplicado a la resolución de otro tipo de problemas, como por ejemplo el TSP.

En lo que concierne al desempeño computacional, la implementación paralela del modelo celular es muy promisoria. Sin embargo, la incorporación del operador de búsqueda local iterado provoca una leve degradación en el desempeño. Modificar la implementación de forma que la comunicación sea asincrónica es una línea de trabajo futuro que puede contribuir a obtener mejoras en el desempeño. Por otro lado, las pruebas fueron realizadas sobre un cluster de solamente cuatro computadoras, por lo que se deben realizar pruebas complementarias sobre un cluster de más equipos para evaluar como escala el desempeño al utilizar una mayor cantidad de procesadores. Finalmente, es interesante hacer un estudio que permita evaluar el impacto del tamaño de la población utilizada en el desempeño computacional de la implementación paralela.

7.4. Conclusiones

En este capítulo se presentó la evaluación experimental de los algoritmos ACO aplicados a la resolución del GSP propuestos en este trabajo. A continuación, se ofrece un análisis de los resultados obtenidos.

En el enfoque basado en aristas, el mecanismo de construcción consiste en la selección e incorporación de aristas hasta satisfacer un requisito dado. Se implementaron siete versiones que difieren en la visibilidad, la función de calidad de la solución para el depósito de feromona y la regla de transición de estados usados. En primer lugar, se realizó la calibración de forma de determinar los parámetros óptimos, constatándose que valores de $\alpha > 1$ obtienen los mejores resultados. Posteriormente, se realizó la evaluación de las siete versiones propuestas, comprobándose que la versión *Alternativo* muestra resultados aceptables con tiempos de ejecución bajos, aunque no es competitiva con CHC2. El resto de las versiones propuestas no obtiene resultados aceptables, estando lejos de las mejores soluciones conocidas. Como consecuencia de la baja calidad de las soluciones obtenidas por las versiones del enfoque orientado a aristas, se decidió no profundizar en esta línea de investigación.

En el enfoque basado en caminos, el mecanismo de construcción consiste en la determinación de los K caminos más cortos entre el par dado de terminales y la elección de uno de ellos. Se implementaron dos versiones que difieren en la variante de ACO utilizada (AS y HCF-$\mathcal{MM}$AS). Primeramente, se realizó la calibración de parámetros, apreciándose que los mejores resultados se obtienen con valores altos de α y β. En las

pruebas sobre el conjunto de instancias de evaluación, ambas versiones obtuvieron resultados altamente satisfactorios, aventajando claramente al enfoque basado en aristas. Los resultados obtenidos son competitivos con el GA propuesto por Nesmachnow et al., incluso en el *Grafo 100-10* siendo superior a CHC2, aunque a costa de un mayor tiempo de cómputo. Estos resultados alentaron a profundizar el análisis del enfoque basado en caminos incorporando el estudio del efecto del tamaño de la población, de la cantidad de caminos considerados para satisfacer un requerimiento de conexión entre dos terminales y de la incorporación de un operador de búsqueda local. Estos estudios fueron realizados sobre la versión *HCF-$\mathcal{MMAS}$*.

El estudio del efecto del tamaño de la población en los resultados, mostró una leve tendencia a la mejora en las soluciones obtenidas al considerar tamaños crecientes de la población con un incremento lineal en el tiempo de ejecución. Es posible trabajar con una población de menor tamaño, logrando una reducción significativa en el tiempo de ejecución prácticamente sin comprometer la calidad de las soluciones obtenidas.

Las conclusiones que pueden extraerse del estudio del efecto de la variación en la cantidad de caminos considerados para satisfacer un requerimiento de conexión entre dos terminales en los resultados son menos terminantes. El incremento en la cantidad de caminos considerados puede provocar pequeñas mejoras en la calidad de las soluciones obtenidas aunque en algún caso se evidencia el estancamiento del algoritmo. Los resultados obtenidos aumentando la cantidad de caminos (dentro de los márgenes de la prueba) no son competitivos con los obtenidos por CHC2. Respecto a la eficiencia computacional, existe una relación lineal entre la cantidad de caminos y el tiempo promedio de ejecución.

Los operadores de búsqueda local estudiados provocaron importantes mejoras en la calidad de las soluciones obtenidas, a costa de un incremento sustancial en el tiempo promedio de ejecución. La versión *BL* supera a CHC2 en la instancia *Grafo 100-10*, mientras que en el resto de las instancias supera al resto de las propuestas formuladas por Nesmachnow et al. excepto CHC2. Por otro lado, los resultados obtenidos por la versión *BL iterada* son excelentes, superando las mejores soluciones conocidas para las instancias *Grafo 100-10* y *Grafo 75-25*. Asimismo, *BL iterada* obtiene soluciones con costo promedio inferior a CHC2, aunque con un tiempo de ejecución muy superior. La reducción en el tamaño de la población permite balancear la calidad de las soluciones y el tiempo de ejecución, pudiendo obtenerse soluciones precisas en un tiempo de ejecución que si bien es mayor que el de CHC2, se considera aceptable.

Posteriormente, se realizaron RTDs para cada una de las instancias con el objetivo de estudiar la evolución en la calidad de las soluciones, mostrando la probabilidad que *BL iterada* obtenga una solución con una cierta precisión en función del número de iteración. Se pudo detectar que en algunas instancias la evolución es lenta, existiendo un cierto nivel de estancamiento. Este inconveniente es provocado por la desaparición y saturación de feromona en las componentes con el paso de las iteraciones. Una alternativa para intentar solucionarlo es incorporar algún mecanismo de reinicialización de los rastros al detectar el estado de estancamiento. Otra posible mejora es aumentar en forma dinámica durante la ejecución del algoritmo la cantidad de caminos considerados para satisfacer un requerimiento de conexión entre dos terminales, diversificando la exploración de posibles soluciones en etapas avanzadas de la ejecución.

A partir de los excelentes resultados obtenidos es posible establecer dos líneas de trabajo futuro. Considerando que los operadores de búsqueda local pueden ser fácilmente

adaptados para ser utilizado por otras metaheurísticas para la resolución del GSP, una posible línea de trabajo es incorporarlos al algoritmo CHC2. La otra línea de trabajo consiste en el estudio de alternativas para la reducción del tiempo de ejecución. Una idea promisoria para lograr esa reducción es la reutilización de los caminos ya calculados cuando se debe resolver más de un requisito para un mismo par de terminales. Esto permitiría disminuir la cantidad de veces que se ejecuta el algoritmo para obtener los K caminos más cortos. Otra idea que podría contribuir a la reducción del tiempo de ejecución, es limitar la aplicación del operador de búsqueda local (fundamentalmente cuando el operador es el iterado) solamente sobre una de las hormigas o sobre un conjunto pequeño de estas. Ambas ideas propuestas podrían causar perdidas en la calidad de las soluciones obtenidas por lo que deben ser validadas en la práctica.

El modelo celular planteado ubica en cada celda una hormiga con su propia matriz de feromona que se actualiza en forma sincrónica a partir de las mejores soluciones construidas por las hormigas de la vecindad. No se consideró necesario realizar una etapa de calibración ya que el procedimiento de construcción de las soluciones de cada hormiga coincidía exactamente con el utilizado por *HCF-$\mathcal{MMAS}$*.

Se evaluaron versiones secuenciales del modelo celular con estructura de vecindad de Von Neumann y de Moore. Los resultados obtenidos no presentaron mayores diferencias entre si. La utilización de un modelo celular en lugar del esquema tradicional, provocó una pérdida de hasta más de un 7 % en la calidad de las soluciones obtenidas y un incremento en el tiempo promedio de ejecución.

Posteriormente, se evaluaron versiones paralelas del modelo celular pudiendo comprobarse que no existen variaciones en la calidad de las soluciones con respecto a las versiones secuenciales. Este resultado era esperable, ya que el mecanismo de interacción entre las hormigas es exactamente la misma en ambos tipos de versiones. Respecto al desempeño computacional de la versión paralela del modelo celular sin operador de búsqueda local, evaluado mediante el speedup y la eficiencia, es posible afirmar que se obtuvo una reducción lineal en el tiempo de ejecución con el aumento de la cantidad de procesadores. Asimismo, la incorporación del operador de búsqueda local iterativo reduce levemente el speedup, apreciándose una cierta degradación en el desempeño. Esta degradación podría estar motivada por la posible pérdida de balance en el tiempo en que cada hormiga construye su solución provocada por el operador y por la implementación fuertemente sincrónica de las versiones paralela.

El modelo celular no obtuvo los resultados esperados al aplicarse a la resolución del GSP, existiendo una perdida en la calidad de las soluciones con relación a estructurar la población en la forma tradicional. Sin embargo, la evaluación puede ser injusta porque el enfoque propuesto para el GSP tiene como particularidad la reutilización de componentes. Para realizar una evaluación definitiva del modelo celular propuesto es conveniente extender su estudio sobre algún problema que no presente dicha particularidad (por ejemplo: el TSP). Una línea de trabajo futuro es el estudio de alternativas que permitan evitar el decremento en la calidad de las soluciones obtenidas. Una idea interesante es que las soluciones enviadas por las hormigas puedan sustituir a las mejores soluciones conocidas por las hormigas receptoras, provocando que las mejores soluciones se propaguen directamente entre las vecindades. Otra idea promisoria es modificar el mecanismo de actualización del rastro de feromona de cada matriz, incorporando mayor información (actualmente solamente se considera la mejor solución perteneciente al vecindario) aunque esto puede provocar un incremento en el tiempo de ejecución del

algoritmo. Un mecanismo de actualización del rastro de feromona que presenta aspectos interesantes es el de la variante P-ACO, que se caracteriza por calcular el rastro a partir de las mejores soluciones de las últimas iteraciones.

El desempeño computacional de la implementación paralela del modelo celular es muy promisorio aunque la escasa disponibilidad de recursos computacionales limitó el estudio realizado. Asimismo, la incorporación del operador de búsqueda local iterado provocó una leve degradación en el desempeño debido a que la implementación paralela es sincrónica. A partir de los resultados obtenidos, se identifican tres líneas de trabajo para abordar en el futuro inmediato. La primera línea de trabajo es modificar la implementación de forma que la comunicación sea asincrónica, permitiendo subsanar la degradación al utilizar el operador de búsqueda local y contribuyendo a un mejor escalado al aumentar la cantidad de recursos computacionales. Una segunda línea de trabajo consiste en realizar pruebas complementarias sobre un cluster más grande, de forma de estudiar la escalabilidad al aumentar los recursos computacionales disponibles. Finalmente, el estudio del impacto del tamaño de la población utilizada en el desempeño computacional de la implementación paralela constituye la tercera línea de trabajo futuro.

Capítulo 8

Conclusiones y trabajo futuro

"No van a cesar todas las preguntas"
Enrique Bunbury, *Bujías para el dolor*

En este capítulo se resumen las conclusiones de este trabajo y las líneas de trabajo futuro. La estructura del capítulo es la siguiente. En la sección 8.1 se presentan las conclusiones de esta tesis. En la sección 8.2 se comentan las líneas de trabajo futuro identificadas durante el proceso de elaboración de este trabajo.

8.1. Conclusiones

Considerando los objetivos planteados para el trabajo de Tesis de Maestría y realizando una evaluación general, es posible concluir que se cumplieron las metas inicialmente fijadas.

Un primer aspecto importante de este trabajo es la realización de un amplio relevamiento de las variantes de ACO propuestas para la resolución de problemas estáticos de optimización combinatoria. El relevamiento cumplió con el objetivo de facilitar el acercamiento a la metaheurística ACO y a sus variantes, a pesar de la dificultad que plantea la proliferación de propuestas. Particularmente, la reseña permitió estudiar en profundidad las variantes más difundidas y utilizadas, así como nuevas propuestas de reciente formulación, logrando una adecuada comprensión de las principales características de cada una de ellas. En esta etapa se pudo comprobar que ACO es una metaheurística consolidada y que ha demostrado ser competitiva con otras técnicas al ser aplicada sobre varios problemas estándares de optimización combinatoria. El relevamiento realizado fue publicado en una versión breve con el título *Ideas recientes en Ant Colony Optimization* en la XXXIV Conferencia Latinoamericana de Informática [134] y en una versión extendida con el título *Ant Colony Optimization* como reporte técnico del InCo-PEDECIBA [132].

Otro aspecto destacable es la realización de un relevamiento de las estrategias de aplicación de las técnicas de alto desempeño a ACO. Este relevamiento permitió detectar la escasez de taxonomías propuestas, focalizando el estudio en la única propuesta relevante. Al analizar esa propuesta fue posible identificar algunas carencias y limitaciones, proponiéndose una extensión de la taxonomía con el fin de corregirlas. El relevamiento también incluye una amplia reseña de trabajos concretos de aplicación de paralelismo a ACO, clasificando los trabajos de acuerdo a la extensión de la taxonomía propuesta. Se constató que existe un desbalance entre los distintos modelos de paralelismo, siendo los

más implementados multicolonia, maestro-esclavo de grano fino y ejecuciones paralelas independientes. Esta etapa permitió comprobar que la aplicación de paralelismo sobre ACO es promisoria, existiendo un buen número de trabajos que muestran buenos resultados en cuanto a calidad de las soluciones y a desempeño computacional. Por otro lado, se pudo identificar una línea de trabajo novedosa para ACO que posteriormente fue explorada como parte de este trabajo, el diseño de un modelo celular y su implementación paralela. El relevamiento realizado de técnicas de alto desempeño y su aplicación a ACO fue publicado con el título *Paralelismo aplicado a Ant Colony Optimization* como reporte técnico del InCo-PEDECIBA [133].

Con respecto al problema de aplicación, el Problema de Steiner Generalizado ha sido poco abordado fuera de nuestro entorno. Si bien existen experiencias exitosas de aplicación de ACO para la resolución de otros problemas de la clase de Steiner y de problemas con fuertes similitudes al GSP, no se han obtenidos buenos resultados al utilizar ACO o EA con un enfoque constructivo de la solución para la resolución del GSP. El desarrollo de una propuesta exitosa con un enfoque constructivo de la solución para la resolución del GSP constituye un aporte relevante del trabajo desarrollado en esta Tesis de Maestría. Los algoritmos diseñados se caracterizan por construir las soluciones procesando iterativamente los requerimientos entre los terminales, evitando la realización de chequeos de factibilidad. Otras características consideradas especialmente en el diseño fueron el fomento de la reutilización de aristas que componen la solución y prevenir la resolución de requisitos de conexión que la solución parcial ya satisface.

En una primera aproximación, el mecanismo de construcción de caminos entre los terminales se basó en la incorporación de aristas hasta completar el camino. Se desarrollaron versiones que diferían en la visibilidad, la función de calidad de la solución para el depósito de feromona y la regla de transición de estados usados. La evaluación de las propuestas mostró resultados aceptables solamente en una de las versiones, discontinuándose el trabajo con esta aproximación. Los resultados obtenidos fueron publicados como parte del artículo *Metaheurísticas basadas en adaptación social para el Problema de Steiner Generalizado* en el VI Congreso Español de Metaheurísticas, Algoritmos Evolutivos y Bioinspirados [130].

Luego, se estudió un segundo mecanismo de construcción de caminos entre los terminales, basado en la determinación de los K caminos más cortos entre el par de terminales. Los resultados obtenidos fueron satisfactorios aunque a costa de un mayor tiempo de cómputo. Estos resultados alentaron a profundizar el análisis, estudiando el efecto del tamaño de la población y de la cantidad de caminos considerados para satisfacer un requerimiento de conexión entre dos terminales.

Posteriormente, se diseñó un mecanismo de búsqueda local iterado con el objetivo de mejorar las soluciones obtenidas por el enfoque basado en caminos, caracterizado por aplicar el procedimiento constructivo sobre el grafo solución, restringiendo de ese modo la búsqueda a un conjunto potencialmente distinto de caminos. La incorporación del mecanismo de búsqueda local iterado provocó mejoras sustanciales en las soluciones obtenidas, incluso superando las mejores soluciones conocidas para dos instancias, pero con un incremento sustancial en el tiempo de ejecución del algoritmo. El análisis de los resultados obtenidos permitió determinar que trabajar con poblaciones de tamaño pequeño puede permitir un buen balance entre la precisión de las soluciones y el tiempo de ejecución del algoritmo. Como parte del análisis se estudió la evolución en la calidad de las soluciones en función del número de iteración, detectándose en algunos casos

cierto nivel de estancamiento.

La última aproximación para la resolución del GSP, consistió en diseñar un modelo celular caracterizado por ubicar en cada celda una hormiga con su propia matriz de feromona. El procedimiento diseñado para actualizar los rastros de feromona de cada matriz considera solamente las mejores soluciones de la iteración construidas por las hormigas de la vecindad de la celda. Se evaluaron versiones secuenciales con un par de estructuras de vecindad distintas. El modelo celular no obtuvo los resultados esperados, existiendo una perdida de hasta un 7 % en la calidad de las soluciones con relación a estructurar la población en la forma tradicional. Por otro lado, la evaluación del desempeño computacional de la implementación paralela del modelo celular se mostró muy promisoria, aunque la escasez de recursos computacionales disponibles limitó el estudio realizado. El estudio mostró que la incorporación del mecanismo de búsqueda local iterado provocó una leve degradación en el desempeño computacional, pudiendo justificarse este hecho en que la implementación paralela desarrollada es sincrónica.

Los resultados mediante el enfoque basado en caminos y el modelo celular serán publicados con el título *Sequential and parallel ACO methods for solving the Generalized Steiner Problem* en el 24th IFIP TC 7 Conference on System Modelling and Optimization [135].

8.2. Trabajo futuro

A continuación se describen las líneas de trabajo que se identificaron como continuadoras de este trabajo de Maestría y que se abordarán en un futuro cercano.

Las instancias disponibles para la evaluación de las propuestas de resolución del GSP eran escasas. La ampliación del conjunto de problemas de prueba es fundamental, porque permitirá realizar un análisis más profundo sobre las bondades de los algoritmos propuestos. El diseño de nuevas instancias que representen una dificultad real para los algoritmos existentes es una tarea compleja. Si bien los algoritmos propuestos no están diseñados específicamente para el STP, una alternativa temporal para paliar la escasez de instancias es completar la evaluación realizando pruebas sobre instancias del STP (el STP es un caso particular del GSP).

Los resultados obtenidos mediante el enfoque basado en caminos con el operador de búsqueda local iterado son excelentes. Sin embargo, el análisis en la evolución en la calidad de las soluciones en función del número de iteración, reveló que por momentos la evolución es lenta y que se produce un cierto nivel de estancamiento. Este inconveniente es provocado por la desaparición y/o la saturación de feromona en las componentes con el paso de las iteraciones. Una línea de trabajo futura constituye el estudio de alternativas a este inconveniente. Dos ideas promisorias son la incorporación de mecanismos de reinicialización de los rastros cuando se detecta el estancamiento y el ajuste dinámico durante la ejecución del algoritmo de la cantidad de caminos considerados entre dos terminales, buscando diversificar la exploración de posibles soluciones en etapas avanzadas de la ejecución.

Otra línea de trabajo futuro para el enfoque basado en caminos con el operador de búsqueda local iterado consiste en el estudio de alternativas para la reducción del tiempo de ejecución. Una idea promisoria, cuando se debe resolver más de un requisito para un mismo par de terminales, es la reutilización de los caminos ya calculados. Esto permitiría disminuir la cantidad de veces que se ejecuta el algoritmo para obtener los

K caminos más cortos. Otra idea que permite la reducción del tiempo de ejecución, es limitar la aplicación del operador de búsqueda local iterado sobre una de las hormigas o sobre un conjunto pequeño de estas. Ambas ideas propuestas podrían causar pérdidas en la calidad de las soluciones obtenidas por lo que debe realizarse un estudio empírico de su impacto.

El impacto del operador de búsqueda local iterado en la calidad de las soluciones obtenidas es considerable. La adaptación del operador de forma de poder ser incorporado a otras metaheurísticas para la resolución del GSP debe ser explorada.

El modelo celular no obtuvo los resultados esperados al aplicarse a la resolución del GSP, existiendo una perdida en la calidad de las soluciones con relación a estructurar la población en la forma tradicional. Sin embargo, la evaluación puede ser injusta porque el enfoque propuesto para el GSP tiene como particularidad la reutilización de componentes. Una línea de trabajo futuro es extender el estudio del modelo celular sobre otro tipo de problemas que no presenten dicha particularidad, de forma de realizar una evaluación definitiva de la validez o no del modelo propuesto. Otra línea de trabajo futuro es el estudio de alternativas que permitan evitar el decremento en la calidad de las soluciones obtenidas. Una idea interesante es que las soluciones enviadas por las hormigas puedan sustituir a las mejores solución de las hormigas receptoras, ya que provocaría que las mejores soluciones se propaguen directamente entre las vecindades. Otra idea promisoria es modificar el mecanismo de actualización del rastro de feromona de cada matriz, incorporando mayor información aunque esto puede provocar un incremento en el tiempo de ejecución del algoritmo.

El desempeño computacional de la implementación paralela del modelo celular es muy promisorio aunque la escasa disponibilidad de recursos computacionales limitó el estudio realizado. A partir de los resultados obtenidos, es posible establecer tres líneas de trabajo para abordar en el futuro inmediato. La primera consiste en modificar la implementación de forma que la comunicación sea asincrónica, permitiendo subsanar la degradación al utilizar el operador de búsqueda local y contribuyendo a un mejor escalado al aumentar la cantidad de recursos computacionales. La otra línea de trabajo futuro corresponde a realizar pruebas complementarias sobre un cluster con más equipos, de forma de estudiar la escalabilidad de los resultados obtenidos al aumentar los recursos computacionales disponibles. Finalmente, el estudio del impacto del tamaño de la población utilizada en el desempeño computacional de la implementación paralela constituye la tercera línea de trabajo futuro.

Apéndice A

El problema de los K caminos más cortos

"Cuando el objetivo te parezca difícil, no cambies de objetivo;
busca un nuevo camino para llegar a él."
Confucio

El problema de los K caminos más cortos o del K-ésimo camino más corto (K Shortest Path Problem o K-th Shortest Path Problem, KSP) es una generalización del problema clásico del camino más corto (Shortest Path Problem). Consiste en determinar los K caminos más cortos, es decir de menor costo, que conectan un vértice origen y un vértice destino en un grafo.

Existen diversas formulaciones del problema que pueden ser clasificadas según si consideran restricciones sobre los caminos o si no lo hacen. Por las características particulares del GSP resulta interesante incorporar como restricción la utilización de caminos que no tengan ciclos.

La estructura de este apéndice es la siguiente. En la sección A.1 se presenta la formulación del problema de los K caminos más cortos. Posteriormente, en la sección A.2 se presenta un relevamiento de los principales algoritmos que abordan este problema sin considerar restricciones sobre los caminos y considerando caminos sin ciclos.

A.1. Formalización del problema

Sea el grafo $G = (V, E)$ donde V es un conjunto finito de n nodos o vértices ($V = \{v_1, \ldots, v_n\}$) y E es un conjunto finito de m aristas o arcos que conectan los nodos ($E = \{e_1, \ldots, e_m\}$). Cada arco puede ser notado como (i, j), ya que se corresponde con el par de vértices i y j. En general, para los problemas de esta familia se considera a los pares (i, j) como ordenados. El vértice i se llama cola (tail) y el vértice j cabeza (head).

Sean s y t dos vértices dados de G, se define un camino p de s a t en G como la secuencia alternada de vértices y arcos ($p = \left\{v'_0, e'_1, v'_1, \ldots, v'_{k-1}, e'_k, v'_k\right\}$ con $s = v'_0$ y $t = v'_k$) que cumple que:

- s y $t \notin \left\{v'_1, \ldots, v'_{k-1}\right\}$.
- $e'_i \in E \ \forall i = 1, \ldots, k$ y $v'_i \in V \ \forall i = 1, \ldots, k-1$.
- $e'_i = (v'_{i-1}, v'_i) \in E \ \forall e'_i \in p$.

Se llama al vértice s origen o inicial y al vértice t destino o terminal. Sea el costo $c(a_k)$ o $c_{i,j} \in \Re$ un número real asociado al arco $a_k = (i, j)$. Se define el costo de un camino p como $c(p) = \sum_{(i,j)\in p} c_{i,j} = \sum_{p} c_{i,j}$. Sea $P_{i,j}$ el conjunto de caminos de i a j en G. P se utiliza para notar $P_{s,t}$.

Se define el problema de los K caminos más cortos para un grafo G y los vértices s y t como la determinación del conjunto $P^K = \{p_1, \ldots, p_K\}$ que cumple:

- $P^0 = \emptyset$.
- $c(p_k) \leq c(p_{k+1})\ \forall k \in \{1, \ldots, K-1\}$.
- $c(p_k) \leq c(p)\ \forall p \in P - P^K$.

Un camino es elemental o simple cuando no contiene ciclos, es decir si todos sus vértices son diferentes. La formulación del problema de los K caminos más cortos sin ciclos es exactamente la misma que la del problema de los K caminos más cortos, pero incorporando la restricción que los caminos formados deben ser elementales.

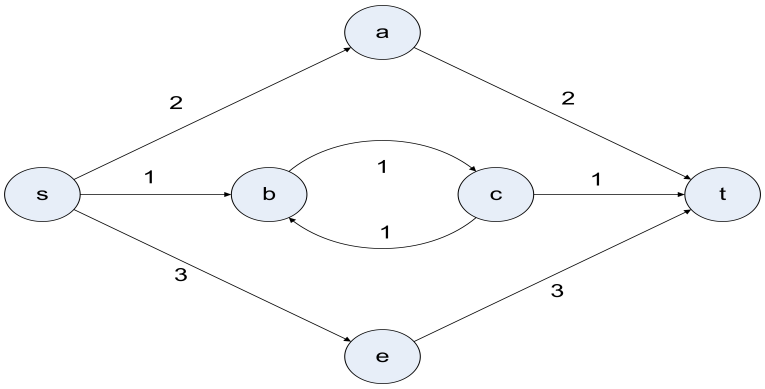

Figura A.1: Ejemplo de las diferencias entre caminos elementales y no elementales

Por ejemplo, en el grafo de la figura A.1, los 3 caminos más cortos entre s y t, sin considerar la restricción de utilizar solamente caminos elementales son:

$$\begin{cases} p_1 = \{s, (s,b), b, (b,c), c, (c,t), t\}, \ c(p_1) = 3 \\ p_2 = \{s, (s,a), a, (a,t), t\}, \ c(p_2) = 4 \\ p_3 = \{s, (s,b), b, (b,c), c, (c,b), b, (b,c), c, (c,t), t\}, \ c(p_3) = 5 \end{cases}$$

Sin embargo, si solamente se consideran caminos elementales, los 3 caminos más cortos son:

$$\begin{cases} p_1 = \{s, (s,b), b, (b,c), c, (c,t), t\}, \ c(p_1) = 3 \\ p_2 = \{s, (s,a), a, (a,t), t\}, \ c(p_2) = 4 \\ p_3 = \{s, (s,e), e, (e,t), t\}, \ c(p_3) = 6 \end{cases}$$

También existen formulaciones del problema que imponen restricciones entre los caminos, por ejemplo que sean disjuntos en sus arcos (K Shortest Arc-Disjoint Path Problem) o que sean disjuntos en sus nodos (K Shortest Node-Disjoint Path Problem).

A.2. Algoritmos para el KSP

En esta sección se resumen los principales algoritmos existentes para el KSP. La sección se divide en dos partes, en la primera se aborda el KSP sin restricciones y en la segunda el KSP con caminos elementales. Debido a las características particulares del GSP para su resolución resultan más atractivos los algoritmos que consideran únicamente caminos elementales.

A.2.1. Algoritmos para el KSP sin restricciones

Cuando se considera el KSP sin restricciones sobre grafos que no contengan ciclos negativos se cumple el principio de optimalidad de Bellman, es decir que los caminos más cortos están formados por sub-caminos más cortos. Por este motivo es posible diseñar algoritmos que funcionen mediante el etiquetado de caminos.

Los algoritmos de etiquetado pueden utilizar dos mecanismos distintos. El primer mecanismo consiste en calcular cada camino en forma incremental, obteniendo de a un camino hasta completar P^K. El segundo mecanismo consiste en realizar correcciones sobre el conjunto de caminos, obteniendo P^K en forma completa y correcta solamente al finalizar la ejecución del algoritmo. Los algoritmos que funcionan mediante el primer mecanismo suelen ser superiores a los del segundo mecanismo [117]. Los algoritmos de etiquetado requieren incorporar varias etiquetas para manejar la información de los caminos.

El mejor algoritmo conocido para el KSP sin restricciones es el propuesto por Eppstein que tiene orden $O(m+n\log n+K)$ [72]. El algoritmo de Eppstein no utiliza el etiquetado de caminos y se caracteriza por obtener una representación implícita de los caminos, a partir de la cual es posible listar las aristas en tiempo proporcional a la cantidad de aristas de cada camino.

El mecanismo que utiliza el algoritmo de Eppstein para encontrar los caminos más cortos entre s y t consiste en buscar caminos entre s y cualquier otro vértice y posteriormente buscar el camino más corto entre ese vértice y t. Las ideas más importantes sobre las que se sustenta son:

- Costos reducidos: en lugar de los costos se utiliza una función potencial δ sobre las aristas de modo que el costo de los caminos más cortos a t sea 0.
  $\delta(e) = c(e)+c(p_1(head(e),t))-c(p_1(tail(e),t))\ \forall (i,j) = e \in E$, siendo $p_1(head(e),t)$ el camino más corto de j a t y $p_1(tail(e),t)$ el camino más corto de i a t.

- Representación implícita: solamente se consideran los caminos desde s en los que la última arista no forma parte del árbol de caminos más cortos a t. Cada camino que se incorpora difiere de un camino incluido previamente en una arista que no forma parte del árbol de los caminos más cortos. La representación implícita corresponde a una referencia al camino incluido previamente y a la arista incorporada.

Otro algoritmo propuesto para resolver el KSP sin restricciones es el MPS [117]. El algoritmo MPS es una generalización del algoritmo de Yen (que se describe en la subsección A.2.2) que no considera restricciones e incorpora la utilización de costos reducidos como el algoritmo de Eppstein. Sin embargo, el algoritmo MPS presenta una mayor complejidad computacional que el algoritmo de Eppstein.

A.2.2. Algoritmos para el KSP con caminos elementales

En el caso del KSP con caminos elementales no se cumple el principio de optimalidad de Bellman [116]. El algoritmo clásico para abordar esta variante del problema fue propuesto por Yen en 1971 [173]. La complejidad de la versión original del algoritmo de Yen tiene orden $O(Kn^3)$, aunque utilizando estructuras de datos más sofisticadas puede reducirse a orden $O(Kn(m + n \log n))$.

La idea principal del algoritmo de Yen es construir candidatos a ser i-ésimo camino a partir de dos secciones: el principio y el final. El principio se toma de uno de los $i - 1$ caminos más cortos calculados previamente y se calcula un nuevo final hasta el destino utilizando algún algoritmo para hallar el camino más corto.

En el algoritmo 8 se presenta la versión original del algoritmo de Yen [24, 173]. Inicialmente se calcula el camino más corto entre s y t mediante *getShortestPath*, evidentemente ese es el primer camino más corto. Para cada vértice del camino más corto (excepto el destino) se calcula el camino más corto desde ese vértice hasta el destino (t). Para armar el nuevo camino, se concatena el sub-camino contenido en el camino más corto hasta el vértice en el cual se va a producir la ramificación, que se obtiene mediante *getRootPath*, y el camino más corto entre el vértice de la ramificación y t, que se obtiene mediante *getShortestPath*.

Algoritmo 8 Algoritmo de Yen(V,E,s,t,K)

```
numPathsFound = 1
paths[numPathsFound] = getShortestPath(V,E,s,t)
list = empty()
while numPathsFound < K do
   previousPath = paths[numPathsFound]
   for all node v ∈ previousPath and v ≠ t do
      rootPath = getRootPath(previousPath,v)
      rootLastNode = v
      V'= setForbiddenNode(V,rootPath)
      E'= setForbiddenArc(E,e | e entrante en rootLastNode,
      e ∈ paths[i] and rootLastNode es raíz de paths[i],
      i = 1..numPathsFound-1)
      spur = getShortestPath(V',E',rootLastNode,d)
      list = insert(list, rootPath + spur)
   end for
   previousPath = getMinCost(list)
   numPathsFound = numPathsFound + 1
   paths[numPathsFound] = previousPath
end while
return paths
```

Sin embargo, existen vértices y aristas que no deben ser considerados para la construcción de la sección final del camino, ya que provocarían ciclos o llevarían a construir una solución que ya forme parte de los K caminos más cortos. Los vértices prohibidos son los que forman parte del sub-camino contenido en el camino más corto hasta el vértice y se marcan mediante *setForbiddenNode* para evitar la incorporación de ciclos. Las aristas prohibidas son las que forman parte de uno de los K caminos más cortos que ya forman

parte de la solución y que fueron usadas en ese camino en forma saliente del vértice en el cual se produjo la ramificación. Esas aristas se marcan mediante $setForbiddenArc$ para evitar la generación de un camino repetido.

Si se logra formar un nuevo camino, se agrega a la lista de caminos construidos ($list$). Cuando se calcularon todos los caminos que es posible armar a partir del camino original y del mecanismo descrito, se selecciona el menor elemento de la lista. Ese elemento es el siguiente camino más corto de de los K y se utiliza para repetir el proceso de construcción. El proceso se repite hasta que se construyan los K caminos más cortos.

Varias mejoras pueden ser realizadas sobre el algoritmo original de Yen. La mejora más simple y eficaz que se puede realizar sobre el algoritmo es sustituir la lista en la que se almacenan los caminos construidos por un montículo. Otra posible mejora consiste en evitar el cálculo de caminos que ya fueron construidos previamente [24].

Existe una variante del algoritmo MPS que considera caminos elementales y que utiliza un mecanismo de funcionamiento similar al del algoritmo de Yen, pero incorporando la utilización de los costos reducidos de Eppstein [116].

Otra propuesta que merece destaque es la de Hoffman [24]. El algoritmo de Hoffman logra reducir la etapa de construcción de cada camino a orden $O(n^2)$ mediante el cálculo inicial del árbol de los caminos más cortos a t. La etapa de construcción de cada camino es similar a la del algoritmo de Yen, pero se utilizan tres secciones. La primera sección es igual a la del algoritmo de Yen. La segunda sección corresponde a la arista en la cual se provoca la ramificación, que debe ser distinta a la utilizada por el camino que se está partiendo, ya que si no se obtendría el mismo camino. Finalmente, la tercera sección corresponde al camino desde el vértice que tiene como arista entrante a la arista de ramificación y t, y se toma directamente del árbol de los caminos más cortos a t. La posible mejora sobre la complejidad total del algoritmo de Yen es difícil de cuantificar debido a que se requiere controlar explícitamente que no se incorporen ciclos. El riesgo de incorporar ciclos es causado por la concatenación de las tres secciones que son resueltas en forma independiente. Esta dificultad depende fuertemente del grafo considerado, con lo cual la mejora en la complejidad no resulta cuantificable.

Una propuesta más moderna y que utiliza un enfoque significativamente distinta es la formulada por Hershberger et al. [96]. La idea de la propuesta de Hershberger es partir el conjunto de todos los caminos candidatos posibles en clases de equivalencia según la estructura a partir de la cual se ramificaron. El algoritmo utiliza técnicas vinculadas al reemplazo de caminos y presenta una gran complejidad conceptual. Si bien se logra mejorar la complejidad del algoritmo de Yen, la propuesta de Hershberger falla en algunas topologías particulares. Existen mecanismos que permiten detectar previamente los casos de falla y ejecutar un algoritmo de mayor orden de complejidad pero con resultado garantizado.

Apéndice B

Soluciones mejoradas

"Que cada uno aporte lo que sepa"
Héroes del Silencio, *Iberia sumergida*

En este apéndice se presentan las soluciones obtenidas mediante la ejecución de los algoritmos ACO planteados en este trabajo que son superiores a las mejores soluciones reportadas previamente. La estructura del apéndice es la siguiente. En la secciones B.1 y B.2 se comentan las mejor soluciones obtenidas para el *Grafo 100-10* y para el *Grafo 75-25*, respectivamente.

B.1. Grafo 100-10

La mejor solución reportada previamente para el *Grafo 100-10* por Nesmachnow et al. tiene costo 291 [129]. Los algoritmos presentados en este trabajo lograron obtener soluciones de costo 290, 289, 288, 287, 286, 285, 284, 283, 282, 281 y 280. En la tabla B.1 se presenta la mejor solución encontrada para el *Grafo 100-10*.

En la figura B.1 se presenta una representación gráfica de la mejor solución encontrada para el *Grafo 100-10*. Los terminales se indican con cuadrados y los nodos de Steiner mediante círculos. El largo de las aristas no guarda relación a su costo, sino que intenta hacer la ilustración lo más clara posible. Las aristas indicadas en punteado no tienen ninguna particularidad, se dibujan de esa forma por claridad ya que cortan a otras aristas.

A continuación se presenta una lista de caminos posibles en la mejor solución encontrada para *Grafo 100-10* de forma de cumplir con los requisitos de conexión.

- 4 caminos entre los terminales 1 y 0:
  - camino 1: 1_22, 13_22, 7_13, 7_35, 0_35
  - camino 2: 1_30, 14_30, 14_15, 0_15
  - camino 3: 1_33, 33_80, 0_80
  - camino 4: 1_96, 51_96, 8_51, 8_76, 47_76, 0_47
- 5 caminos entre los terminales 1 y 2:
  - camino 1: 1_22, 13_22, 7_13, 7_29, 4_29, 4_40, 40_44, 2_44
  - camino 2: 1_30, 14_30, 6_14, 6_70, 5_70, 5_16, 2_16

Tabla B.1: Mejor solución encontrada para el *Grafo 100-10*

Arista	**Vértice**	**Vértice**	**Costo**	**Arista**	**Vértice**	**Vértice**	**Costo**
0_15	0	15	2	6_69	6	69	9
0_35	0	35	5	6_70	6	70	3
0_37	0	37	6	6_96	6	96	6
0_38	0	38	10	7_13	7	13	4
0_47	0	47	12	7_26	7	26	2
0_80	0	80	2	7_29	7	29	2
1_22	1	22	0	7_35	7	35	4
1_30	1	30	11	7_62	7	62	2
1_33	1	33	1	8_9	8	9	13
1_39	1	39	0	8_51	8	51	5
1_96	1	96	1	8_76	8	76	9
2_8	2	8	5	8_82	8	82	4
2_16	2	16	5	9_31	9	31	3
2_23	2	23	3	9_43	9	43	4
2_42	2	42	10	9_69	9	69	5
2_44	2	44	2	9_81	9	81	1
2_98	2	98	2	10_85	10	85	0
3_7	3	7	6	12_28	12	28	0
3_8	3	8	9	12_64	12	64	2
3_10	3	10	3	13_22	13	22	4
3_37	3	37	8	14_15	14	15	2
3_50	3	50	1	14_30	14	30	1
3_93	3	93	4	23_93	23	93	0
4_25	4	25	5	25_39	25	39	1
4_29	4	29	0	26_43	26	43	2
4_31	4	31	5	33_80	33	80	0
4_40	4	40	3	35_82	35	82	1
4_68	4	68	1	38_58	38	58	1
4_90	4	90	4	40_44	40	44	4
5_16	5	16	12	42_68	42	68	2
5_62	5	62	7	47_76	47	76	4
5_70	5	70	12	50_58	50	58	1
5_90	5	90	0	51_96	51	96	1
5_98	5	98	12	62_85	62	85	1
6_14	6	14	6	64_81	64	81	1
6_28	6	28	1	Costo total:		280	

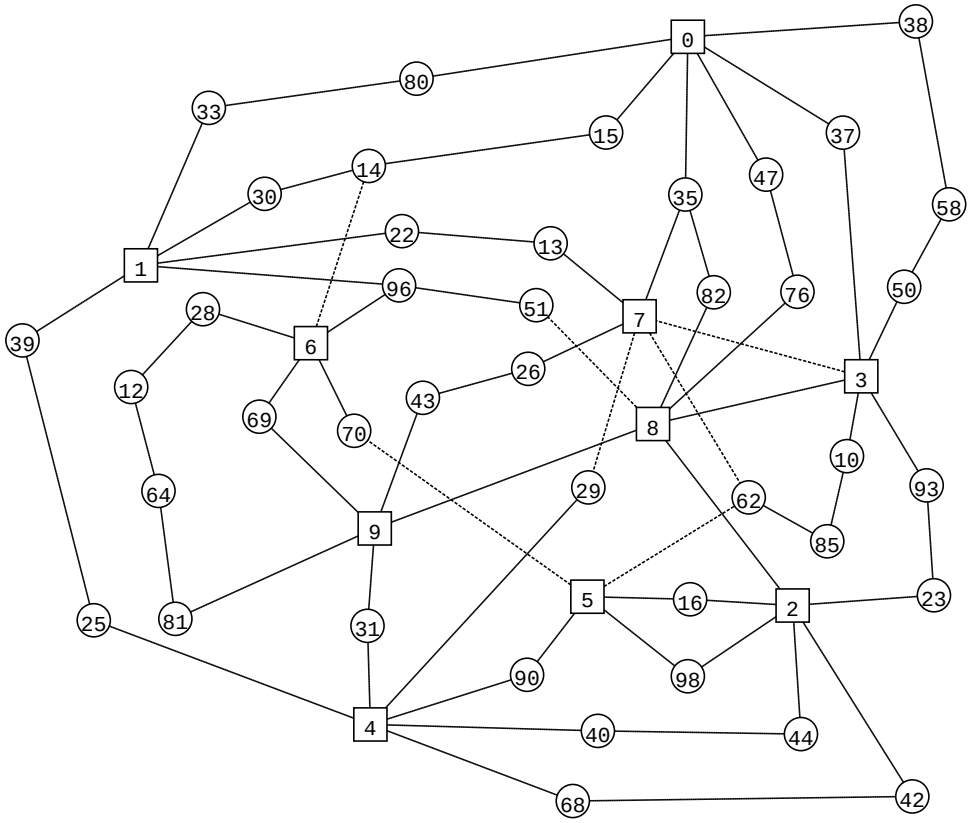

Figura B.1: Mejor solución encontrada para el *Grafo 100-10*

- camino 3: 1_33, 33_80, 0_80, 0_37, 3_37, 3_93, 23_93, 2_23
- camino 4: 1_39, 25_39, 4_25, 4_68, 42_68, 2_42
- camino 5: 1_96, 51_96, 8_51, 2_8

▪ 5 caminos entre los terminales 1 y 3:

- camino 1: 1_22, 13_22, 7_13, 3_7
- camino 2: 1_30, 14_30, 14_15, 0_15, 0_37, 3_37
- camino 3: 1_33, 33_80, 0_80, 0_38, 38_58, 50_58, 3_50
- camino 4: 1_39, 25_39, 4_25, 4_29, 7_29, 7_62, 62_85, 10_85, 3_10
- camino 5: 1_96, 51_96, 8_51, 3_8

▪ 1 camino entre los terminales 1 y 4:

- camino 1: 1_22, 13_22, 7_13, 7_29, 4_29

▪ 5 caminos entre los terminales 1 y 5:

- camino 1: 1_22, 13_22, 7_13, 7_62, 5_62
- camino 2: 1_30, 14_30, 6_14, 6_70, 5_70
- camino 3: 1_33, 33_80, 0_80, 0_37, 3_37, 3_93, 23_93, 2_23, 2_98, 5_98

  - camino 4: 1_39, 25_39, 4_25, 4_90, 5_90
  - camino 5: 1_96, 51_96, 8_51, 2_8, 2_16, 5_16

- 3 caminos entre los terminales 1 y 6:
  - camino 1: 1_22, 13_22, 7_13, 7_62, 5_62, 5_70, 6_70
  - camino 2: 1_30, 14_30, 6_14
  - camino 3: 1_96, 6_96

- 4 caminos entre los terminales 1 y 7:
  - camino 1: 1_22, 13_22, 7_13
  - camino 2: 1_33, 33_80, 0_80, 0_35, 7_35
  - camino 3: 1_39, 25_39, 4_25, 4_29, 7_29
  - camino 4: 1_96, 51_96, 8_51, 8_9, 9_43, 26_43, 7_26

- 1 camino entre los terminales 1 y 8:
  - camino 1: 1_96, 51_96, 8_51

- 4 caminos entre los terminales 1 y 9:
  - camino 1: 1_22, 13_22, 7_13, 7_26, 26_43, 9_43
  - camino 2: 1_30, 14_30, 6_14, 6_69, 9_69
  - camino 3: 1_39, 25_39, 4_25, 4_31, 9_31
  - camino 4: 1_96, 51_96, 8_51, 8_9

- 6 caminos entre los terminales 0 y 2:
  - camino 1: 0_15, 14_15, 6_14, 6_70, 5_70, 5_98, 2_98
  - camino 2: 0_35, 7_35, 7_29, 4_29, 4_40, 40_44, 2_44
  - camino 3: 0_37, 3_37, 3_93, 23_93, 2_23
  - camino 4: 0_38, 38_58, 50_58, 3_50, 3_10, 10_85, 62_85, 5_62, 5_16, 2_16
  - camino 5: 0_47, 47_76, 8_76, 2_8
  - camino 6: 0_80, 33_80, 1_33, 1_39, 25_39, 4_25, 4_68, 42_68, 2_42

- 3 caminos entre los terminales 0 y 3:
  - camino 1: 0_15, 14_15, 6_14, 6_70, 5_70, 5_16, 2_16, 2_8, 3_8
  - camino 2: 0_37, 3_37
  - camino 3: 0_38, 38_58, 50_58, 3_50

- 6 caminos entre los terminales 0 y 4:
  - camino 1: 0_15, 14_15, 14_30, 1_30, 1_39, 25_39, 4_25
  - camino 2: 0_35, 7_35, 7_29, 4_29
  - camino 3: 0_37, 3_37, 3_8, 8_9, 9_31, 4_31

  - camino 4: 0_38, 38_58, 50_58, 3_50, 3_93, 23_93, 2_23, 2_42, 42_68, 4_68
  - camino 5: 0_47, 47_76, 8_76, 2_8, 2_44, 40_44, 4_40
  - camino 6: 0_80, 33_80, 1_33, 1_96, 6_96, 6_70, 5_70, 5_90, 4_90
- 2 caminos entre los terminales 0 y 5:
  - camino 1: 0_15, 14_15, 6_14, 6_70, 5_70
  - camino 2: 0_37, 3_37, 3_93, 23_93, 2_23, 2_98, 5_98
- 3 caminos entre los terminales 0 y 6:
  - camino 1: 0_15, 14_15, 6_14
  - camino 2: 0_37, 3_37, 3_8, 8_9, 9_69, 6_69
  - camino 3: 0_80, 33_80, 1_33, 1_96, 6_96
- 3 caminos entre los terminales 0 y 7:
  - camino 1: 0_35, 7_35
  - camino 2: 0_37, 3_37, 3_7
  - camino 3: 0_80, 33_80, 1_33, 1_22, 13_22, 7_13
- 2 caminos entre los terminales 0 y 8:
  - camino 1: 0_15, 14_15, 6_14, 6_70, 5_70, 5_16, 2_16, 2_8
  - camino 2: 0_37, 3_37, 3_8
- 0 caminos entre los terminales 0 y 9: trivial.
- 6 caminos entre los terminales 2 y 3:
  - camino 1: 2_8, 3_8
  - camino 2: 2_23, 23_93, 3_93
  - camino 3: 2_16, 5_16, 5_62, 62_85, 10_85, 3_10
  - camino 4: 2_42, 42_68, 4_68, 4_29, 7_29, 3_7
  - camino 5: 2_44, 40_44, 4_40, 4_25, 25_39, 1_39, 1_30, 14_30, 14_15, 0_15, 0_38, 38_58, 50_58, 3_50
  - camino 6: 2_98, 5_98, 5_70, 6_70, 6_96, 1_96, 1_33, 33_80, 0_80, 0_37, 3_37
- 1 camino entre los terminales 2 y 4:
  - camino 1: 2_44, 40_44, 4_40
- 2 caminos entre los terminales 2 y 5:
  - camino 1: 2_16, 5_16
  - camino 2: 2_98, 5_98
- 3 caminos entre los terminales 2 y 6:
  - camino 1: 2_8, 8_51, 51_96, 6_96

  - camino 2: 2_16, 5_16, 5_70, 6_70
  - camino 3: 2_23, 23_93, 3_93, 3_37, 0_37, 0_15, 14_15, 6_14
- 5 caminos entre los terminales 2 y 7:
  - camino 1: 2_8, 3_8, 3_7
  - camino 2: 2_23, 23_93, 3_93, 3_10, 10_85, 62_85, 7_62
  - camino 3: 2_42, 42_68, 4_68, 4_29, 7_29
  - camino 4: 2_44, 40_44, 4_40, 4_25, 25_39, 1_39, 1_22, 13_22, 7_13
  - camino 5: 2_98, 5_98, 5_70, 6_70, 6_69, 9_69, 9_43, 26_43, 7_26
- 2 caminos entre los terminales 2 y 8:
  - camino 1: 2_8
  - camino 2: 2_98, 5_98, 5_70, 6_70, 6_69, 9_69, 8_9
- 3 caminos entre los terminales 2 y 9:
  - camino 1: 2_8, 8_9
  - camino 2: 2_23, 23_93, 3_93, 3_10, 10_85, 62_85, 7_62, 7_26, 26_43, 9_43
  - camino 3: 2_98, 5_98, 5_70, 6_70, 6_69, 9_69
- 1 camino entre los terminales 3 y 4:
  - camino 1: 3_10, 10_85, 62_85, 7_62, 7_29, 4_29
- 3 caminos entre los terminales 3 y 5:
  - camino 1: 3_8, 2_8, 2_16, 5_16
  - camino 2: 3_10, 10_85, 62_85, 5_62
  - camino 3: 3_93, 23_93, 2_23, 2_98, 5_98
- 4 caminos entre los terminales 3 y 6:
  - camino 1: 3_8, 8_9, 9_69, 6_69
  - camino 2: 3_10, 10_85, 62_85, 5_62, 5_70, 6_70
  - camino 3: 3_37, 0_37, 0_15, 14_15, 6_14
  - camino 4: 3_50, 50_58, 38_58, 0_38, 0_80, 33_80, 1_33, 1_96, 6_96
- 1 camino entre los terminales 3 y 7:
  - camino 1: 3_7
- 2 caminos entre los terminales 3 y 8:
  - camino 1: 3_8
  - camino 2: 3_93, 23_93, 2_23, 2_8
- 0 caminos entre los terminales 3 y 9: trivial.

- 2 caminos entre los terminales 4 y 5:
  - camino 1: 4_40, 40_44, 2_44, 2_16, 5_16
  - camino 2: 4_90, 5_90
- 4 caminos entre los terminales 4 y 6:
  - camino 1: 4_25, 25_39, 1_39, 1_96, 6_96
  - camino 2: 4_29, 7_29, 3_7, 3_8, 8_9, 9_81, 64_81, 12_64, 12_28, 6_28
  - camino 3: 4_31, 9_31, 9_69, 6_69
  - camino 4: 4_90, 5_90, 5_70, 6_70
- 4 caminos entre los terminales 4 y 7:
  - camino 1: 4_25, 25_39, 1_39, 1_22, 13_22, 7_13
  - camino 2: 4_29, 7_29
  - camino 3: 4_40, 40_44, 2_44, 2_8, 3_8, 3_7
  - camino 4: 4_68, 42_68, 2_42, 2_23, 23_93, 3_93, 3_10, 10_85, 62_85, 7_62
- 4 caminos entre los terminales 4 y 8:
  - camino 1: 4_25, 25_39, 1_39, 1_96, 51_96, 8_51
  - camino 2: 4_29, 7_29, 3_7, 3_8
  - camino 3: 4_31, 9_31, 8_9
  - camino 4: 4_40, 40_44, 2_44, 2_8
- 5 caminos entre los terminales 4 y 9:
  - camino 1: 4_25, 25_39, 1_39, 1_22, 13_22, 7_13, 7_26, 26_43, 9_43
  - camino 2: 4_29, 7_29, 3_7, 3_8, 8_9
  - camino 3: 4_31, 9_31
  - camino 4: 4_40, 40_44, 2_44, 2_8, 8_51, 51_96, 6_96, 6_69, 9_69
  - camino 5: 4_90, 5_90, 5_70, 6_70, 6_28, 12_28, 12_64, 64_81, 9_81
- 2 caminos entre los terminales 5 y 6:
  - camino 1: 5_70, 6_70
  - camino 2: 5_90, 4_90, 4_25, 25_39, 1_39, 1_96, 6_96
- 2 caminos entre los terminales 5 y 7:
  - camino 1: 5_90, 4_90, 4_29, 7_29
  - camino 2: 5_98, 2_98, 2_23, 23_93, 3_93, 3_7
- 5 caminos entre los terminales 5 y 8:
  - camino 1: 5_16, 2_16, 2_8
  - camino 2: 5_62, 7_62, 3_7, 3_8

  - camino 3: 5_70, 6_70, 6_69, 9_69, 8_9
  - camino 4: 5_90, 4_90, 4_29, 7_29, 7_13, 13_22, 1_22, 1_96, 51_96, 8_51
  - camino 5: 5_98, 2_98, 2_23, 23_93, 3_93, 3_37, 0_37, 0_47, 47_76, 8_76

- 1 camino entre los terminales 5 y 9:
  - camino 1: 5_70, 6_70, 6_69, 9_69

- 5 caminos entre los terminales 6 y 7:
  - camino 1: 6_14, 14_15, 0_15, 0_35, 7_35
  - camino 2: 6_28, 12_28, 12_64, 64_81, 9_81, 8_9, 3_8, 3_7
  - camino 3: 6_69, 9_69, 9_43, 26_43, 7_26
  - camino 4: 6_70, 5_70, 5_62, 7_62
  - camino 5: 6_96, 1_96, 1_22, 13_22, 7_13

- 3 caminos entre los terminales 6 y 8:
  - camino 1: 6_14, 14_15, 0_15, 0_37, 3_37, 3_8
  - camino 2: 6_69, 9_69, 8_9
  - camino 3: 6_96, 51_96, 8_51

- 0 caminos entre los terminales 6 y 9: trivial.

- 6 caminos entre los terminales 7 y 8:
  - camino 1: 3_7, 3_8
  - camino 2: 7_13, 13_22, 1_22, 1_96, 51_96, 8_51
  - camino 3: 7_26, 26_43, 9_43, 8_9
  - camino 4: 7_29, 4_29, 4_40, 40_44, 2_44, 2_8
  - camino 5: 7_35, 35_82, 8_82
  - camino 6: 7_62, 62_85, 10_85, 3_10, 3_37, 0_37, 0_47, 47_76, 8_76

- 0 caminos entre los terminales 7 y 9: trivial.

- 5 caminos entre los terminales 8 y 9:
  - camino 1: 2_8, 2_44, 40_44, 4_40, 4_31, 9_31
  - camino 2: 3_8, 3_7, 7_26, 26_43, 9_43
  - camino 3: 8_9
  - camino 4: 8_51, 51_96, 6_96, 6_69, 9_69
  - camino 5: 8_76, 47_76, 0_47, 0_15, 14_15, 6_14, 6_28, 12_28, 12_64, 64_81, 9_81

B.2. Grafo 75-25

La mejor solución reportada previamente para el *Grafo 75-25* por Nesmachnow et al. tiene costo 773.09 (el valor presentado por los autores está redondeado) [129]. Los algoritmos presentados en este trabajo lograron obtener una solución de costo 770.532706. En la tabla B.2 se presenta la mejor solución encontrada para el *Grafo 75-25*.

En la figura B.2 se presenta una representación gráfica de la mejor solución encontrada para el *Grafo 75-25*. Los terminales se indican con cuadrados y los nodos de Steiner mediante círculos. El largo de las aristas no guarda relación a su costo, sino que intenta hacer la ilustración lo más clara posible. Las aristas indicadas en punteado no tienen ninguna particularidad, se dibujan de esa forma por claridad ya que cortan a otras aristas.

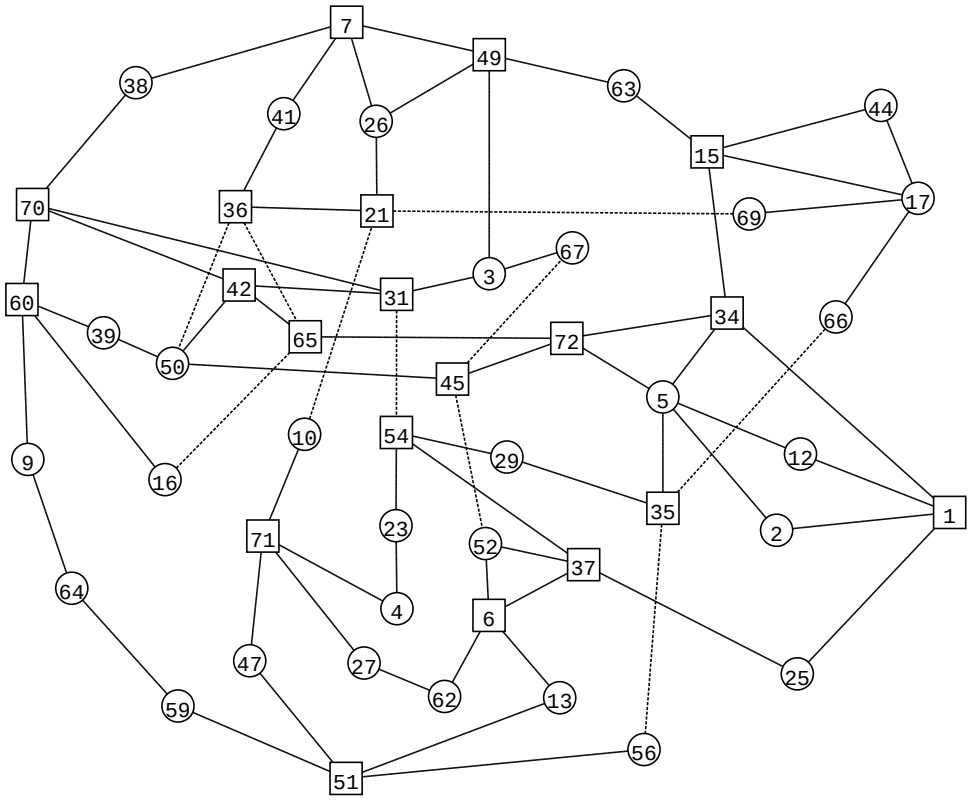

Figura B.2: Mejor solución encontrada para el *Grafo 75-25*

A continuación se presenta una lista de caminos posibles en la mejor solución encontrada para *Grafo 75-25* de forma de cumplir con los requisitos de conexión.

- 3 caminos entre los terminales 6 y 1:
  - camino 1: 6_52, 45_52, 45_72, 5_72, 2_5, 1_2
  - camino 2: 6_37, 25_37, 1_25
  - camino 3: 6_13, 13_51, 47_51, 47_71, 4_71, 4_23, 23_54, 29_54, 29_35, 5_35, 5_12, 1_12
- 3 caminos entre los terminales 7 y 1:

Tabla B.2: Mejor solución encontrada para el Grafo 75-25

Arista	**Vértice**	**Vértice**	**Costo**	**Arista**	**Vértice**	**Vértice**	**Costo**
1_2	1	2	5.46474	21_69	21	69	20.5606
1_12	1	12	2.78969	23_54	23	54	9.57763
1_25	1	25	15.6463	25_37	25	37	11.9029
1_34	1	34	5.93163	26_49	26	49	6.03573
2_5	2	5	1.42433	27_62	27	62	6.28119
3_31	3	31	1.66514	27_71	27	71	9.8053
3_49	3	49	21.4148	29_35	29	35	13.9223
3_67	3	67	3.14284	29_54	29	54	12.8059
4_23	4	23	8.80195	31_42	31	42	8.59697
4_71	4	71	2.73812	31_54	31	54	15.4339
5_12	5	12	2.52724	31_70	31	70	9.94423
5_34	5	34	4.52268	34_72	34	72	16.7352
5_35	5	35	9.49507	35_56	35	56	7.0932
5_72	5	72	14.0034	35_66	35	66	3.35768
6_13	6	13	0.953489	36_41	36	41	3.17189
6_37	6	37	2.85258	36_50	36	50	4.47296
6_52	6	52	0.981392	36_65	36	65	16.6969
6_62	6	62	4.57222	37_52	37	52	2.9922
7_26	7	26	4.76399	37_54	37	54	11.5455
7_38	7	38	8.78917	38_70	38	70	0.331935
7_41	7	41	3.50701	39_50	39	50	5.4852
7_49	7	49	6.32715	39_60	39	60	3.01007
9_60	9	60	14.3498	42_50	42	50	7.73136
9_64	9	64	6.17271	42_65	42	65	19.0381
10_21	10	21	28.8394	42_70	42	70	11.8464
10_71	10	71	8.82777	45_50	45	50	41.2399
13_51	13	51	30.6988	45_52	45	52	7.01528
15_17	15	17	1.76321	45_67	45	67	23.9002
15_34	15	34	15.5922	45_72	45	72	13.0614
15_44	15	44	0.9413	47_51	47	51	4.62825
15_63	15	63	13.4371	47_71	47	71	10.8346
16_60	16	60	6.46598	49_63	49	63	8.67739
16_65	16	65	12.8804	51_56	51	56	34.6797
17_44	17	44	1.09531	51_59	51	59	1.61335
17_66	17	66	6.78169	59_64	59	64	6.25377
17_69	17	69	7.25631	60_70	60	70	20.8823
21_26	21	26	6.16762	65_72	65	72	59.845
21_36	21	36	5.94179	Costo total:		770.532706	

  - camino 1: 7_49, 49_63, 15_63, 15_34, 1_34
  - camino 2: 7_26, 21_26, 21_69, 17_69, 17_66, 35_66, 5_35, 2_5, 1_2
  - camino 3: 7_41, 36_41, 36_50, 45_50, 45_72, 5_72, 5_12, 1_12
- 1 camino entre los terminales 7 y 6:
  - camino 1: 7_26, 21_26, 10_21, 10_71, 27_71, 27_62, 6_62
- 0 caminos entre los terminales 15 y 1: trivial.
- 0 caminos entre los terminales 15 y 6: trivial.
- 2 caminos entre los terminales 15 y 7:
  - camino 1: 15_63, 49_63, 7_49
  - camino 2: 15_17, 17_69, 21_69, 21_26, 7_26
- 3 caminos entre los terminales 21 y 1:
  - camino 1: 21_36, 36_50, 45_50, 45_72, 5_72, 5_12, 1_12
  - camino 2: 21_69, 17_69, 17_66, 35_66, 5_35, 2_5, 1_2
  - camino 3: 10_21, 10_71, 47_71, 47_51, 13_51, 6_13, 6_37, 25_37, 1_25
- 4 caminos entre los terminales 21 y 6:
  - camino 1: 10_21, 10_71, 47_71, 47_51, 13_51, 6_13
  - camino 2: 21_26, 26_49, 3_49, 3_31, 31_54, 37_54, 6_37
  - camino 3: 21_36, 36_50, 45_50, 45_52, 6_52
  - camino 4: 21_69, 17_69, 15_17, 15_34, 5_34, 5_35, 29_35, 29_54, 23_54, 4_23, 4_71, 27_71, 27_62, 6_62
- 0 caminos entre los terminales 21 y 7: trivial.
- 3 caminos entre los terminales 21 y 15:
  - camino 1: 21_69, 17_69, 15_17
  - camino 2: 21_26, 26_49, 49_63, 15_63
  - camino 3: 21_36, 36_65, 65_72, 34_72, 15_34
- 3 caminos entre los terminales 31 y 1:
  - camino 1: 31_42, 42_50, 45_50, 45_72, 5_72, 5_12, 1_12
  - camino 2: 31_54, 37_54, 25_37, 1_25
  - camino 3: 3_31, 3_67, 45_67, 45_52, 37_52, 6_37, 6_13, 13_51, 51_56, 35_56, 5_35, 2_5, 1_2
- 0 caminos entre los terminales 31 y 6: trivial.
- 4 caminos entre los terminales 31 y 7:
  - camino 1: 31_70, 38_70, 7_38

  - camino 2: 31_42, 42_50, 36_50, 36_41, 7_41
  - camino 3: 3_31, 3_49, 7_49
  - camino 4: 31_54, 23_54, 4_23, 4_71, 10_71, 10_21, 21_26, 7_26
- 2 caminos entre los terminales 31 y 15:
  - camino 1: 31_70, 38_70, 7_38, 7_26, 21_26, 21_69, 17_69, 15_17
  - camino 2: 31_42, 42_50, 36_50, 36_41, 7_41, 7_49, 49_63, 15_63
- 0 caminos entre los terminales 31 y 21: trivial.
- 4 caminos entre los terminales 34 y 1:
  - camino 1: 1_34
  - camino 2: 5_34, 2_5, 1_2
  - camino 3: 34_72, 5_72, 5_12, 1_12
  - camino 4: 15_34, 15_17, 17_66, 35_66, 29_35, 29_54, 37_54, 25_37, 1_25
- 3 caminos entre los terminales 34 y 6:
  - camino 1: 15_34, 15_17, 17_69, 21_69, 10_21, 10_71, 27_71, 27_62, 6_62
  - camino 2: 5_34, 5_35, 29_35, 29_54, 37_54, 6_37
  - camino 3: 34_72, 65_72, 36_65, 36_50, 45_50, 45_52, 6_52
- 1 camino entre los terminales 34 y 7:
  - camino 1: 34_72, 65_72, 36_65, 36_41, 7_41
- 3 caminos entre los terminales 34 y 15:
  - camino 1: 15_34
  - camino 2: 5_34, 5_35, 35_66, 17_66, 15_17
  - camino 3: 34_72, 65_72, 36_65, 36_41, 7_41, 7_49, 49_63, 15_63
- 0 caminos entre los terminales 34 y 21: trivial.
- 1 camino entre los terminales 34 y 31:
  - camino 1: 34_72, 65_72, 42_65, 31_42
- 4 caminos entre los terminales 35 y 1:
  - camino 1: 5_35, 2_5, 1_2
  - camino 2: 35_66, 17_66, 15_17, 15_34, 1_34
  - camino 3: 35_56, 51_56, 13_51, 6_13, 6_37, 25_37, 1_25
  - camino 4: 29_35, 29_54, 37_54, 37_52, 45_52, 45_72, 5_72, 5_12, 1_12
- 4 caminos entre los terminales 35 y 6:
  - camino 1: 35_56, 51_56, 13_51, 6_13

  - camino 2: 29_35, 29_54, 37_54, 6_37
  - camino 3: 35_66, 17_66, 17_69, 21_69, 10_21, 10_71, 27_71, 27_62, 6_62
  - camino 4: 5_35, 5_34, 34_72, 65_72, 36_65, 36_50, 45_50, 45_52, 6_52
- 0 caminos entre los terminales 35 y 7: trivial.
- 4 caminos entre los terminales 35 y 15:
  - camino 1: 35_66, 17_66, 15_17
  - camino 2: 5_35, 5_34, 15_34
  - camino 3: 29_35, 29_54, 23_54, 4_23, 4_71, 10_71, 10_21, 21_69, 17_69, 17_44, 15_44
  - camino 4: 35_56, 51_56, 13_51, 6_13, 6_37, 37_54, 31_54, 3_31, 3_49, 49_63, 15_63
- 0 caminos entre los terminales 35 y 21: trivial.
- 2 caminos entre los terminales 35 y 31:
  - camino 1: 35_56, 51_56, 13_51, 6_13, 6_37, 37_54, 31_54
  - camino 2: 5_35, 5_34, 15_34, 15_17, 17_69, 21_69, 21_26, 7_26, 7_38, 38_70, 31_70
- 1 camino entre los terminales 35 y 34:
  - camino 1: 5_35, 5_34
- 1 camino entre los terminales 36 y 1:
  - camino 1: 36_50, 45_50, 45_72, 5_72, 2_5, 1_2
- 3 caminos entre los terminales 36 y 6:
  - camino 1: 36_50, 45_50, 45_52, 6_52
  - camino 2: 21_36, 10_21, 10_71, 27_71, 27_62, 6_62
  - camino 3: 36_65, 65_72, 5_72, 5_35, 29_35, 29_54, 37_54, 6_37
- 2 caminos entre los terminales 36 y 7:
  - camino 1: 36_41, 7_41
  - camino 2: 21_36, 21_26, 7_26
- 2 caminos entre los terminales 36 y 15:
  - camino 1: 36_41, 7_41, 7_49, 49_63, 15_63
  - camino 2: 21_36, 21_69, 17_69, 15_17
- 1 camino entre los terminales 36 y 21:
  - camino 1: 21_36
- 1 camino entre los terminales 36 y 31:

  - camino 1: 36_50, 42_50, 31_42

- 4 caminos entre los terminales 36 y 34:
  - camino 1: 36_65, 65_72, 34_72
  - camino 2: 21_36, 21_69, 17_69, 15_17, 15_34
  - camino 3: 36_50, 45_50, 45_52, 37_52, 25_37, 1_25, 1_34
  - camino 4: 36_41, 7_41, 7_38, 38_70, 31_70, 31_54, 29_54, 29_35, 5_35, 5_34
- 1 camino entre los terminales 36 y 35:
  - camino 1: 36_41, 7_41, 7_38, 38_70, 31_70, 31_54, 29_54, 29_35
- 0 caminos entre los terminales 37 y 1: trivial.
- 2 caminos entre los terminales 37 y 6:
  - camino 1: 6_37
  - camino 2: 37_52, 6_52
- 3 caminos entre los terminales 37 y 7:
  - camino 1: 37_52, 45_52, 45_50, 36_50, 36_41, 7_41
  - camino 2: 37_54, 23_54, 4_23, 4_71, 10_71, 10_21, 21_26, 7_26
  - camino 3: 6_37, 6_62, 27_62, 27_71, 47_71, 47_51, 51_59, 59_64, 9_64, 9_60, 60_70, 38_70, 7_38
- 3 caminos entre los terminales 37 y 15:
  - camino 1: 37_54, 23_54, 4_23, 4_71, 10_71, 10_21, 21_69, 17_69, 17_44, 15_44
  - camino 2: 25_37, 1_25, 1_2, 2_5, 5_34, 15_34
  - camino 3: 6_37, 6_13, 13_51, 51_56, 35_56, 35_66, 17_66, 15_17
- 3 caminos entre los terminales 37 y 21:
  - camino 1: 37_54, 23_54, 4_23, 4_71, 10_71, 10_21
  - camino 2: 37_52, 45_52, 45_50, 36_50, 21_36
  - camino 3: 25_37, 1_25, 1_34, 15_34, 15_17, 17_69, 21_69
- 4 caminos entre los terminales 37 y 31:
  - camino 1: 37_54, 31_54
  - camino 2: 37_52, 45_52, 45_50, 42_50, 31_42
  - camino 3: 6_37, 6_62, 27_62, 27_71, 47_71, 47_51, 51_59, 59_64, 9_64, 9_60, 60_70, 31_70
  - camino 4: 25_37, 1_25, 1_2, 2_5, 5_35, 35_66, 17_66, 15_17, 15_63, 49_63, 3_49, 3_31
- 1 camino entre los terminales 37 y 34:

  - camino 1: 25_37, 1_25, 1_34

- 3 caminos entre los terminales 37 y 35:
  - camino 1: 6_37, 6_13, 13_51, 51_56, 35_56
  - camino 2: 25_37, 1_25, 1_2, 2_5, 5_35
  - camino 3: 37_54, 29_54, 29_35
- 1 camino entre los terminales 37 y 36:
  - camino 1: 37_52, 45_52, 45_50, 36_50
- 3 caminos entre los terminales 42 y 1:
  - camino 1: 42_50, 45_50, 45_72, 5_72, 5_12, 1_12
  - camino 2: 31_42, 31_54, 29_54, 29_35, 5_35, 2_5, 1_2
  - camino 3: 42_70, 38_70, 7_38, 7_26, 21_26, 10_21, 10_71, 27_71, 27_62, 6_62, 6_37, 25_37, 1_25
- 4 caminos entre los terminales 42 y 6:
  - camino 1: 42_50, 45_50, 45_52, 6_52
  - camino 2: 31_42, 31_54, 37_54, 6_37
  - camino 3: 42_70, 38_70, 7_38, 7_26, 21_26, 10_21, 10_71, 27_71, 27_62, 6_62
  - camino 4: 42_65, 65_72, 5_72, 5_35, 29_35, 29_54, 23_54, 4_23, 4_71, 47_71, 47_51, 13_51, 6_13
- 2 caminos entre los terminales 42 y 7:
  - camino 1: 42_70, 38_70, 7_38
  - camino 2: 42_50, 36_50, 36_41, 7_41
- 4 caminos entre los terminales 42 y 15:
  - camino 1: 42_70, 38_70, 7_38, 7_49, 49_63, 15_63
  - camino 2: 42_65, 65_72, 34_72, 15_34
  - camino 3: 42_50, 36_50, 21_36, 21_69, 17_69, 17_44, 15_44
  - camino 4: 31_42, 31_54, 29_54, 29_35, 35_66, 17_66, 15_17
- 4 caminos entre los terminales 42 y 21:
  - camino 1: 31_42, 3_31, 3_49, 26_49, 21_26
  - camino 2: 42_70, 31_70, 31_54, 23_54, 4_23, 4_71, 10_71, 10_21
  - camino 3: 42_65, 65_72, 34_72, 15_34, 15_17, 17_69, 21_69
  - camino 4: 42_50, 36_50, 21_36
- 3 caminos entre los terminales 42 y 31:
  - camino 1: 31_42

  - camino 2: 42_70, 31_70
  - camino 3: 42_50, 45_50, 45_67, 3_67, 3_31
- 1 camino entre los terminales 42 y 34:
  - camino 1: 42_65, 65_72, 34_72
- 3 caminos entre los terminales 42 y 35:
  - camino 1: 42_65, 65_72, 34_72, 5_34, 5_35
  - camino 2: 31_42, 31_54, 29_54, 29_35
  - camino 3: 42_50, 45_50, 45_52, 6_52, 6_13, 13_51, 51_56, 35_56
- 1 camino entre los terminales 42 y 36:
  - camino 1: 42_65, 36_65
- 4 caminos entre los terminales 42 y 37:
  - camino 1: 31_42, 31_54, 37_54
  - camino 2: 42_50, 45_50, 45_52, 37_52
  - camino 3: 42_65, 65_72, 5_72, 2_5, 1_2, 1_25, 25_37
  - camino 4: 42_70, 60_70, 9_60, 9_64, 59_64, 51_59, 47_51, 47_71, 27_71, 27_62, 6_62, 6_37
- 0 caminos entre los terminales 45 y 1: trivial.
- 3 caminos entre los terminales 45 y 6:
  - camino 1: 45_52, 6_52
  - camino 2: 45_72, 5_72, 5_35, 29_35, 29_54, 37_54, 6_37
  - camino 3: 45_50, 36_50, 21_36, 10_21, 10_71, 27_71, 27_62, 6_62
- 3 caminos entre los terminales 45 y 7:
  - camino 1: 45_50, 36_50, 36_41, 7_41
  - camino 2: 45_67, 3_67, 3_49, 7_49
  - camino 3: 45_72, 34_72, 15_34, 15_17, 17_69, 21_69, 21_26, 7_26
- 4 caminos entre los terminales 45 y 15:
  - camino 1: 45_72, 34_72, 15_34
  - camino 2: 45_50, 42_50, 42_70, 38_70, 7_38, 7_49, 49_63, 15_63
  - camino 3: 45_52, 37_52, 37_54, 29_54, 29_35, 35_66, 17_66, 15_17
  - camino 4: 45_67, 3_67, 3_31, 31_54, 23_54, 4_23, 4_71, 10_71, 10_21, 21_69, 17_69, 17_44, 15_44
- 3 caminos entre los terminales 45 y 21:
  - camino 1: 45_50, 36_50, 21_36

  - camino 2: 45_52, 6_52, 6_62, 27_62, 27_71, 10_71, 10_21
  - camino 3: 45_67, 3_67, 3_49, 7_49, 7_26, 21_26

- 3 caminos entre los terminales 45 y 31:
  - camino 1: 45_67, 3_67, 3_31
  - camino 2: 45_50, 42_50, 31_42
  - camino 3: 45_52, 37_52, 37_54, 31_54

- 1 camino entre los terminales 45 y 34:
  - camino 1: 45_72, 34_72

- 4 caminos entre los terminales 45 y 35:
  - camino 1: 45_72, 5_72, 5_35
  - camino 2: 45_52, 37_52, 37_54, 29_54, 29_35
  - camino 3: 45_50, 36_50, 21_36, 21_69, 17_69, 17_66, 35_66
  - camino 4: 45_67, 3_67, 3_31, 31_70, 60_70, 9_60, 9_64, 59_64, 51_59, 51_56, 35_56

- 2 caminos entre los terminales 45 y 36:
  - camino 1: 45_50, 36_50
  - camino 2: 45_67, 3_67, 3_49, 7_49, 7_41, 36_41

- 2 caminos entre los terminales 45 y 37:
  - camino 1: 45_52, 37_52
  - camino 1: 45_72, 5_72, 2_5, 1_2, 1_25, 25_37

- 0 caminos entre los terminales 45 y 42: trivial.

- 0 caminos entre los terminales 49 y 1: trivial.

- 3 caminos entre los terminales 49 y 6:
  - camino 1: 7_49, 7_41, 36_41, 36_50, 45_50, 45_52, 6_52
  - camino 2: 26_49, 21_26, 10_21, 10_71, 27_71, 27_62, 6_62
  - camino 3: 3_49, 3_31, 31_54, 37_54, 6_37

- 2 caminos entre los terminales 49 y 7:
  - camino 1: 7_49
  - camino 2: 26_49, 7_26

- 0 caminos entre los terminales 49 y 15: trivial.

- 3 caminos entre los terminales 49 y 21:
  - camino 1: 26_49, 21_26

  - camino 2: 49_63, 15_63, 15_17, 17_69, 21_69
  - camino 3: 7_49, 7_41, 36_41, 21_36

- 4 caminos entre los terminales 49 y 31:
  - camino 1: 3_49, 3_31
  - camino 2: 7_49, 7_38, 38_70, 31_70
  - camino 3: 26_49, 21_26, 21_36, 36_50, 42_50, 31_42
  - camino 4: 49_63, 15_63, 15_34, 1_34, 1_25, 25_37, 37_54, 31_54
- 3 caminos entre los terminales 49 y 34:
  - camino 1: 49_63, 15_63, 15_34
  - camino 2: 7_49, 7_41, 36_41, 36_65, 65_72, 34_72
  - camino 3: 3_49, 3_31, 31_54, 29_54, 29_35, 5_35, 5_34
- 0 caminos entre los terminales 49 y 35: trivial.
- 0 caminos entre los terminales 49 y 36: trivial.
- 1 camino entre los terminales 49 y 37:
  - camino 1: 49_63, 15_63, 15_34, 1_34, 1_25, 25_37
- 0 caminos entre los terminales 49 y 42: trivial.
- 1 camino entre los terminales 49 y 45:
  - camino 1: 3_49, 3_67, 45_67
- 0 caminos entre los terminales 51 y 1: trivial.
- 0 caminos entre los terminales 51 y 6: trivial.
- 0 caminos entre los terminales 51 y 7: trivial.
- 0 caminos entre los terminales 51 y 15: trivial.
- 3 caminos entre los terminales 51 y 21:
  - camino 1: 47_51, 47_71, 10_71, 10_21
  - camino 2: 51_56, 35_56, 5_35, 5_34, 15_34, 15_17, 17_69, 21_69
  - camino 3: 51_59, 59_64, 9_64, 9_60, 16_60, 16_65, 36_65, 21_36
- 3 caminos entre los terminales 51 y 31:
  - camino 1: 51_59, 59_64, 9_64, 9_60, 60_70, 31_70
  - camino 2: 51_56, 35_56, 29_35, 29_54, 31_54
  - camino 3: 13_51, 6_13, 6_52, 45_52, 45_67, 3_67, 3_31
- 2 caminos entre los terminales 51 y 34:

  - camino 1: 51_56, 35_56, 5_35, 5_34
  - camino 2: 51_59, 59_64, 9_64, 9_60, 16_60, 16_65, 65_72, 34_72

- 0 caminos entre los terminales 51 y 35: trivial.
- 3 caminos entre los terminales 51 y 36:
  - camino 1: 13_51, 6_13, 6_52, 45_52, 45_50, 36_50
  - camino 2: 51_59, 59_64, 9_64, 9_60, 60_70, 38_70, 7_38, 7_41, 36_41
  - camino 3: 47_51, 47_71, 10_71, 10_21, 21_36
- 3 caminos entre los terminales 51 y 37:
  - camino 1: 13_51, 6_13, 6_37
  - camino 2: 47_51, 47_71, 4_71, 4_23, 23_54, 37_54
  - camino 3: 51_59, 59_64, 9_64, 9_60, 16_60, 16_65, 65_72, 5_72, 2_5, 1_2, 1_25, 25_37
- 4 caminos entre los terminales 51 y 42:
  - camino 1: 51_59, 59_64, 9_64, 9_60, 60_70, 42_70
  - camino 2: 13_51, 6_13, 6_37, 37_54, 31_54, 31_42
  - camino 3: 51_56, 35_56, 5_35, 5_72, 65_72, 42_65
  - camino 4: 47_51, 47_71, 10_71, 10_21, 21_36, 36_50, 42_50
- 3 caminos entre los terminales 51 y 45:
  - camino 1: 13_51, 6_13, 6_52, 45_52
  - camino 2: 51_59, 59_64, 9_64, 9_60, 39_60, 39_50, 45_50
  - camino 3: 51_56, 35_56, 5_35, 5_72, 45_72
- 0 caminos entre los terminales 51 y 49: trivial.
- 4 caminos entre los terminales 54 y 1:
  - camino 1: 29_54, 29_35, 5_35, 2_5, 1_2
  - camino 2: 37_54, 25_37, 1_25
  - camino 3: 31_54, 31_42, 42_50, 45_50, 45_72, 5_72, 5_12, 1_12
  - camino 4: 23_54, 4_23, 4_71, 10_71, 10_21, 21_69, 17_69, 15_17, 15_34, 1_34
- 3 caminos entre los terminales 54 y 6:
  - camino 1: 23_54, 4_23, 4_71, 47_71, 47_51, 13_51, 6_13
  - camino 2: 37_54, 6_37
  - camino 3: 31_54, 31_42, 42_65, 36_65, 36_50, 45_50, 45_52, 6_52
- 1 camino entre los terminales 54 y 7:
  - camino 1: 31_54, 31_70, 38_70, 7_38

- 2 caminos entre los terminales 54 y 15:
  - camino 1: 31_54, 3_31, 3_49, 49_63, 15_63
  - camino 2: 29_54, 29_35, 35_66, 17_66, 15_17
- 3 caminos entre los terminales 54 y 21:
  - camino 1: 23_54, 4_23, 4_71, 10_71, 10_21
  - camino 2: 31_54, 3_31, 3_49, 26_49, 21_26
  - camino 3: 29_54, 29_35, 35_66, 17_66, 17_69, 21_69
- 3 caminos entre los terminales 54 y 31:
  - camino 1: 31_54
  - camino 2: 37_54, 37_52, 45_52, 45_67, 3_67, 3_31
  - camino 3: 29_54, 29_35, 35_56, 51_56, 51_59, 59_64, 9_64, 9_60, 60_70, 31_70
- 2 caminos entre los terminales 54 y 34:
  - camino 1: 29_54, 29_35, 5_35, 5_34
  - camino 2: 37_54, 25_37, 1_25, 1_34
- 0 caminos entre los terminales 54 y 35: trivial.
- 4 caminos entre los terminales 54 y 36:
  - camino 1: 31_54, 31_70, 38_70, 7_38, 7_41, 36_41
  - camino 2: 23_54, 4_23, 4_71, 10_71, 10_21, 21_36
  - camino 3: 37_54, 37_52, 45_52, 45_50, 36_50
  - camino 4: 29_54, 29_35, 5_35, 5_34, 34_72, 65_72, 36_65
- 1 camino entre los terminales 54 y 37:
  - camino 1: 37_54
- 3 caminos entre los terminales 54 y 42:
  - camino 1: 31_54, 31_42
  - camino 2: 37_54, 37_52, 45_52, 45_50, 42_50
  - camino 3: 29_54, 29_35, 35_56, 51_56, 51_59, 59_64, 9_64, 9_60, 60_70, 42_70
- 2 caminos entre los terminales 54 y 45:
  - camino 1: 31_54, 3_31, 3_67, 45_67
  - camino 2: 37_54 , 37_52, 45_52
- 4 caminos entre los terminales 54 y 49:
  - camino 1: 31_54, 3_31, 3_49
  - camino 2: 23_54, 4_23, 4_71, 10_71, 10_21, 21_26, 26_49

  - camino 3: 37_54, 25_37, 1_25, 1_34, 15_34, 15_63, 49_63
  - camino 4: 29_54, 29_35, 5_35, 5_34, 34_72, 65_72, 36_65, 36_41, 7_41, 7_49

- 0 caminos entre los terminales 54 y 51: trivial.
- 3 caminos entre los terminales 60 y 1:
  - camino 1: 39_60, 39_50, 45_50, 45_72, 5_72, 5_12, 1_12
  - camino 2: 60_70, 31_70, 31_54, 29_54, 29_35, 5_35, 2_5, 1_2
  - camino 3: 9_60, 9_64, 59_64, 51_59, 13_51, 6_13, 6_37, 25_37, 1_25
- 0 caminos entre los terminales 60 y 6: trivial.
- 0 caminos entre los terminales 60 y 7: trivial.
- 4 caminos entre los terminales 60 y 15:
  - camino 1: 16_60, 16_65, 36_65, 36_41, 7_41, 7_49, 49_63, 15_63
  - camino 2: 9_60, 9_64, 59_64, 51_59, 47_51, 47_71, 10_71, 10_21, 21_69, 17_69, 17_44, 15_44
  - camino 3: 39_60, 39_50, 45_50, 45_52, 37_52, 25_37, 1_25, 1_34, 15_34
  - camino 4: 60_70, 31_70, 31_54, 29_54, 29_35, 35_66, 17_66, 15_17
- 2 caminos entre los terminales 60 y 21:
  - camino 1: 60_70, 38_70, 7_38, 7_41, 36_41, 21_36
  - camino 2: 16_60, 16_65, 42_65, 31_42, 3_31, 3_49, 26_49, 21_26
- 1 camino entre los terminales 60 y 31:
  - camino 1: 60_70, 31_70
- 1 camino entre los terminales 60 y 34:
  - camino 1: 16_60, 16_65, 65_72, 34_72
- 1 camino entre los terminales 60 y 35:
  - camino 1: 60_70, 31_70, 31_54, 29_54, 29_35
- 3 caminos entre los terminales 60 y 36:
  - camino 1: 16_60, 16_65, 36_65
  - camino 2: 39_60, 39_50, 36_50
  - camino 3: 60_70, 38_70, 7_38, 7_41, 36_41
- 4 caminos entre los terminales 60 y 37:
  - camino 1: 39_60, 39_50, 45_50, 45_52, 37_52
  - camino 2: 60_70, 31_70, 31_54, 37_54
  - camino 3: 16_60, 16_65, 65_72, 34_72, 1_34, 1_25, 25_37

    - camino 4: 9_60, 9_64, 59_64, 51_59, 13_51, 6_13, 6_37

- 1 camino entre los terminales 60 y 42:
    - camino 1: 16_60, 16_65, 42_65
- 0 caminos entre los terminales 60 y 45: trivial.
- 1 camino entre los terminales 60 y 49:
    - camino 1: 60_70, 38_70, 7_38, 7_49
- 3 caminos entre los terminales 60 y 51:
    - camino 1: 9_60, 9_64, 59_64, 51_59
    - camino 2: 39_60, 39_50, 45_50, 45_52, 6_52, 6_13, 13_51
    - camino 3: 60_70, 31_70, 31_54, 23_54, 4_23, 4_71, 47_71, 47_51
- 0 caminos entre los terminales 60 y 54: trivial.
- 1 camino entre los terminales 65 y 1:
    - camino 1: 65_72, 5_72, 2_5, 1_2
- 4 caminos entre los terminales 65 y 6:
    - camino 1: 65_72, 45_72, 45_52, 6_52
    - camino 2: 36_65, 21_36, 10_21, 10_71, 27_71, 27_62, 6_62
    - camino 3: 42_65, 31_42, 31_54, 37_54, 6_37
    - camino 4: 16_65, 16_60, 9_60, 9_64, 59_64, 51_59, 13_51, 6_13
- 0 caminos entre los terminales 65 y 7: trivial.
- 0 caminos entre los terminales 65 y 15: trivial.
- 2 caminos entre los terminales 65 y 21:
    - camino 1: 36_65, 21_36
    - camino 2: 42_65, 31_42, 3_31, 3_49, 26_49, 21_26
- 4 caminos entre los terminales 65 y 31:
    - camino 1: 16_65, 16_60, 60_70, 31_70
    - camino 2: 42_65, 31_42
    - camino 3: 36_65, 36_50, 45_50, 45_67, 3_67, 3_31
    - camino 4: 65_72, 34_72, 1_34, 1_25, 25_37, 37_54, 31_54
- 4 caminos entre los terminales 65 y 34:
    - camino 1: 65_72, 34_72
    - camino 2: 16_65, 16_60, 60_70, 31_70, 31_54, 29_54, 29_35, 5_35, 5_34
    - camino 3: 36_65, 21_36, 21_69, 17_69, 15_17, 15_34

  - camino 4: 42_65, 42_50, 45_50, 45_52, 37_52, 25_37, 1_25, 1_34

- 3 caminos entre los terminales 65 y 35:
  - camino 1: 65_72, 34_72, 5_34, 5_35
  - camino 2: 42_65, 31_42, 31_54, 29_54, 29_35
  - camino 3: 36_65, 36_41, 7_41, 7_26, 21_26, 21_69, 17_69, 17_66, 35_66
- 1 camino entre los terminales 65 y 36:
  - camino 1: 36_65
- 4 caminos entre los terminales 65 y 37:
  - camino 1: 65_72, 34_72, 1_34, 1_25, 25_37
  - camino 2: 36_65, 21_36, 10_21, 10_71, 4_71, 4_23, 23_54, 37_54
  - camino 3: 42_65, 42_50, 45_50, 45_52, 37_52
  - camino 4: 16_65, 16_60, 9_60, 9_64, 59_64, 51_59, 13_51, 6_13, 6_37
- 0 caminos entre los terminales 65 y 42: trivial.
- 3 caminos entre los terminales 65 y 45:
  - camino 1: 65_72, 45_72
  - camino 2: 42_65, 31_42, 3_31, 3_67, 45_67
  - camino 3: 36_65, 36_50, 45_50
- 2 caminos entre los terminales 65 y 49:
  - camino 1: 36_65, 36_41, 7_41, 7_49
  - camino 2: 65_72, 34_72, 15_34, 15_63, 49_63
- 3 caminos entre los terminales 65 y 51:
  - camino 1: 16_65, 16_60, 9_60, 9_64, 59_64, 51_59
  - camino 2: 36_65, 36_50, 45_50, 45_52, 6_52, 6_13, 13_51
  - camino 3: 65_72, 5_72, 5_35, 35_56, 51_56
- 1 camino entre los terminales 65 y 54:
  - camino 1: 16_65, 16_60, 60_70, 31_70, 31_54
- 1 camino entre los terminales 65 y 60:
  - camino 1: 16_65, 16_60
- 1 camino entre los terminales 70 y 1:
  - camino 1: 42_70, 42_50, 45_50, 45_72, 5_72, 2_5, 1_2
- 1 camino entre los terminales 70 y 6:

  - camino 1: 42_70, 42_50, 45_50, 45_52, 6_52

- 4 caminos entre los terminales 70 y 7:
  - camino 1: 38_70, 7_38
  - camino 2: 60_70, 39_60, 39_50, 36_50, 36_41, 7_41
  - camino 3: 31_70, 3_31, 3_49, 7_49
  - camino 4: 42_70, 42_65, 36_65, 21_36, 21_26, 7_26
- 3 caminos entre los terminales 70 y 15:
  - camino 1: 31_70, 3_31, 3_49, 49_63, 15_63
  - camino 2: 42_70, 42_65, 65_72, 34_72, 15_34
  - camino 3: 60_70, 9_60, 9_64, 59_64, 51_59, 51_56, 35_56, 35_66, 17_66, 15_17
- 4 caminos entre los terminales 70 y 21:
  - camino 1: 60_70, 39_60, 39_50, 36_50, 21_36
  - camino 2: 38_70, 7_38, 7_26, 21_26
  - camino 3: 31_70, 31_54, 23_54, 4_23, 4_71, 10_71, 10_21
  - camino 4: 42_70, 42_65, 36_65, 36_41, 7_41, 7_49, 49_63, 15_63, 15_17, 17_69, 21_69
- 3 caminos entre los terminales 70 y 31:
  - camino 1: 31_70
  - camino 2: 42_70, 31_42
  - camino 3: 38_70, 7_38, 7_49, 3_49, 3_31
- 2 caminos entre los terminales 70 y 34:
  - camino 1: 38_70, 7_38, 7_26, 26_49, 49_63, 15_63, 15_34
  - camino 2: 42_70, 42_65, 65_72, 34_72
- 2 caminos entre los terminales 70 y 35:
  - camino 1: 31_70, 31_54, 29_54, 29_35
  - camino 2: 60_70, 16_60, 16_65, 65_72, 34_72, 5_34, 5_35
- 1 camino entre los terminales 70 y 36:
  - camino 1: 38_70, 7_38, 7_41, 36_41
- 1 camino entre los terminales 70 y 37:
  - camino 1: 60_70, 16_60, 16_65, 65_72, 34_72, 1_34, 1_25, 25_37
- 1 camino entre los terminales 70 y 42:
  - camino 1: 42_70

- 1 camino entre los terminales 70 y 45:
  - camino 1: 31_70, 3_31, 3_67, 45_67
- 2 caminos entre los terminales 70 y 49:
  - camino 1: 38_70, 7_38, 7_49
  - camino 2: 31_70, 3_31, 3_49
- 4 caminos entre los terminales 70 y 51:
  - camino 1: 60_70, 9_60, 9_64, 59_64, 51_59
  - camino 2: 31_70, 31_54, 29_54, 29_35, 35_56, 51_56
  - camino 3: 38_70, 7_38, 7_26, 21_26, 10_21, 10_71, 47_71, 47_51
  - camino 4: 42_70, 42_50, 45_50, 45_52, 6_52, 6_13, 13_51
- 4 caminos entre los terminales 70 y 54:
  - camino 1: 31_70, 31_54
  - camino 2: 38_70, 7_38, 7_26, 21_26, 10_21, 10_71, 4_71, 4_23, 23_54
  - camino 3: 42_70, 42_65, 65_72, 34_72, 5_34, 5_35, 29_35, 29_54
  - camino 4: 60_70, 9_60, 9_64, 59_64, 51_59, 13_51, 6_13, 6_37, 37_54
- 0 caminos entre los terminales 70 y 60: trivial.
- 2 caminos entre los terminales 70 y 65:
  - camino 1: 60_70, 16_60, 16_65
  - camino 2: 42_70, 42_65
- 1 caminos entre los terminales 71 y 1:
  - camino 1: 47_71, 47_51, 51_56, 35_56, 5_35, 2_5, 1_2
- 3 caminos entre los terminales 71 y 6:
  - camino 1: 27_71, 27_62, 6_62
  - camino 2: 47_71, 47_51, 13_51, 6_13
  - camino 3: 4_71, 4_23, 23_54, 37_54, 6_37
- 0 caminos entre los terminales 71 y 7: trivial.
- 1 camino entre los terminales 71 y 15:
  - camino 1: 10_71, 10_21, 21_69, 17_69, 15_17
- 1 camino entre los terminales 71 y 21:
  - camino 1: 10_71, 10_21
- 2 caminos entre los terminales 71 y 31:

  - camino 1: 10_71, 10_21, 21_26, 26_49, 3_49, 3_31
  - camino 2: 4_71, 4_23, 23_54, 31_54

- 4 caminos entre los terminales 71 y 34:
  - camino 1: 10_71, 10_21, 21_69, 17_69, 15_17, 15_34
  - camino 2: 47_71, 47_51, 51_56, 35_56, 5_35, 5_34
  - camino 3: 4_71, 4_23, 23_54, 37_54, 25_37, 1_25, 1_34
  - camino 4: 27_71, 27_62, 6_62, 6_37, 37_52, 45_52, 45_72, 34_72
- 2 caminos entre los terminales 71 y 35:
  - camino 1: 10_71, 10_21, 21_69, 17_69, 17_66, 35_66
  - camino 2: 27_71, 27_62, 6_62, 6_37, 37_54, 29_54, 29_35
- 3 caminos entre los terminales 71 y 36:
  - camino 1: 10_71, 10_21, 21_36
  - camino 2: 27_71, 27_62, 6_62, 6_52, 45_52, 45_50, 36_50
  - camino 3: 4_71, 4_23, 23_54, 31_54, 31_70, 38_70, 7_38, 7_41, 36_41
- 0 caminos entre los terminales 71 y 37: trivial.
- 0 caminos entre los terminales 71 y 42: trivial.
- 1 camino entre los terminales 71 y 45:
  - camino 1: 4_71, 4_23, 23_54, 37_54, 37_52, 45_52
- 1 camino entre los terminales 71 y 49:
  - camino 1: 10_71, 10_21, 21_26, 26_49
- 0 caminos entre los terminales 71 y 51: trivial.
- 3 caminos entre los terminales 71 y 54:
  - camino 1: 4_71, 4_23, 23_54
  - camino 2: 27_71, 27_62, 6_62, 6_37, 37_54
  - camino 3: 10_71, 10_21, 21_26, 26_49, 3_49, 3_31, 31_54
- 4 caminos entre los terminales 71 y 60:
  - camino 1: 47_71, 47_51, 51_59, 59_64, 9_64, 9_60
  - camino 2: 27_71, 27_62, 6_62, 6_52, 45_52, 45_50, 39_50, 39_60
  - camino 3: 10_71, 10_21, 21_26, 26_49, 7_49, 7_38, 38_70, 60_70
  - camino 4: 4_71, 4_23, 23_54, 37_54, 25_37, 1_25, 1_2, 2_5, 5_72, 65_72, 16_65, 16_60
- 1 camino entre los terminales 71 y 65:

  - camino 1: 4_71, 4_23, 23_54, 37_54, 25_37, 1_25, 1_2, 2_5, 5_72, 65_72

- 0 caminos entre los terminales 71 y 70: trivial.
- 3 caminos entre los terminales 72 y 1:
  - camino 1: 5_72, 2_5, 1_2
  - camino 2: 34_72, 1_34
  - camino 3: 45_72, 45_52, 37_52, 25_37, 1_25
- 3 caminos entre los terminales 72 y 6:
  - camino 1: 45_72, 45_52, 6_52
  - camino 2: 5_72, 2_5, 1_2, 1_25, 25_37, 6_37
  - camino 3: 34_72, 5_34, 5_35, 29_35, 29_54, 23_54, 4_23, 4_71, 47_71, 47_51, 13_51, 6_13
- 0 caminos entre los terminales 72 y 7: trivial.
- 1 camino entre los terminales 72 y 15:
  - camino 1: 5_72, 5_34, 15_34
- 2 caminos entre los terminales 72 y 21:
  - camino 1: 65_72, 36_65, 21_36
  - camino 2: 5_72, 5_35, 29_35, 29_54, 31_54, 3_31, 3_49, 26_49, 21_26
- 2 caminos entre los terminales 72 y 31:
  - camino 1: 5_72, 5_35, 29_35, 29_54, 31_54
  - camino 2: 65_72, 42_65, 31_42
- 0 caminos entre los terminales 72 y 34: trivial.
- 0 caminos entre los terminales 72 y 35: trivial.
- 0 caminos entre los terminales 72 y 36: trivial.
- 1 camino entre los terminales 72 y 37:
  - camino 1: 5_72, 2_5, 1_2, 1_25, 25_37
- 4 caminos entre los terminales 72 y 42:
  - camino 1: 65_72, 42_65
  - camino 2: 45_72, 45_50, 42_50
  - camino 3: 34_72, 5_34, 5_35, 35_56, 51_56, 51_59, 59_64, 9_64, 9_60, 60_70, 42_70
  - camino 4: 5_72, 2_5, 1_2, 1_25, 25_37, 37_54, 31_54, 31_42
- 0 caminos entre los terminales 72 y 45: trivial.

- 2 caminos entre los terminales 72 y 49:
  - camino 1: 5_72, 5_34, 15_34, 15_63, 49_63
  - camino 2: 45_72, 45_67, 3_67, 3_49
- 0 caminos entre los terminales 72 y 51: trivial.
- 1 camino entre los terminales 72 y 54:
  - camino 1: 5_72, 5_35, 29_35, 29_54
- 2 caminos entre los terminales 72 y 60:
  - camino 1: 65_72, 16_65, 16_60
  - camino 2: 5_72, 5_35, 35_56, 51_56, 51_59, 59_64, 9_64, 9_60
- 3 caminos entre los terminales 72 y 65:
  - camino 1: 65_72
  - camino 2: 45_72, 45_50, 42_50, 42_65
  - camino 3: 5_72, 5_35, 35_56, 51_56, 51_59, 59_64, 9_64, 9_60, 16_60, 16_65
- 2 caminos entre los terminales 72 y 70:
  - camino 1: 65_72, 16_65, 16_60, 60_70
  - camino 2: 45_72, 45_50, 42_50, 42_70
- 2 caminos entre los terminales 72 y 71:
  - camino 1: 65_72, 16_65, 16_60, 9_60, 9_64, 59_64, 51_59, 47_51, 47_71
  - camino 2: 34_72, 15_34, 15_63, 49_63, 3_49, 3_31, 31_54, 23_54, 4_23, 4_71

Bibliografía

[1] *GeoSteiner: Software for Computing Steiner Trees Website.* `http://www.diku.dk/geosteiner/` consultada en marzo de 2009.

[2] *Metaheuristics Network Website.* `http://www.metaheuristics.org/` consultada en marzo de 2009.

[3] *OR-Library Website.* `http://people.brunel.ac.uk/~mastjjb/jeb/info.html` consultada en marzo de 2009.

[4] *Website del Problema de Steiner Generalizado.* `http://www.fing.edu.uy/inco/grupos/cecal/hpc/gsp/` consultada en marzo de 2009.

[5] A. Acan: *An External Memory Implementation in Ant Colony Optimization.* En *Proceedings of ANTS 2004 – Fourth International Workshop on Ant Colony Optimization and Swarm Intelligence*, volumen 3172 de *Lecture Notes in Computer Science*, páginas 73–82. Springer, 2004.

[6] A. Acan: *An External Partial Permutations Memory for Ant Colony Optimization.* En *Fifth International Conference on Evolutionary Computation in Combinatorial Optimization - EvoCOP 2005*, volumen 3448 de *Lecture Notes in Computer Science*, páginas 1–11. Springer, 2005.

[7] M. Afshar y M. Marino: *Application of an Ant Algorithm for Layout Optimization of Tree Networks.* Engineering Optimization, 38(3):353–369, 2006.

[8] E. Alba, G. Leguizamon y G. Ordoñez: *Two Models of Parallel ACO Algorithms for the Minimum Tardy Task Problem.* International Journal of High Perfomance Systems Architecture, 1(1):50–59, 2007.

[9] E. Alba, G. Leguizamón y G. Ordoñez: *Analyzing the Behavior of Parallel Ant Colony Systems for Large Instances of the Task Scheduling Problem.* En *Proceedings of the 19th International Parallel and Distributed Processing Symposium, IPDPS 2005*, página 14. IEEE Computer Society, 2005.

[10] E. Alba y A. J. Nebro: *New Technologies in Parallelism.* En E. Alba (editor): *Parallel Metaheuristics*, páginas 63–78. Wiley, 2005.

[11] M. Aroztegui, S. Arraga y S. Nesmachnow: *Resolucion del Problema de Steiner Generalizado Utilizando un Algoritmo Genético Paralelo.* En *II Congreso Español de Metaheuristicas, Algoritmos Evolutivos y Bioinspirados (MAEB 2003)*, páginas 387–394, 2003.

[12] S. Baluja y R. Caruana: *Removing the Genetics from the Standard Genetic Algorithm.* En *Proceedings of the Twelfth International Conference on Machine Learning (ML-95)*, páginas 38–46. Morgan Kaufmann, 1995.

[13] B. Barán y M. Almirón: *Colonia de Hormigas en un Ambiente Paralelo Asíncrono.* En *XXVIII Conferencia Latinoamericana de Informática - CLEI 2002*, 2002.

[14] A. Baykasoğlu, T. Dereli y I. Sabuncu: *A Multiple Objective Ant Colony Optimization Approach to Assembly Line Balancing Problems.* En *35th International Conference on Computers and Industrial Engineering (CIE35)*, páginas 263–268, 2005.

[15] S. Benker, K. Doerner, R.F. Hartl, G. Kiechle y M. Lucká: *Communication Strategies for Parallel Cooperative Ant Colony Optimization on Clusters and Grids.* En *Complementary Proceedings of PARA'04 Workshop on State-of-the-Art in Scientific Computing*, páginas 3–12, 2005.

[16] G. Bilchev y I.C. Parmee: *The Ant Colony Metaphor for Searching Continuous Design Spaces.* En *Selected Papers from AISB Workshop on Evolutionary Computing*, volumen 993 de *Lecture Notes in Computer Science*, páginas 24–39. Springer, 1995.

[17] M.J. Blesa y C. Blum: *Ant Colony Optimization for the Maximum Edge-Disjoint Paths Problem.* En *Applications of Evolutionary Computing, EvoWorkshops 2004, Proceedings*, volumen 3005 de *Lecture Notes in Computer Science*, páginas 160–169. Springer, 2004.

[18] M.J. Blesa y C. Blum: *A Nature-inspired Algorithm for the Disjoint Paths Problem.* En *Proceedings of the 20th International Parallel and Distributed Processing Symposium, IPDPS 2006*, página 8. IEEE Computer Society, 2006.

[19] M.J. Blesa y C. Blum: *Finding Edge-disjoint Paths in Networks: An Ant Colony Optimization Algorithm.* Journal of Mathematical Modelling and Algorithms, 6(3):361–391, 2007.

[20] C. Blum y M. Dorigo: *The Hyper-Cube Framework for Ant Colony Optimization.* IEEE Transactions on Systems, Man, and Cybernetics - Part B, 34(2):1161–1172, 2004.

[21] C. Blum y A. Roli: *Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison.* ACM Computing Surveys, 35(3):268–308, 2003.

[22] C. Blum, A. Roli y M. Dorigo: *HC-ACO: The Hyper-Cube Framework for Ant Colony Optimization.* En *Proceedings of the Fourth Metaheuristics International Conference (MIC'01)*, páginas 399–403, 2001.

[23] M. Bolondi y M. Bondaza: *Parallelizzazione di un Algoritmo per la Risoluzione del Problema del Comesso Viaggiatore (in Italian).* Tesis de Maestría, Politecnico di Milano, Italy, 1993.

[24] A. Brander y M. Sinclair: *A Comparative Study of k-shortest Path Algorithms.* En *Proceedings of the 11th UK Performance Engineering Workshop for Computer and Telecommunications Systems*, páginas 370–379, 1995.

[25] T. Bui y C. Zrncic: *An Ant-based Algorithm for Finding Degree-constrained Minimum Spanning Tree*. En *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 2006*, páginas 11–18. ACM Press, 2006.

[26] B. Bullnheimer, R.F. Hartl y C. Strauss: *A New Rank Based Version of the Ant System - A Computational Study*. Working Paper 3/97, Institute of Management Science, University of Vienna, Austria, April 1997.

[27] B. Bullnheimer, R.F. Hartl y C. Strauss: *Applying the Ant System to the Vehicle Routing Problem*. En *Proceedings of the Second Metaheuristics International Conference (MIC'97)*, 1997.

[28] B. Bullnheimer, R.F. Hartl y C. Strauss: *A New Rank Based Version of the Ant System: A Computational Study*. Central European Journal for Operations Research and Economics, 7(1):25–38, 1999.

[29] B. Bullnheimer, G. Kotsis y C. Strauss: *Parallelization Strategies for the Ant System*. En R. De Leone, A. Murli, P. M. Pardalos y G. Toraldo (editores): *High Performance Algorithms and Software in Nonlinear Optimization, Kluwer International Series in Applied Optimization*, volumen 24, páginas 263–308. Kluwer Academic Publisher, 1998.

[30] R. Burkard, E. Çela, S. Karisch y F. Rendl: *QAPLIB - A Quadratic Assignment Problem Library Website*. `http://www.opt.math.tu-graz.ac.at/qaplib/` consultada en marzo de 2009.

[31] D. Calegari: *Algoritmos Genéticos Aplicados al Diseño de una Red de Comunicaciones Confiable*. Proyecto de Grado, Instituto de Computación, Facultad de Ingeniería,UDELAR, Uruguay, 2004.

[32] E. Cantú-Paz: *Theory of Parallel Genetic Algorithms*. En E. Alba (editor): *Parallel Metaheuristics*, páginas 423–445. Wiley, 2005.

[33] L. Chen y C. Zhang: *Adaptive Parallel Ant Colony Algorithm*. En *Advances in Natural Computation, First International Conference, ICNC 2005, Proceedings, Part II*, volumen 3611 de *Lecture Notes in Computer Science*, páginas 1239–1249. Springer, 2005.

[34] N. Christofides, A. Mingozzi y P. Toth: *The Vehicle Routing Problem*. En N. Christofides, A. Mingozzi, P. Toth y C. Sandi (editores): *Combinatorial Optimization*, páginas 315–338. Wiley, Chicester, 1979.

[35] D. Chu, M. Till y A. Zomaya: *Parallel Ant Colony Optimization for 3D Protein Structure Prediction using the HP Lattice Model*. En *Proceedings of the 19th International Parallel and Distributed Processing Symposium, IPDPS 2005*, página 193b. IEEE Computer Society, 2005.

[36] S-C. Chu, J.F. Roddick y J-S. Pan: *Ant Colony System with Communication Strategies*. Information Sciences, 167(1-4):63–76, 2004.

[37] S-C. Chu, J.F. Roddick, J-S. Pan y C-J. Su: *Parallel Ant Colony Systems*. En *Proceedings of the 14th International Symposium on Methodologies for Intelligent*

Systems, ISMIS 2003, volumen 2871 de *Lecture Notes in Artificial Intelligence*, páginas 279–284. Springer, 2003.

[38] A. Colorni, M. Dorigo y V. Maniezzo: *Distributed Optimization by Ant Colonies.* En *Proceedings of the First European Conference on Artificial Life (ECAL)*, páginas 134–142. MIT Press, 1991.

[39] O. Cordón, I. Fernández de Viana y F. Herrera: *Analysis of the Best-Worst Ant System and its Variants on the QAP.* En *Proceedings of the Third International Workshop on Ant Algorithms (ANTS'2002)*, volumen 2463 de *Lecture Notes in Computer Science*, páginas 228–234. Springer, 2002.

[40] O. Cordón, I. Fernández de Viana y F. Herrera: *Analysis of the Best-Worst Ant System and its Variants on the TSP.* Mathware & Soft Computing, (9):177–192, 2002.

[41] O. Cordón, I. Fernández de Viana, F. Herrera y L. Moreno: *A New ACO Model Integrating Evolutionary Computation Concepts: The Best-Worst Ant System.* En *From Ant Colonies to Artificial Ants: Proceedings of the Second International Workshop on Ant Algorithms (ANTS'2000)*, páginas 22–29, 2000.

[42] T.G. Crainic y N. Hail: *Parallel Metaheuristics Applications.* En E. Alba (editor): *Parallel Metaheuristics*, páginas 447–494. Wiley, 2005.

[43] M. Craus y L. Rudeanu: *Parallel Framework for Ant-Like Algorithms.* En *3rd International Symposium on Parallel and Distributed Computing (ISPDC 2004), 3rd International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogenous Networks (HeteroPar 2004), Proceedings*, páginas 36–41. IEEE Computer Society, 2004.

[44] M. Craus y L. Rudeanu: *Multi-Level Parallel Framework.* International Scientific Journal of Computing, 4(1), 2005.

[45] P. Crescenzi y V. Kann: *A Compendium of NP Optimization Problems Website.* `http://www.nada.kth.se/~viggo/problemlist/` consultada en marzo de 2009.

[46] V-D. Cung, S. L. Martins, C.C. Ribeiro y C. Roucairol: *Strategies for the Parallel Implementation of Metaheuristics.* En C.C. Ribeiro y P. Hansen (editores): *Essays and Surveys in Metaheuristics*, páginas 263–308. Kluwer Academic Publisher, 2001.

[47] A. Das, P. Arabshahi y A. Gray: *An Ant Colony System Approach for Solving the At-least Version of the Generalized Minimum Spanning Tree Problem.* En *SIS '05: Proceedings of the 2005 IEEE Swarm Intelligence Symposium*, páginas 60–67, 2005.

[48] S. Das, S. Gosavi, W. Hsu y S. Vaze: *An Ant Colony Approach For The Steiner Tree Problem.* En *GECCO '02: Proceedings of the Genetic and Evolutionary Computation Conference*, página 135. Morgan Kaufmann Publishers Inc., 2002.

[49] P. Delisle, M. Gravel, M. Krajecki, C. Gagné y W.L. Price: *Comparing Parallelization of an ACO: Message Passing vs. Shared Memory.* En *Hybrid Metaheuristics,*

Second International Workshop, HM 2005, Proceedings, volumen 3636 de *Lecture Notes in Computer Science*, páginas 1–11. Springer, 2005.

[50] P. Delisle, M. Gravel, M. Krajecki, C. Gagné y W.L. Price: *A Shared Memory Parallel Implementation of Ant Colony Optimization.* En *Proceedings of the 6th Metaheuristics International Conference, MIC'2005*, páginas 257–264, 2005.

[51] G. Di Caro: *Ant Colony Optimization and its Application to Adaptive Routing in Telecommunication Networks.* PhD thesis, Faculé des Sciences Appliquées, Université Libre de Bruxelles, Brussels, Belgium, 2004.

[52] G. Di Caro y M. Dorigo: *AntNet: Distributed Stigmergetic Control for Communications Networks.* Journal of Artificial Intelligence Research, 9:317–365, 1998.

[53] G. Di Caro, F. Ducatelle y L.M. Gambardella: *AntHocNet: an Ant-Based Hybrid Routing Algorithm for Mobile Ad Hoc Networks.* En *Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII)*, volumen 3242 de *Lecture Notes in Computer Science*, páginas 461–470. Springer, 2004.

[54] G. Di Caro, F. Ducatelle y L.M. Gambardella: *Studies of Routing Performance in a City-like Testbed for Mobile Ad Hoc Networks.* Technical Report 07-06, IDSIA, Lugano, Switzerland, March 2006.

[55] G. Di Caro y T. Vasilakos: *Ant-SELA: Ant-Agents and Stochastic Automata Learn Adaptive Routing Tables for QoS Routing in ATM Networks.* En *From Ant Colonies to Artificial Ants: Proceedings of the Second International Workshop on Ant Algorithms (ANTS'2000)*, 2000.

[56] K. Doerner, W. Gutjahr, R.F. Hartl, C. Strauss y C. Stummer: *Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection.* Annals of Operations Research, 131(1–4):79–99, 2004.

[57] K. Doerner, R.F. Hartl, S. Benkner y M. Lucká: *Parallel Cooperative Savings Based Ant Colony Optimization - Multiple Search and Decomposition Approaches.* Parallel Processing Letters, 16(3):351–370, 2006.

[58] K. Doerner, R.F. Hartl, G. Kiechle, M. Lucká y M. Reimann: *Parallel Ant Systems for the Capacitated Vehicle Routing Problem.* En *Evolutionary Computation in Combinatorial Optimization, 4th European Conference, EvoCOP 2004, Proceedings*, volumen 3004 de *Lecture Notes in Computer Science*, páginas 72–83. Springer, 2004.

[59] K. Doerner, R.F. Hartl y M.Lucká: *A Parallel Version of the D-Ant Algorithm for the Vehicle Routing Problem.* En *Proceedings of the International Workshop on Parallel Numerics 2005*, páginas 109–118, 2005.

[60] K. Doerner, R.F. Hartl y M. Reimann: *Are COMPETants More Competent for Problem Solving? - The Case of Full Truckload Transportation.* Central European Journal of Operations Research, 11(2):115–141, 2003.

[61] M. Dorigo: *The Ant Colony Optimization Home Page.* `http://iridia.ulb.ac.be/~mdorigo/ACO/ACO.html` consultada en marzo de 2009.

[62] M. Dorigo: *Optimization, Learning and Natural Algorithms (in Italian).* Tesis de Doctorado, Politecnico di Milano, Italy, 1992.

[63] M. Dorigo y G. Di Caro: *Ant Colony Optimization: A New Meta-Heuristic.* En *Proceedings of Congress on Evolutionary Computation (CEC99)*, páginas 1470–1477. IEEE Press, 1999.

[64] M. Dorigo y G. Di Caro: *The Ant Colony Optimization Meta-Heuristic.* En D. Corne, M. Dorigo y F. Glover (editores): *New Ideas in Optimization*, páginas 11–32. McGraw-Hill, 1999. También disponible como Technical Report IRIDIA/99-1, Université Libre de Bruxelles, Belgium.

[65] M. Dorigo y L.M. Gambardella: *Ant Colony Sytem: A Cooperative Learning Approach to the Travelling Salesman Problem.* Technical Report 1996-5, IRIDIA, Université Libre de Bruxelles, Belgium, 1996.

[66] M. Dorigo y L.M. Gambardella: *Ant Colony Sytem: A Cooperative Learning Approach to the Travelling Salesman Problem.* IEEE Transactions on Evolutionary Computation, 1(1):53–66, 1997.

[67] M. Dorigo, V. Maniezzo y A. Colorni: *The Ant System: Optimization by a Colony of Cooperating Agents.* IEEE Transactions on Systems, Man, and Cybernetics-Part B, 26(1):29–41, 1996.

[68] M. Dorigo y T. Stützle: *Ant Colony Optimization.* MIT Press, Cambridge, MA, USA, 2004.

[69] J. Dréo y P. Siarry: *A New Ant Colony Algorithm Using the Heterarchical Concept Aimed at Optimization of Multiminima Continuous Functions.* En *Proceedings of the Third International Workshop on Ant Algorithms (ANTS'2002)*, volumen 2463 de *Lecture Notes in Computer Science*, páginas 216–221. Springer, 2002.

[70] M. Dror, M. Haouari y J. Chaouachi: *Generalized Spanning Trees.* European Journal of Operational Research, 120(3):583–592, 2000.

[71] I. Ellabib, P.H. Calamai y O. A. Basir: *Exchange Strategies for Multiple Ant Colony System.* Information Sciences, 177(5):1248–1264, 2007.

[72] D. Eppstein: *Finding the k Shortest Paths.* SIAM Journal on Computing, 28(2):652–673, 1998.

[73] H. Esbensen: *Computing Near-optimal Solutions to the Steiner Problem in a Graph Using a Genetic Algorithm.* Networks, 26(4):173–185, 1995.

[74] T. Feo y M. Resende: *Greedy Randomized Adaptive Search Procedures.* Journal of Global Optimization, 6:109–133, 1995.

[75] S. Fidanova: *ACO Algorithm with Additional Reinforcement.* En *Proceedings of the Third International Workshop on Ant Algorithms (ANTS'2002)*, volumen 2463 de *Lecture Notes in Computer Science*, páginas 292–293. Springer, 2002.

[76] M. Flynn: *Some Computer Organizations and Their Effectiveness.* IEEE Transactions on Computers, 21(9):948–960, 1972.

[77] L. Fogel, A. Owens y M. Walsh: *Artificial Intelligence Through Simulated Evolution.* John Wiley and Sons, New York, 1966.

[78] L. Ford y D. Fulkerson: *Flows in Networks.* Princeton University Press, Princeton., 1962.

[79] L.M. Gambardella y M. Dorigo: *Ant-Q: A Reinforcement Learning Approach to the Travelling Salesman Problem.* En *Proceedings of the Twelfth International Conference on Machine Learning (ICML)*, páginas 252–260. Morgan Kaufmann, 1995.

[80] L.M. Gambardella y M. Dorigo: *HAS-SOP: An Hybrid Ant System for the Sequential Ordering Problem.* Technical Report 97-11, IDSIA, Lugano, Switzerland, 1997.

[81] L.M. Gambardella, É. Taillard y M. Dorigo: *Ant Colonies for the Quadratic Assignment Problem.* Journal of the Operational Research Society, 50:167–176, 1999.

[82] C. García-Martínez, O. Cordón y F. Herrera: *An Empirical Analysis of Multiple Objective Ant Colony Optimization Algorithms for the Bi-criteria TSP.* En *Proceedings of the 4th International Workshop on Ant Colony Optimization and Swarm Intelligence*, volumen 3172 de *Lecture Notes in Computer Science*, páginas 61–72. Springer, 2004.

[83] M. Garey y D. Johnson: *Computers and Intractability: A Guide to the Theory of NP–Completeness.* Freeman, San Francisco, 1979.

[84] F. Glover: *Future Paths for Integer Programming and Links to Artificial Intelligence.* Computers & Operations Research, 13(5):533–549, 1986.

[85] O. Gómez y B. Barán: *Omicron ACO.* En *30ma Conferencia Latinoamericana de Informática (CLEI2004)*, páginas 932–939. Sociedad Peruana de Computación, 2004.

[86] D. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[87] J.F. Gomez, H.M. Khodr, P.M. De Oliveira, L. Ocque, J.M. Yusta, R. Villasana y A.J. Urdaneta: *Ant Colony System Algorithm for the Planning of Primary Distribution Circuits.* IEEE Transactions on Power Systems, 19(2):996–1004, 2004.

[88] S. Gosavi, S. Das, S. Vaze, G. Singh y E. Buehler: *Obtaining Subtrees from Graphs: an Ant Colony Approach.* En *SIS '03: Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, páginas 160–166, 2003.

[89] M. Gravel, W.L. Price y C. Gagné: *Scheduling Continuous Casting of Aluminum Using a Multiple Objective Ant Colony Optimization Metaheuristic.* European Journal of Operational Research, 143(1):218–229, 2002.

[90] W. Gropp, E. Lusk y A. Skjellum: *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press, 1999.

[91] M. Gruber, J. Van Hemert y G. Raidl: *Neighborhood Searches for the Bounded Diameter Minimum Spanning Tree Problem Embedded in a VNS, EA, and ACO*. En *Proceedings of the Genetic and Evolutionary Computation Conference - GECCO 2006*, volumen 2, páginas 1187–1194. ACM Press, 2006.

[92] M. Guntsch y M. Middendorf: *A Population Based Approach for ACO*. En *Applications of Evolutionary Computing, Proceedings of EvoWorkshops 2002: EvoCOP, EvoIASP, EvoSTim*, volumen 2279, páginas 71–80. Springer, 2002.

[93] M. Guntsch y M. Middendorf: *Applying Population Based ACO to Dynamic Optimization Problems*. En *Proceedings of the Third International Workshop on Ant Algorithms (ANTS'2002)*, volumen 2463 de *Lecture Notes in Computer Science*, páginas 97–104. Springer, 2002.

[94] A. Hara, T. Ichimura, N. Fujita y T. Takahama: *Effective Diversification of Ant-Based Search Using Colony Fission and Extinction*. En *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, páginas 1028–1035. IEEE Press, 2006.

[95] G. Harik, F. Lobo y D. Goldberg: *The Compact Genetic Algorithm*. IEEE Transactions on Evolutionary Computation, 3(4):287–297, 1999.

[96] J. Hershberger, M. Maxel y S. Suri: *Finding the k Shortest Simple Paths: A New Algorithm and its Implementation*. En *Proceedings of the Fifth Workshop on Algorithm Engineering and Experiments (ALENEX)*, páginas 26–36. SIAM, 2003.

[97] Y. Hu, T. Jing, Z. Feng, X-L. Hong, X-D. Hu y G-Y. Yan: *ACO-Steiner: Ant Colony Optimization Based Rectilinear Steiner Minimal Tree Algorithm*. Journal of Computer Science and Technology, 21(1):147–152, 2006.

[98] I. Iimura, K. Hamaguchi, T. Ito y S. Nakayama: *A Study of Distributed Parallel Processing for Queen Ant Strategy in Ant Colony Optimization*. En *PDCAT '05: Proceedings of the 6th International Conference on Parallel and Distributed Computing Applications and Technologies*, páginas 553–557. IEEE Computer Society, 2005.

[99] M. Islam, P. Thulasiraman y R. Thulasiram: *A Parallel Ant Colony Optimization Algorithm for All-Pair Routing in MANETs*. En *Proceedings of the 17th International Parallel and Distributed Processing Symposium, IPDPS 2003*, página 259a. IEEE Computer Society, 2003.

[100] S. Janson, D. Merkle y M. Middendorf: *Parallel Ant Colony Algorithms*. En E. Alba (editor): *Parallel Metaheuristics*, páginas 171–201. Wiley, 2005.

[101] S. Kaegi y T. White: *Using Local Information to Guide Ant Based Search*. En *Proceedings of the 16th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE 2003)*, volumen 2718 de *Lecture Notes in Computer Science*, páginas 692–701. Springer, 2003.

[102] R.M. Karp: *Complexity of Computer Computations*, capítulo Reducibility Among Combinatorial Problems, páginas 85–103. Plenum Press, 1972.

[103] H. Kawamura, M. Yamamoto, K. Suzuki y A. Ohuchi: *Multiple Ant Colonies Algorithm Based on Colony Level Interactions*. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, E83-A(2):371–379, 2000.

[104] S. Kirkpatrick, C. Gelatt y M. Vecchi: *Optimization by Simulated Annealing*. Science, 220:671–680, 1983.

[105] B. Kopinitsch: *An Ant Colony Optimisation Algorithm for the Bounded Diameter Minimum Spanning Tree Problem*. Tesis de Maestría, Institute of Computer Graphics and Algorithms, Vienna University of Technology, Austria, 2006.

[106] F. Krüger, M. Middendorf y D. Merkle: *Studies on a Parallel Ant System for the BSP Model*. Unpublished manuscript, 1998.

[107] J. Kruskal: *On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem*. Proceedings of the American Mathematical Society, 7:48–50, 1956.

[108] H. Lourenço, O. Martin y T. Stützle: *Iterated Local Search*. En F. Glover y G. Kochenberger (editores): *Handbook of Metaheuristics*, páginas 321–353. Kluwer Academic Publisher, 2002.

[109] G. Luque, E. Alba y B. Dorronsoro: *Parallel Genetic Algorithms*. En E. Alba (editor): *Parallel Metaheuristics*, páginas 105–125. Wiley, 2005.

[110] L. Luyet, S. Varone y N. Zufferey: *An Ant Algorithm for the Steiner Tree Problem in Graphs*. En *Applications of Evolutinary Computing, EvoWorkshops 2007: EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog, Proceedings*, volumen 4448 de *Lecture Notes in Computer Science*, páginas 42–51. Springer, 2007.

[111] M. Manfrin, M. Birattari, T. Stützle y M. Dorigo: *Parallel Ant Colony Optimization for the Traveling Salesman Problem*. En *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, Proceedings*, volumen 4150 de *Lecture Notes in Computer Science*, páginas 224–234. Springer, 2006.

[112] V. Maniezzo: *Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem*. Technical Report CSR 98-1, C. L. in Scienze dellÍnformazione, Università di Bologna, Sede di Cesena, Italy, 1998.

[113] V. Maniezzo y A. Carbonaro: *An ANTS Heuristic for the Frequency Assignment Problem*. Technical Report CSR 98-4, C. L. in Scienze dellÍnformazione, Università di Bologna, Sede di Cesena, Italy, 1998.

[114] V. Maniezzo y A. Carbonaro: *An ANTS Heuristic for the Frequency Assignment Problem*. Future Generation Computer Systems, 16(8):927–935, 2000.

[115] V. Maniezzo, A. Carbonaro, M. Golfarelli y S. Rizzi: *ANTS for Data Warehouse Logical Design*. En *Proceedings of the Fourth Metaheuristics International Conference (MIC'01)*, páginas 249–254, 2001.

[116] E. Martins, M. Pascoal y J. Santos: *The K Shortest Loopless Paths Problem.* Research Report, Centre for Informatics and Systems of the University of Coimbra (CISUC), Portugal, Julio 1998.

[117] E. Martins, M. Pascoal y J. Santos: *The K Shortest Paths Problem.* Research Report, Centre for Informatics and Systems of the University of Coimbra (CISUC), Portugal, Junio 1998.

[118] B. Melián, J. Moreno y J. Moreno: *Metaheuristicas: Una visión global.* Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, (19):7–28, 2003.

[119] R. Michel y M. Middendorf: *An Island Model Based Ant System with Lookahead for the Shortest Supersequence Problem.* En *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, volumen 1498 de *Lecture Notes in Computer Science*, páginas 692–701. Springer, 1998.

[120] M. Middendorf, F. Reischle y H. Schmeck: *Information Exchange in Multi Colony Ant Algorithms.* En *Proceedings of the 15th International Parallel and Distributed Processing Symposium, IPDPS 2000*, volumen 1800 de *Lecture Notes in Computer Science*, páginas 645–652. Springer, 2000.

[121] M. Middendorf, F. Reischle y H. Schmeck: *Multi Colony Ant Algorithms.* Journal of Heuristics, 8:305–320, 2002.

[122] N. Mladenović y P. Hansen: *Variable Neighborhood Search.* Computers & Operations Research, 24(11):1097–1100, 1997.

[123] J. Mocholí, J. Martínez y J. Canós: *A Grid Ant Colony Algorithm for the Orienteering Problem.* En *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, páginas 942–949, 2005.

[124] H. Mühlenbein y G. Paaß: *From Recombination of Genes to the Estimation of Distributions I. Binary Parameters.* En *PPSN IV: Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, páginas 178–187. Springer, 1996.

[125] Y. Nakamichi y T. Arita: *Diversity Control in Ant Colony Optimization.* En *Proceedings of the Inaugural Workshop on Artificial Life (AL'01)*, páginas 69–78, 2001.

[126] Y. Nakamichi y T. Arita: *Diversity Control in Ant Colony Optimization.* Artificial Life and Robotics, 7(4):198–204, 2004.

[127] S. Nesmachnow: *Algoritmos Genéticos Paralelos y su Aplicación al Diseño de Redes de Comunicaciones Confiables.* Tesis de Maestría, PEDECIBA Informática, UDELAR, Uruguay, 2004.

[128] S. Nesmachnow: *Evaluating Simple Metaheuristics for the Generalized Steiner Problem.* Journal of Computer Science & Technology, 5(4), 2005.

[129] S. Nesmachnow, H. Cancela y E. Alba: *Evolutionary Algorithms Applied to Reliable Communication Network Design.* Engineering Optimization, 39:831–855(25), 2007.

[130] S. Nesmachnow y M. Pedemonte: *Metaheurísticas basadas en adaptación social para el Problema de Steiner Generalizado.* En *VI Congreso Español de Metaheuristicas, Algoritmos Evolutivos y Bioinspirados (MAEB 2009)*, 2009.

[131] A. Noé, K. Verbeeck y P. Vrancx: *Multi-type Ant Colony: The Edge Disjoint Paths Problem.* En *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Proceedings*, volumen 3172 de *Lecture Notes in Computer Science*, páginas 202–213. Springer, 2004.

[132] M. Pedemonte: *Ant Colony Optimization.* Reporte Técnico RT 07-11, INCO-PEDECIBA, Uruguay, 2007.

[133] M. Pedemonte: *Paralelismo aplicado a Ant Colony Optimization.* Reporte Técnico RT 08-02, INCO-PEDECIBA, Uruguay, 2008.

[134] M. Pedemonte y H. Cancela: *Ideas recientes en Ant Colony Optimization.* En *XXXIV Conferencia Latinoamericana de Informática (CLEI 2008)*, Santa Fe, Argentina, 2008.

[135] M. Pedemonte y H. Cancela: *Sequential and parallel ACO methods for solving the Generalized Steiner Problem.* En *24th IFIP TC 7 Conference on System Modelling and Optimization*, Buenos Aires, Argentina, 2009. Aceptado para su publicación.

[136] M. Pedemonte y S. Nesmachnow: *Estudio Empírico de Operadores de Cruzamiento en un Algoritmo Genético Aplicado al Problema de Steiner Generalizado.* En *Actas del IX Congreso Argentino de Ciencias de Computación*, 2003.

[137] D. Piriyakumar y P. Levi: *A New Approach to Exploiting Parallelism in Ant Colony Optimization.* En *International Symposium on Micromechatronics and Human Science (MHS)*, páginas 237–243, 2002.

[138] R. Prim: *Shortest Connection Networks and Some Generalizations.* The Bell System Technical Journal, 3:1389–1401, 1957.

[139] M. Rahoual, R. Hadji y V. Bachelet: *Parallel Ant System for the Set Covering Problem.* En *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, volumen 2463 de *Lecture Notes in Computer Science*, páginas 262–267. Springer, 2002.

[140] M. Randall y A. Lewis: *A Parallel Implementation of Ant Colony Optimization.* Journal of Parallel and Distributed Computing, 62(9):1421–1432, 2002.

[141] E. Rappos y E. Hadjiconstantinou: *An Ant Colony Heuristic for the Design of Two-Edge Connected Flow Networks.* En *Ant Colony Optimization and Swarm Intelligence, 4th International Workshop, ANTS 2004, Proceedings*, volumen 3172 de *Lecture Notes in Computer Science*, páginas 270–277. Springer, 2004.

[142] I. Rechenberg: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Fromman-Holzboog, Stuttgart, Germany, 1973.

[143] G. Reinelt: *TSPLIB Website.* `http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/` consultada en marzo de 2009.

[144] F. Robledo: *Diseño Topológico de Redes, Casos de Estudio: The Generalized Steiner Problem y The Steiner 2-Edge-Connected Subgraph Problem.* Tesis de Maestría, PEDECIBA Informática, UDELAR, Uruguay, 2001.

[145] S. Shyu, P. Yin, B. Lin y M. Haouari: *Ant-Tree: an Ant Colony Optimization Approach to the Generalized Minimum Spanning Tree Problem.* Journal of Experimental & Theoretical Artificial Intelligence, 15:103–112(10), 2003.

[146] G. Singh, S. Das, S. Gosavi y S. Pujar: *Ant Colony Algorithms for Steiner Trees: An Application to Routing in Sensor Networks*, páginas 181–206. Idea Group Publishing, 2004.

[147] M. Solomon: *Algorithms for the Vehicle Routing and Scheduling with Time Window Constraints.* Operations Research, (15):254–265, 1987.

[148] T. Stützle: *Parallelization Strategies for Ant Colony Optimization.* En *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, volumen 1498 de *Lecture Notes in Computer Science*, páginas 722–731. Springer, 1998.

[149] T. Stützle y H. Hoos: *Improving the Ant System: A Detailed Report on the $\mathcal{MAX}-\mathcal{MIN}$ Ant System.* Technical Report 96-12, AIDA, Germany, 1996.

[150] T. Stützle y H. Hoos: *$\mathcal{MAX}-\mathcal{MIN}$ Ant System and Local Search for the Traveling Salesman Problem.* En *Proceedings of the Fourth International Conference on Evolutionary Computation (ICEC)*, páginas 308–313. IEEE Press, 1997.

[151] T. Stützle y H. Hoos: *The $\mathcal{MAX}-\mathcal{MIN}$ Ant System and Local Search for Combinatorial Optimization Problems: Towards Adaptive Tools for Global Optimization.* En *Proceedings of the Second Metaheuristics International Conference (MIC'97)*, 1997.

[152] T. Stützle y H. Hoos: *$\mathcal{MAX}-\mathcal{MIN}$ Ant System.* Future Generation Computer Systems, 16(8):889–914, 2000.

[153] T. Stützle y S. Linke: *Experiments with Variants of Ant Algorithms.* Mathware & Soft Computing, 9 (2-3):193–207, 2002.

[154] É. Taillard: *FANT: Fast Ant System.* Technical Report 98-46, IDSIA, Lugano, Switzerland, 1998.

[155] É. Taillard y L.M. Gambardella: *Adaptive Memories for the Quadratic Assignment Problem.* Technical Report 97-87, IDSIA, Lugano, Switzerland, 1997.

[156] E-G. Talbi, O. Roux, C. Fonlupt y D. Robillard: *Parallel Ant Colonies for Combinatorial Optimization Problems.* Volumen 1586 de *Lecture Notes in Computer Science*. Springer, 1999.

[157] E-G. Talbi, O. Roux, C. Fonlupt y D. Robillard: *Parallel Ant Colonies for the Quadratic Assignement Problem.* Future Generation Computer Systems, 17(4):441–449, 2001.

[158] S. Tsutsui: *Ant Colony Optimization for the Continuous Domains with Aggregation Pheromones Metaphor.* En *5th International Conference on Recent Advances in Soft Computing (RASC'04)*, páginas 207–212, 2004.

[159] S. Tsutsui: *cAS: Ant Colony Optimization with Cunning Ants.* En *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference, Proceedings*, volumen 4193 de *Lecture Notes in Computer Science*, páginas 162–171. Springer, 2006.

[160] S. Tsutsui: *Parallel Ant Colony Optimization for the Quadratic Assignment Problem with Symmetric Multi Processing.* Technical Report, MEDAL Report No. 2008006, Missouri Estimation of Distribution Algorithms Laboratory, University of Missouri, St. Louis, MO, 2008.

[161] S. Tsutsui y L. Liu: *Cunning Ant System for Quadratic Assignment Problem with Local Search and Parallelization.* Technical Report, MEDAL Report No. 2007006, Missouri Estimation of Distribution Algorithms Laboratory, University of Missouri, St. Louis, MO, 2007.

[162] S. Tsutsui y M. Pelikan: *cAS: The Cunning Ant System.* Technical Report, MEDAL Report No. 2007007, Missouri Estimation of Distribution Algorithms Laboratory, University of Missouri, St. Louis, MO, 2007.

[163] S. Tsutsui, M. Pelikan y A. Ghosh: *Performance of Aggregation Pheromone System on Unimodal and Multimodal Problems.* En *The IEEE Congress on Evolutionary Computation, 2005 (CEC2005)*, volumen 1, páginas 880–887, 2005.

[164] S. Voß: *Handbook of Optimization in Telecommunications*, capítulo Steiner Tree Problems in Telecommunications, páginas 459–492. Springer, 2006.

[165] C. Voudouris y E. Tsang: *Guided Local Search.* Technical Report CSM-247, Department of Computer Science, University of Essex, UK, 1995.

[166] P. Vrancx y A. Nowé: *Using Pheromone Repulsion to Find Disjoint Paths (Extended Abstract).* En *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, Proceedings*, volumen 4150 de *Lecture Notes in Computer Science*, páginas 522–524. Springer, 2006.

[167] C. Watkins y P. Dayan: *Q-Learning.* Machine Learning, 8(3-4):279–292, 1992.

[168] T. White, S. Kaegi y T. Oda: *Revisiting Elitism in Ant Colony Optimization.* En *Genetic and Evolutionary Computation – GECCO-2003*, volumen 2723 de *Lecture Notes in Computer Science*, páginas 122–133. Springer, 2003.

[169] W. Wiesemann y T. Stützle: *An Experimental Investigation of Iterated Ants for the Quadratic Assignment Problem.* Technical Report 2006-3, IRIDIA, Université Libre de Bruxelles, Belgium, 2006.

[170] P. Winter: *Steiner Problem in Networks: A Survey.* Networks, 17:129–167, 1987.

[171] X-L. Hong Z. Feng X-D. Hu G-Y. Yan Y. Hu, T. Jing: *An Efficient Rectilinear Steiner Minimum Tree Algorithm Based on Ant Colony Optimization.* En *Proceedings of IEEE ICCCAS*, páginas 1276–1280, 2004.

[172] Z. Yang, B. Yu y C. Cheng: *A Parallel Ant Colony Algorithm for Bus Network Optimization*. Computer-Aided Civil and Infrastructure Engineering, 22(1):44–55, 2007.

[173] J. Yen: *Finding the K Shortest Loopless Paths in a Network*. Management Science, 17:712–716, 1971.

[174] M. Zlochin, M. Birattari, N. Meuleau y M. Dorigo: *Model-Based Search for Combinatorial Optimization: A Critical Survey*. Annals of Operations Research, 131:373–395, 2004.

[175] M. Zlochin y M. Dorigo: *Model-Based Search for Combinatorial Optimization: A Comparative Study*. En *PPSN VII: Proceedings of the 7th International Conference on Parallel Problem Solving from Nature*, páginas 651–664. Springer, 2002.