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

Reporte Técnico RT 08-02

Paralelismo aplicado a ant colony optimization

Martín Pedemonte

2008

Paralelismo aplicado a ant colony optimization
Pedemonte, Martín
ISSN 0797-6410
Reporte Técnico RT 08-02
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, 2008

Paralelismo aplicado a Ant Colony Optimization

Martín Pedemonte
Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
mpedemon@fing.edu.uy

Enero de 2008

La utilización de metaheurísticas para la resolución de problemas de optimización combinatoria del tipo NP-difícil ha permitido afrontar instancias grandes obteniendo soluciones cercanas al óptimo en tiempos razonables. En los últimos años la aplicación de paralelismo a las metaheurísticas ha demostrado su utilidad no solamente porque ha permitido disminuir considerablemente los tiempos de ejecución, sino también porque ha permitido obtener mejoras en la calidad de las soluciones encontradas.

Ant Colony Optimization (ACO) es una metaheurística de las más recientes que ha sido aplicada con éxito sobre varios de los problemas estándares de optimización demostrando su potencial. Las primeras propuestas de paralelismo aplicado a ACO se remontan a los orígenes de la propia metaheurística. Sin embargo, la investigación en esta temática ha crecido notablemente en los últimos cinco años.

El presente reporte es un relevamiento sobre la aplicación de técnicas de alto desempeño sobre ACO. El objetivo de este relevamiento es brindar un resumen de las principales propuestas existentes en la literatura sobre esta temática. Solamente se consideran las implementaciones paralelas aplicadas sobre problemas estáticos monobjetivos de optimización combinatoria.

Palabras claves: Ant Colony Optimization, Paralelismo, Metaheurísticas.

1. Introducción

La resolución de problemas de optimización combinatoria del tipo NP-difícil mediante la aplicación de metaheurísticas puede presentar dificultades en la práctica. La utilización de metaheurísticas permite reducir significativamente el tiempo de ejecución asegurando la obtención de soluciones cercanas al óptimo, aunque existen condiciones ante las cuales puede resultar impracticable. Algunos factores que pueden incidir en un aumento significativo en el tiempo de cómputo necesario para su ejecución son la utilización de instancias de una alta dimensionalidad o trabajar sobre problemas fuertemente restringidos. En ese contexto 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 considerable en su tiempo de ejecución.

Podría pensarse que la disminución del tiempo de ejecución es el único beneficio que podría brindar la incorporación del paralelismo a las metaheurísticas, sin embargo no lo es. Las implementaciones paralelas suelen explorar el espacio de búsqueda de forma distinta a las implementaciones seriales resultando en cambios en el comportamiento del algoritmo que pueden provocar mejoras significativas en la calidad de las soluciones encontradas.

Ant Colony Optimization (ACO) es una metaheurística basada en población de las más recientes. Tal vez por ese motivo, la aplicación de paralelismo sobre ACO es incipiente y no ha completado su maduración. Actualmente existe un interés creciente en la incorporación de técnicas de alto desempeño sobre esta metaheurística, constatándose una cantidad considerable de artículos publicados sobre esta temática. Sin embargo, es posible detectar algunas carencias importantes, como por ejemplo la escasez de implementaciones paralelas por hardware y sobre sistemas heterogéneos [32].

En un reporte técnico reciente se realizó un amplio relevamiento sobre ACO [43]. El presente documento es complementario a dicho reporte y aborda la aplicación de técnicas de alto desempeño sobre ACO aplicados sobre problemas estáticos monobjetivos de optimización combinatoria, presentando un amplio resumen de las principales propuestas existentes en la literatura sobre esta temática. El relevamiento de propuestas se enmarca en la realidad informática del Instituto de Computación, por lo cual se consideran especialmente los trabajos realizados sobre clusters de PCs y equipos de memoria compartida. Por este motivo se excluyen los trabajos vinculados a implementaciones paralelas por hardware.

El resto del reporte se estructura de la siguiente forma. En la próxima sección se presenta una descripción de las arquitecturas de computadoras paralelas existentes en la actualidad. Cuando se trabaja con implementaciones

paralelas es fundamental poder realizar una evaluación de la mejora en el desempeño. Para realizar dicha evaluación se utilizan una amplia gama de métricas, algunas de las más usadas se presentan en la sección 3. Las implementaciones paralelas de una metaheurística se suelen clasificar según las estrategias seguidas para la incorporación del paralelismo. La presentación de las taxonomías existentes para la paralelización de ACO, así como posibles mejoras se abordan en la sección 4. Posteriormente en la sección 5 se presentan las principales aplicaciones de paralelismo a ACO. En la sección 6 se presentan trabajos que no incluyen la aplicación de paralelismo, pero que por abordar temáticas afines (multicolonia), se considera apropiada su inclusión en este reporte. Finalmente en la sección 7 se resumen las principales conclusiones de este reporte.

2. Arquitecturas de computadoras paralelas

Varias taxonomías han sido propuestas para clasificar las computadoras paralelas. La más utilizada corresponde a Flynn [26] y se caracteriza por considerar en forma independiente las instrucciones y los datos. Al distinguirse para cada uno de esos ítems entre únicos y múltiples, surgen las 4 categorías siguientes: SISD (instrucciones únicas, datos únicos; Single Instruction, Single Data Stream), SIMD (instrucciones únicas, datos múltiples; Single Instruction, Multiple Data Stream), MISD (instrucciones múltiples, datos únicos; Multiple Instruction, Single Data Stream) y MIMD (instrucciones múltiples, datos múltiples; Multiple Instruction, Multiple Data Stream). Las computadoras paralelas son las que están dentro de las categorías SIMD y MIMD.

A continuación se describen brevemente las características de cada categoría.

- SISD: corresponde a las computadoras monoprocesadores comunes.
- MISD: 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 los procesadores que actúan como un pipeline, para ser devueltos a la memoria.
- SIMD: se caracterizan por tener múltiples procesadores que ejecutan las mismas instrucciones sobre diferentes datos en forma sincrónica. Debido a las restricciones causadas por el sincronismo y por la distribución geográfica de los datos típicamente se utilizan para aplicaciones de propósito específico [3].

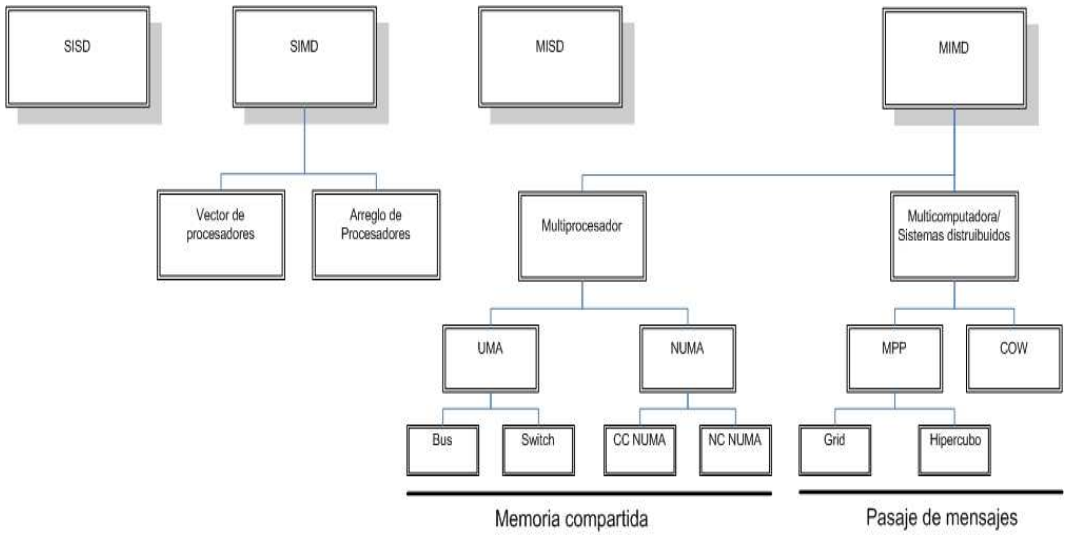

Figura 1: Extensión a la taxonomía de Flynn

- MIMD: se caracterizan por permitir que distintas instrucciones se ejecuten sobre distintos datos en los distintos procesadores en forma asincrónica.

Existen refinamientos que, al considerar aspectos omitidos en la taxonomía original, reflejan en mayor medida la realidad. Un ejemplo es incorporar en el caso de los MIMD subcategorías dependiendo si la memoria está compuesta por un único espacio de direcciones (memoria compartida) o si existen varios espacios diferentes (memoria distribuida). En la figura 1 se presenta una versión ampliada de la taxonomía de Flynn [3].

En el caso de la categoría SIMD es posible realizar un refinamiento entre vector de procesadores (vector processor) y arreglo de procesadores (array processor). Los vectores de procesadores se caracterizan por ser unidimensionales, es decir que trabajan sobre un vector de datos, y realizan las operaciones sobre el vector completo. En el caso de los arreglos de procesadores, la dimensionalidad no está restringida y las operaciones se realizan sobre las componentes de los arreglos.

Los 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 presentan limitaciones en cuanto a la cantidad de procesadores que pueden manejar y costos elevados

[3]. 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.

Se clasifican según las características del tiempo de acceso a memoria en UMA (Uniform Memory Adress) y NUMA (Non Uniform Memory Adress). La primera categoría corresponde a un tiempo de acceso constante para todos los procesadores y la segunda para cuando no lo es.

Dentro de la categoría UMA se incluyen los multiprocesadores simétricos (SMP, Symmetric Multiprocessor). A partir de la forma en que se realizan las conexiones entre los procesadores, es posible distinguir dentro de la categoría UMA, las que son realizadas mediante un bus o un switch.

La categoría NUMA se caracteriza porque los procesadores actúan en forma asincrónica y por incorporar cachés. Suelen existir limitaciones en la escalabilidad debido a que el bus de datos que accede a la memoria global es un cuello de botella. 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 (NC-NUMA, Not Coherent Cache-Non Uniform Memory Adress).

Las multicomputadoras son computadoras en las que cada procesador dispone de una memoria independiente que solamente puede ser accedida por el propio procesador, no existiendo 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. Las multicomputadoras presentan como ventajas significativas su alta escalabilidad y disponibilidad, ya que en caso de provocarse un fallo en uno de los procesadores no afecta al resto.

La categoría COW (Cluster Of Workstations) está formada por los clusters de PCs, es decir un conjunto de computadoras personales interconectadas a través de una red. La cantidad de PCs que se pueden interconectar suele estar limitada por las características inherentes a la tecnología de red utilizada.

Es posible distinguir una categoría adicional, la compuesta por las computadoras masivamente paralelas (MPP, Massively Parallel Processor). Esta categoría puede ser refinada de acuerdo a si el sistema es fuertemente acoplado o no. En el primero de los casos resultan sistemas con topologías como el hipercubo o el toro [3]. En el segundo de los casos los equipos pertenecen a diferentes organizaciones y están geográficamente dispersos, utilizando la infraestructura provista por Internet para realizar la interconexión. Estos

sistemas se conocen como grid o metacomputadoras.

La granularidad de un algoritmo paralelo indica 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 es corto. Cuando la granularidad es gruesa, el tamaño de las tareas es grande y el tiempo de procesamiento es grande. En general, cuanto más pequeña es la granularidad mayor es el potencial para paralelizar el algoritmo pero pueden existir problemas al aumentar los tiempos utilizados para sincronizar procesos y para comunicarse. Se suele asociar los algoritmos de granularidad gruesa con implementaciones en computadoras con memoria distribuida y los algoritmos de granularidad fina con implementación en computadoras con memoria compartida.

3. Medidas de desempeño

La evaluación del desempeño de un algoritmo serial se suele realizar siguiendo un enfoque teórico que consiste en analizar su complejidad computacional en función del tamaño de su entrada. Existen casos en los cuales la dificultad de esa tarea o características propias del algoritmo (típicamente ser no determinístico) vuelven imposible ese enfoque. En esos casos se suele optar por una aproximación empírica centrada en el tiempo de ejecución del algoritmo sobre alguna plataforma específica. Evidentemente, realizar una evaluación empírica implica que esta 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 abordaje usualmente utilizado es el empírico. Otro aspecto que incide en la utilización de un enfoque empírico es la necesidad de conocer la ganancia asociada al trabajar en paralelo en comparación con hacerlo en forma serial.

La primera opción para realizar la evaluación del desempeño computacional podría ser utilizar el tiempo de ejecución al igual que en el caso de los algoritmos seriales. 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. 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, eficacia, utilización de los procesadores y fracción serial. Algunas otras métricas que también se han utilizado en la práctica aunque en menor medida por lo cual no han sido contempladas en este reporte son escalabilidad, isoeficiencia e isovelocidad [47].

La propuesta de métricas específicas para evaluar el desempeño de metaheurísticas ejecutadas en un entorno paralelo prácticamente no ha sido abordado por la comunidad científica. Básicamente se realizan ajustes sobre las métricas generales, por ejemplo utilizando los tiempos promedios de varias ejecuciones debido a que suele trabajarse con algoritmos no determinísticos. La única métrica específica que se ha propuesto es el speedup para metaheurísticas paralelas [16, 48] que se incluye al final de esta sección.

3.1. Speedup absoluto y algorítmico

El speedup es una métrica que permite apreciar el rendimiento relativo al aumentar la cantidad de procesadores en comparación con la utilización de solamente uno. Se conocen 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 1.

$$S_N^{abs} = \frac{T_{serial}}{T_N} \tag{1}$$

Esta definición es poco utilizada en la práctica, debido a que no siempre está disponible el mejor algoritmo serial para poder realizar la comparación en la misma plataforma. Cuando se trabaja con metaheurísticas surgen complicaciones adicionales, por ejemplo determinar cuál es el mejor algoritmo serial que se debería considerar. Algunas posibles opciones que podrían ser consideradas como el mejor algoritmo serial son la mejor versión serial: de la propia metaheurística considerada, de cualquier metaheurística o de cualquier algoritmo que resuelva el problema.

Se suele utilizar como alternativa el speedup algorítmico, también conocido como speedup débil. El speedup algorítmico considera el tiempo de ejecución del propio algoritmo paralelo al ejecutarse sobre un único procesador, es decir en forma serial, en lugar del mejor algoritmo serial. En la ecuación 2 se presenta su formulación, siendo T_i el tiempo del algoritmo paralelo

ejecutando sobre i procesadores.

$$S_N = \frac{T_1}{T_N} \tag{2}$$

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 mejora en el 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 datos y la sincronización de procesos; y la existencia en el algoritmo de tareas intrínsecamente no paralelizables.

3.2. Eficiencia

La eficiencia es una métrica que permite visualizar variaciones en la ganancia de rendimiento al aumentar la cantidad de procesadores utilizados mostrando el porcentaje de tiempo que los procesadores son usados para realizar los cómputos. Su formulación corresponde al valor normalizado del speedup como se muestra en la ecuación 3.

$$E_N = \frac{S_N}{N} = \frac{T_1}{N * T_N} \tag{3}$$

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

3.3. Eficacia

La eficacia es una métrica que mide la relación costo-beneficio de incorporar un nuevo procesador. 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 ($\rightarrow +\infty$) a eficiencias bajas (cercanas a 0) y un costo bajo (~ 1) a eficiencias altas (cercanas a 1). En la ecuación 4 se presenta la formulación de la métrica eficacia.

$$e_N = E_N * S_N = \frac{{S_N}^2}{N} \tag{4}$$

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 el procesador es superior al beneficio de hacerlo. Por lo tanto, el máximo valor de la eficacia indica el mejor compromiso costo-beneficio para el algoritmo paralelo considerado.

3.4. Utilización de los procesadores

La métrica utilización de los procesadores persigue un objetivo distinto al de las métricas presentadas anteriormente. En este caso se busca evaluar qué tan balanceada es la carga entre los distintos procesadores. Por lo cual, el interés es medir la utilización de cada uno de los procesadores que contribuyen con tiempo de cómputo para la ejecución del algoritmo paralelo.

En la ecuación 5 se presenta la formulación de esta métrica, siendo $TC(P_i)$ el tiempo de cómputo efectivo del procesador i y $TO(P_i)$ el tiempo ocioso del procesador i.

$$U(P_i) = \frac{TC(P_i)}{TO(P_i) + TC(P_i)} \tag{5}$$

A pesar de ser una métrica muy intuitiva suelen existir dificultades prácticas para medir los tiempos de cómputo y ocioso de cada procesador, con lo cual es poco utilizada.

3.5. Fracción serial

La fracción serial es una métrica que vincula el tiempo del algoritmo paralelo que necesariamente debe ser ejecutado en forma serial (T_S) y el tiempo de ejecución del algoritmo paralelo en un procesador (T_1) como se muestra en la ecuación 6.

$$f = \frac{T_S}{T_1} \tag{6}$$

Sin embargo, Karp y Flatt [33] plantearon una forma alternativa de calcular la fracción serial, a partir del speedup y la cantidad de procesadores como se muestra en la ecuación 7.

$$f_N = \frac{1/S_N - 1/N}{1 - 1/N} \tag{7}$$

La fracción serial de un algoritmo paralelo es constante, evidentemente al calcularlo utilizando la ecuación 7 debería mantenerse constante. Si el valor de f_N se mantiene constante aunque se provoquen pérdidas en la eficiencia, estas pérdidas se deben a una capacidad limitada en el algoritmo de ser paralelizado. Un crecimiento en f_N puede indicar un desbalance en la carga o que

la granularidad del algoritmo es demasiado fina. Un valor de f_N decreciente indica comportamientos similares a los que se presentan cuando el speedup es superlineal.

3.6. Speedup para metaheurísticas paralelas

La métrica speedup para metaheurísticas paralelas permite evaluar las mejoras en el tiempo de ejecución para obtener niveles de calidad fijos al realizar ejecuciones paralelas independientes de un algoritmo secuencial [16, 48].

Se fija un valor de calidad Q y se realizan ejecuciones paralelas independientes determinándose el tiempo en el que el primer procesador obtuvo una solución de calidad al menos tan buena como Q. 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 la versión paralela sobre N procesadores (T_N^Q). En la ecuación 8 se muestra la formulación de esta métrica.

$$S_N^Q = \frac{T_1^Q}{T_N^Q} \tag{8}$$

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 [16, 48].

Está métrica no ha sido muy utilizada en la práctica debido a la restricción de que solamente es aplicable sobre una implementación particular del paralelismo, las ejecuciones paralelas independientes. Sin embargo, existen puntos de contacto entre esta métrica y el enfoque propuesto por Alba para el speedup algorítmico [3]. El enfoque propuesto por Alba consiste en no utilizar un criterio de esfuerzo prefijado para calcular el speedup algorítmico, sino un nivel de calidad fijo. Se señala que es posible realizar la comparación entre la ejecución en paralelo y una versión secuencial o la versión paralela ejecutada sobre un procesador.

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 coinciden en que no se ha

realizado un trabajo exhaustivo como para otras metaheurísticas [24, 32]. En [24] se señala 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 seriales. También es posible detectar carencias en la propuesta de taxonomías de estrategias para la aplicación del paralelismo. Algunos autores atribuyen estas carencias fundamentalmente a que las formas de paralelizar los algoritmos son bastante naturales [32].

En el resto de la sección se presentan estrategias para aplicar paralelismo sobre ACO. En primer lugar se comentan brevemente algunas taxonomías generales para todas las metaheurísticas. Posteriormente, se abordan las propuestas existentes para ACO. Estas propuestas no son del todo satisfactorias, por lo cual y con el objetivo de detectar carencias se incluye la taxonomía más comúnmente aceptada para Algoritmos Genéticos (AG). Para finalizar esta sección se complementan las taxonomías propuestas para ACO, realizando ajustes para eliminar las carencias detectadas.

4.1. Taxonomías genéricas

La definición de estrategias para la aplicación de paralelismo sobre metaheurísticas ha sido abordada por varios autores, intentando reflejar características generales que trascienden a una metaheurística en particular. Utilizar un enfoque tan genérico puede resultar poco flexible para incorporar particularidades, pero permite una primera conceptualización sobre las ideas que fueron utilizadas en más de una metaheurística. Es conveniente destacar que las taxonomías genéricas en la práctica son poco utilizadas por la comunidad académica cuando se trabaja sobre una metaheurística específica.

La clasificación propuesta por Cung et al. [16] distingue según la cantidad de trayectorias que se exploren entre trayectoria única y trayectorias múltiples. En el caso de la categoría trayectoria única, se paraleliza la búsqueda de una solución, generalmente coincide con la paralelización de la evaluación de la función de costo o la descomposición del dominio cuando se explora un vecindario de soluciones y por consiguiente los algoritmos resultantes tienen granularidad fina o media. En el caso de la categoría trayectorias múltiples, se buscan en paralelo más de una solución resultando algoritmos con granularidad media o gruesa. Al buscar en paralelo más de una solución, es posible que se comparta información entre las búsquedas (búsquedas cooperativas) o que se realicen en forma independiente (búsquedas independientes).

La taxonomía de Cung refleja en gran medida las principales estrategias existentes para implementaciones paralelas de metaheurísticas (ver secciones

4.2 y 4.3).

La clasificación propuesta por Crainic et al. [13] plantea tres dimensiones a ser consideradas:

- *Cardinalidad del control de la búsqueda:* Cuando se trabaja con más de un proceso, es necesario determinar quien y como se 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. En este caso, se incorpora además la noción de conocimiento para reflejar el intercambio de información que implica conocimiento derivado de la búsqueda. De esta forma resultan 4 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 si los distintos procesos utilizan o no las mismas soluciones y las mismas estrategias de búsqueda. Se establecen 4 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 es más amplia que la de Cung abordando algunos aspectos que no suelen ser recogidos comúnmente en las taxonomías, por ejemplo si la comunicación es sincrónica o asincrónica, si utilizan o no las mismas soluciones o si utilizan o no las mismas estrategias de búsqueda.

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 artículos se limitan a realizar una propuesta concreta de implementación y a relevar otras propuestas de implementación.

El trabajo de Randall y Lewis de 2002 es una de las pocas definiciones de estrategias existentes [46]. La propuesta de los autores distingue 5 categorías:

- *Colonias de hormigas paralelas independientes (parallel independent ant colonies):* Corresponde a la ejecución en paralelo de un algoritmo

secuencial ACO sobre los procesadores disponibles. Las ejecuciones son completamente independientes, no existiendo comunicación entre las colonias. Cada una de las colonias puede 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 secuencial ACO sobre los procesadores disponibles. Cada una cantidad fija de iteraciones, las colonias sincronizan sus matrices de feromona, obteniendo la matriz de feromona de la mejor colonia hasta el momento. Evidentemente el volumen de datos que es necesario transmitir entre las colonias es alto por tratarse de 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. Cada proceso esclavo, como si se tratara de una hormiga de la colonia, construye una solución. La comunicación involucra el envío de las soluciones (o la actualización correspondiente en 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. En el artículo se señala que esta estrategia 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 combinar las estrategias hormigas paralelas y evaluación en paralelo de componentes de soluciones. Implica tener dos niveles de 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 (ver sección 4.1), es posible identificar la correspondencia entre: 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 solamente condiciona la forma en la cual las colonias realizan la búsqueda, ya que todas las colonias hacen esencialmente una búsqueda similar, sino que implica la necesidad de determinar la mejor colonia hasta el momento. En el artículo no se establece como determinar cual es la mejor colonia de hormigas, algunas posibles opciones son la que ha obtenido la mejor solución hasta el momento o la que obtuvo la mejor solución en la última iteración. Adicionalmente, pueden surgir problemas asociados a esta determinación, como ser la necesidad de requerir algún agente adicional que decida cual es la mejor colonia para evitar la realización de comunicaciones todos contra todos.

La categoría evaluación en paralelo de componentes de soluciones no ha sido llevada a la práctica y parece de difícil aplicación.

En [32] no se realiza un planteo explícito sobre las formas en las cuales es posible implementar ACO paralelos, sin embargo se presentan varios aspectos relativos a algunas implementaciones paralelas que convienen ser tenidos en cuenta y 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. A partir de lo cual, la granularidad mínima del algoritmo debiera ser la construcción de una solución. Algunos problemas pueden requerir paralelizar el cálculo de la calidad de una solución debido a que sea una tarea compleja, pero este hecho esta más fuertemente relacionado a dificultades particulares de un problema específico y no a características intrínsecas a la forma de paralelizar ACO. Este aspecto del proceso de creación de las soluciones va en correlación con la dificultad señalada para llevar a la práctica la categoría de Randall y Lewis 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 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 la eficiencia, pudiendo modificar el comportamiento 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, pudiendo no coincidir estas matrices. La propuesta original del modelo multicolonia tiene como objetivos la mejora en la calidad de los resultados y su aplicación a problemas multiobjetivo. Si bien es posible implementar una multicolonia en forma secuencial, resulta natural la aplicación de técnicas de alta performance para mejorar los tiempos de ejecución.

Janson et al. [32] resumen los principales aspectos que deben ser considerados al momento de diseñar un algoritmo ACO multicolonia. Estos aspectos son:

- *Estructura de comunicación y topología:* Típicamente se han utilizado las topologías todos contra todos, 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 elitista si la solución recibida es mejor, agregarla a la matriz de feromona o agregarla a la generación actual.
- *Cadencia de la comunicación:* Las variantes típicamente utilizadas han sido realizar la comunicación en cada iteración, cada una cantidad 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, es decir que en cada iteración las colonias tienen diferentes parámetros o un enfoque heterogéneo entre las iteraciones, es decir que cada una cierta cantidad de iteraciones se modifican los parámetros de las colonias pero todas utilizan los mismos valores.

Evidentemente existe una fuerte similitud entre multicolonia y colonias de hormigas paralelas con interacción. El enfoque presentado por Janson et al. es superior que el de Randall y Lewis, ya que flexibiza la restricción de sincronizar las matrices de feromona, permitiendo agrupar una mayor cantidad de propuesta bajo esta categoría.

4.3. Taxonomía de paralelismo aplicado a AG

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

Actualmente, las estrategias para aplicar paralelismo a AG están bien definidas y están resumidas en una taxonomía que es aceptada por la comunidad académica. La taxonomía establece las siguientes categorías [8, 37]:

- *Ejecuciones paralelas independientes:* Este modelo consiste en la ejecución en paralelo de un mismo algoritmo secuencial sin ningún tipo de interacción. Se caracteriza por tener más de una población pero sin comunicación entre ellas.

- *Modelo maestro-esclavo:* Este modelo se caracteriza por trabajar sobre una única población. El maestro se encarga de la ejecución del AG y utiliza a los esclavos para la evaluación de la función de fitness. La forma en que se realiza la 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 múltiples poblaciones. Cada población trabaja en forma independiente y cada un cierto tiempo cada población comunica individuos (migrantes) a las otras para influirlas. Los parámetros de las migraciones que deben ser tenidos en cuenta son la tasa de migración, el intervalo entre migraciones, las políticas de selección y/o remplazo de migrantes y la topología de las poblaciones.

- *Modelo celular:* Este modelo se caracteriza por trabajar con una única poblacion que se estructura en vecindades. Cada individuo interactúa solamente con los individuos vecinos.

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

Existen muchos puntos de contacto entre esta taxonomía y la presentada en la sección 4.2 en lo que respecta a los modelos identificados. La gran diferencia que se puede apreciar es la ausencia en ACO de una categoría de características similares al modelo celular de AG.

La ausencia de un modelo celular en ACO puede explicarse porque la matriz de feromona, que mantiene la memoria de la búsqueda, debe ser utilizada por 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. Evidentemente este enfoque implica cambiar sustancialmente la forma de implementar el ACO y su funcionamiento.

4.4. Ampliación de la taxonomía de paralelismo aplicado a ACO

La taxonomía de Randall y Lewis presenta fundamentalmente dos aspectos que deben ser corregidos: la ausencia de un modelo celular y las fuertes restricciones impuestas sobre el modelo de colonias de hormigas paralelas con interacción. A continuación, se presenta una propuesta de mejora sobre dicha taxonomía. Se prefirió utilizar los nombres de las categorías presentes en AG por ser ampliamente conocidos por la comunidad científica.

La taxonomía de estrategias para implementaciones de ACO paralelos propuesta 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 dos subcategorías. La primer subcategoría es un modelo maestro-esclavo de grano fino, en el cual los esclavos se encargan de construir soluciones completas. Los esclavos no tienen porque 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 la/s solución/es al maestro. La subcategoría anteriormente descripta se corresponde en gran medida con la categoría hormigas paralelas de Randall y Lewis. La segunda subcategoría tiene grano más fino que la primera 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 et al. [32]. Las implementaciones comprendidas en este modelo son de grano grueso.

- *Modelo celular:* Este modelo se caracteriza por trabajar con una única población que se estructura en vecindades. Cada hormiga interactúa solamente con las hormigas vecinas, teniendo su propia matriz de feromona. A partir del relevamiento de aplicaciones de paralelismo sobre ACO realizado (ver sección 5) todavía no existen implementaciones de este modelo. Las implementaciones comprendidas en este modelo son de grano fino.

- *Otros modelos o híbridos:* En esta categoría, del mismo modo que en AG, se incluyen las implementaciones que utilicen más de un modelo. La categoría combinación paralela de hormigas y evaluación de componentes de soluciones de Randall y Lewis es un ejemplo de híbrido ya que utiliza los dos modelos maestro-esclavo en forma jerárquica.

La categoría colonia de hormigas paralelas con interacción de Randall y Lewis queda comprendida dentro del modelo multicolonia.

Es conveniente distinguir la diferencia entre el modelo detrás del paralelismo y la implementación particular utilizada. Por ejemplo, si se utiliza un modelo multicolonia con topología todos contra todos, es común utilizar un proceso maestro que reciba las soluciones y envíe a las colonias la mejor solución. La utilización de una implementación del tipo maestro-esclavo reduce sustancialmente la comunicación entre procesos. En este reporte siempre se hace referencia a la estrategia de paralelismo seguida y no a los detalles particulares de la implementación.

5. Trabajos de paralelismo aplicado a ACO

Una de las primeras aproximaciones al problema de implementar un algoritmo ACO paralelo se atribuye a Bolondi y Bondaza en el año 1993 [6]. Ese trabajo es difícil de conseguir en la actualidad, por lo cual se resume la impresión que Middendorf et al. [41] tienen de él. La implementación propuesta por Bolondi y Bondaza es de grano muy fino y corresponde a colocar una hormiga en cada procesador. Al crecer la cantidad de procesadores, la implementación propuesta no escalaba debido al overhead causado por las comunicaciones.

A partir de la descripción que realiza Middendorf et al. no es claro si cada hormiga tenía su propia matriz de feromona y si cada hormiga comunicaba sus soluciones a todas las hormigas o si se definían vecindades (modelo celular). Como se señala como una falencia el no escalamiento de la solución al aumentar la cantidad de procesadores, es razonable asumir que la comunicación es todos contra todos.

A continuación se presenta un resumen de artículos relevantes en los cuales se implementan ACO paralelos:

- **Parallelization Strategies for Ant Colony Optimization (1998)**

  El artículo [48] aborda el TSP mediante ejecuciones paralelas independientes de $\mathcal{MM}$AS con búsqueda local. Su objetivo es evaluar si es mejor ejecutar en paralelo que realizar una única ejecución de mayor tiempo. Solamente se estudia el efecto del paralelismo en la calidad de las soluciones obtenidas. Para realizar una evaluación justa a las ejecuciones en paralelo se les da un tiempo máximo de procesamiento igual a la k-ésima parte del tiempo que se le da a la ejecución serial (siendo k la cantidad de procesadores). Los resultados son favorables a las ejecuciones paralelas obteniendo en todos los casos soluciones de mayor calidad que el algoritmo secuencial.

- **Parallelization Strategies for the Ant System (1998)**

  El artículo [7] aborda la implementación paralela mediante el modelo maestro-esclavo para AS. Se formulan dos propuestas de paralelización del AS. La primera propuesta es 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 busca evitar ese inconveniente y consiste en una implementación paralela parcialmente asíncrona. En la implementación parcialmente asíncrona cada esclavo ejecuta durante una cierta cantidad de iteraciones realizando actualizaciones locales de la matriz de feromona en forma independiente. Posteriormente, cada una cierta cantidad 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.

  En el artículo no se entra en mayores detalles sobre como se realizarían las actualizaciones de la matriz de feromona en el caso de la implementación parcialmente asíncrona porque el interés de los autores es únicamente evaluar el efecto de las comunicaciones. Para realizar esa evaluación simulan las comunicaciones mediante N-MAP [7]. Las métricas utilizadas para realizar la evaluación son speedup, eficiencia y eficacia, comprobándose que la implementación parcialmente asíncrona tiene un mejor desempeño que la implementación sincrónica. De todas formas, la implementación sincrónica en instancias grandes logra spee-

dups casi lineales. Por tratarse de una simulación no hay estudios sobre la calidad de las soluciones.

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

  En los artículos [49, 50] se presenta una implementación paralela de ANTabu para el QAP. ANTabu es un híbrido ya que es un ACO que incorpora un mecanismo de búsqueda local basado en Tabu Search. La búsqueda local solamente incorpora una memoria de corto plazo para evitar la generación de ciclos. Las movidas solamente se aceptan si generan una mejor solución y se permite solamente una cantidad pequeña de iteraciones para evitar provocar una convergencia prematura. La propuesta también incluye un mecanismo de diversificación que consiste en la reinicialización de la matriz de feromona cuando no se producen mejoras por una cierta cantidad de iteraciones.

  La implementación paralela consiste en un maestro-esclavo de grano fino sincrónico. El maestro maneja la información global de la colonia, es decir la matriz de feromona y la mejor solución hasta el momento. Los esclavos cumplen con las tareas propias del proceso de búsqueda de cada hormiga, es decir construyen las soluciones y aplican el Tabu Search. Los esclavos envían las soluciones que generan al maestro y este envía la matriz de feromona a los esclavos.

  Los autores evalúan la calidad de las soluciones obtenidas por su propuesta contra las obtenidas por otras variantes de hormigas (HAS-QAP), otros algoritmos paralelos (PATS, Parallel Adaptive Tabu Search), variantes de Algoritmos Genéticos y variantes de Variable Neighbourhood Search. La evaluación la realizan sobre un cluster de workstations utilizando un tiempo fijado previamente, obteniendo buenos resultados que aventajan a la gran mayoría de las otras variantes.

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

  El artículo [40] es uno de los primeros trabajos en los cuales se aborda el modelo multicolonia. Se trabaja sobre la variante AS con elitismo y el problema TSP.

  Cada colonia tiene el mismo comportamiento pero utiliza su propia matriz de feromona. Cada un número fijo de iteraciones se produce el intercambio de información entre las colonias. Los autores señalan que en un trabajo previo que permanece sin publicar ([35]) se pudo comprobar que es mejor intercambiar las mejores soluciones que las matrices

de feromona completas. Por tanto se plantean cuatro propuestas de intercambio: la mejor solución global a todas las colonias; las mejores soluciones locales utilizando una topología de anillo unidireccional, solamente se modifica la mejor solución de la colonia destino en caso de ser necesario; migrantes utilizando una topología de anillo unidireccional, se actualiza la matriz de feromona a partir de las mejores soluciones, ya sean generadas por la colonia o recibidas como migrantes; y la aplicación a la vez de los dos últimos criterios.

Solamente se realiza la evaluación de la calidad de las soluciones obtenidas considerando también la no incorporación de intercambios. Los mejores resultados se obtienen utilizando solamente las mejores soluciones locales. Finalmente, se analiza cada cuantas iteraciones es conveniente realizar el intercambio. Naturalmente debe existir un equilibrio, si es cada muy pocas o cada muchas iteraciones se degrada la calidad de las soluciones obtenidas.

- **Multi Colony Ant Algorithms (2002)**

  El artículo [41] es complementario del artículo [40]. Se trabaja sobre la misma propuesta, incorporando a la evaluación la aplicación sobre el QAP. Solamente se evalúa el intercambio de las mejores soluciones locales y la no utilización de intercambios. Se obtienen buenos resultados aunque en menor medida que para el TSP.

  Finalmente se evalúa la conveniencia de hacer una ejecución más larga pero con solamente una colonia contra la utilización de varias colonias (con o sin comunicación) resultando superior el enfoque con más de una colonia.

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

  En el artículo [46] se aborda el TSP mediante una implementación paralela de ACS siguiendo un modelo maestro-esclavo de grano fino.

  La propuesta presenta la particularidad que la actualización local del rastro de feromona la realiza el maestro. Cuando se incorpora un componente a un esclavo, este envía al maestro que componente incluyó. Posteriormente el maestro envía la actualización local del rastro de feromona correspondiente.

  Los autores evalúan su propuesta sobre un IBM SP2 utilizando entre 2 y 8 procesadores. Solamente se realiza una evaluación del speedup y eficiencia pero no de la calidad de las soluciones obtenidas. El speedup está muy por debajo de ser lineal y la eficiencia máxima se obtiene utilizando 2 procesadores. Las instancias más grandes son las que presentan

los mejores resultados. Los pobres resultados muestran que el enfoque seguido para la actualización local no es un buen esquema para el paralelismo. Los autores señalan que la utilización de una arquitectura de memoria compartida podría simplificar la comunicación.

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

  El artículo [4] aborda una implementación paralela multicolonia de ACS sobre el TSP. La implementación es en forma asincrónica y con la topología todos contra todos. La evaluación se realiza en un cluster de PCs homogéneo de 4 computadoras. En lo que concierne a la calidad de los resultados obtenidos, se obtienen leves mejoras al paralelizar. En cuanto al desempeño los resultados son muy buenos, obteniendo un speedup superior a 3,8 y una eficiencia del 98 %.

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

  En el artículo [45] el problema abordado es el Set Covering Problem (SCP) mediante una variante de ACO específica para el problema. Este problema suele presentar dificultades para los algoritmos al trabajar sobre instancias grandes. La propuesta plantea la incorporación de un operador de búsqueda local debido a que se obtuvieron resultados pobres al utilizar el algoritmo puro. La utilización del operador de búsqueda mejora sustancialmente los resultados pero duplica el tiempo de ejecución del algoritmo.

  Buscando reducir el tiempo de ejecución se plantean dos implementaciones paralelas, una mediante ejecuciones independientes y otra que corresponde a un maestro-esclavo de grano fino. La implementación mediante ejecuciones independientes logra speedups casi lineales y eficiencias cercanas al 100 % debido a que la comunicación entre procesos es escasa. En el caso de la implementación maestro-esclavo en términos de eficiencia depende fuertemente del tamaño de las instancias consideradas. Para dicha propuesta el speedup promedio es 10,95 (trabajan sobre 20 procesadores) y la eficiencia varía entre 21,03 y 83,47.

  En el trabajo no se presenta específicamente un estudio sobre la calidad de las soluciones obtenidas por ambas propuestas.

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

  En el artículo [44] se aborda el TSP mediante una variante que incorpora características de ACS y $\mathcal{MM}$AS. La implementación paralela es sincrónica y utiliza un modelo multicolonia en la cual cada colonia

realiza la actualización local del rastro de feromona y la actualización global se realiza en forma conjunta cada una cantidad fija de iteraciones. Para la actualización global se utilizan la mejor solución de cada colonia que es enviada cada una cantidad fija de iteraciones.

La evaluación es realizada sobre un equipo Cray T3E y no sigue un enfoque ortodoxo. Se utiliza como indicadores el costo de la solución, el tiempo total de cómputo, el tiempo por procesador, el porcentaje de tiempo total de comunicación con respecto al tiempo total de cómputo y el porcentaje de tiempo ocioso con respecto al tiempo total de cómputo. Se obtienen resultados de buena calidad manteniendo acotados el tiempo ocioso de los procesadores y el tiempo de comunicación.

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

  En el artículo [31] se aborda un problema de ruteo sobre una red MANET (Mobile Ad-hoc Network). El problema consiste en determinar el mejor camino entre todos los pares de nodos de la red, se conoce a este problema como Áll-pair routing'. Los autores proponen un ACO específico con la particularidad que las hormigas cuando construyen las soluciones, antes de considerar el rastro de feromona, consideran los nodos que todavía no han sido seleccionados.

  En cada procesador se coloca una hormiga y una parte de la información disponible de toda la red. Cada hormiga determina las mejores rutas de 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. El artículo no es claro sobre como se realiza la comunicación entre las hormigas para armar las soluciones completas ni que estrategia se sigue para la paralelización.

  La evaluación se realiza sobre un cluster de PCs considerando el tiempo de ejecución y el speedup. El tiempo de ejecución disminuye al aumentar la cantidad de procesadores. Del tiempo de ejecución más del 90 % se utiliza en comunicación. El speedup es sublineal, teniendo un valor de 7 para 10 procesadores.

- **Parallel Ant Colony Systems (2003)**

  En el artículo [12] se aborda el TSP mediante la implementación paralela de la variante ACS. Los autores, además de reducir el tiempo de ejecución, se plantean como objetivo la mejora de la calidad de las soluciones obtenidas.

El enfoque seguido es el de una multicolonia y la idea es que cada colonia pueda influir en las matrices de feromona de las otras colonias. Cada colonia ejecuta un ACS y se produce la comunicación entre las colonias cada una cantidad fija de iteraciones. La información enviada por cada colonia es la mejor solución hasta el momento. Cada colonia realiza la actualización normal de feromona y en las iteraciones en las cuales recibe soluciones de otras colonias incorpora una actualización adicional.

Se proponen tres formas diferentes de utilizar la información intercambiada: 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).

Se realiza solamente la evaluación de la calidad de las soluciones obtenidas, en la cual PACS supera ampliamente a AS y a ACS.

- **Ant Colony System with Communication Strategies (2004)**

  El artículo [11] es complementario del [12] e incorpora la evaluación de distintas estrategias de comunicación. La información enviada por cada colonia es la mejor solución hasta el momento. Cada colonia realiza la actualización normal de feromona y en las iteraciones en las cuales recibe soluciones de otras colonias incorpora una actualización adicional.

  Las estrategias de comunicación que se evalúan son: 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.

  Se realiza solamente la evaluación de la calidad de las soluciones obtenidas, en la cual PACS supera ampliamente a AS y a ACS.

- **Parallel Framework for Ant-like Algorithms (2004)**

  En el artículo [14] se aborda la implementación de un framework reusable para la ejecución de algoritmos secuenciales en ambientes paralelos siguiendo un modelo maestro-esclavo. Se utiliza para probar el framework el algoritmo ACO sobre TSP. Dentro de los objetivos de

diseño se destaca obtener un buen nivel de abstracción y optimizar la comunicación entre los procesos.

La comunicación entre el maestro y los esclavos se da en checkpoints. Los esclavos solicitan intercambios de datos con el maestro por turnos (scheduling) en forma asíncrona. Solamente se intercambia, entre el maestro y el esclavo, la información que se haya modificado desde el último checkpoint de ese esclavo.

Para la evaluación solamente se considera el speedup, siendo este casi lineal hasta los 25 procesadores y degradándose si se utilizan más procesadores. Los autores plantean la necesidad de estudiar un modelo multinivel que permita mantener el speedup casi lineal.

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

  El artículo [21] aborda el Vehicle Routing Problem (VRP) mediante una implementación paralela de AS_{rank} siguiendo un modelo maestro-esclavo. En el modelo maestro-esclavo planteado es posible colocar en cada procesador más de una hormiga. El objetivo de la propuesta es mejorar el tiempo de ejecución del algoritmo.

  Para evaluar la propuesta, se evalúa el speedup y la eficiencia sobre un cluster Beowulf. La evaluación se realiza sobre 1, 2, 4, 8, 16 y 32 procesadores. El speedup es sublineal y la eficiencia decrece al aumentar la cantidad de procesadores. Un buen compromiso entre el desempeño y la cantidad de procesadores se da utilizando 8 procesadores (speedup 6 y eficiencia superior al 70 %).

- **Analyzing the Behavior of Parallel Ant Colony Systems for Large Instances of the Task Scheduling Problem (2005)**

  En el artículo [2] se aborda la aplicación de paralelismo sobre ACS para el Minimum Tardy Task Problem (MTTP).

  Se implementan los siguientes modelos: ejecuciones paralelas independientes, modelo de islas con topología de anillo y un modelo maestro-esclavo de grano fino. En el modelo de islas, cada una cantidad fija de iteraciones se comunica el mejor individuo a la colonia correspondiente. Esa colonia para actualizar el rastro de feromona en la iteración que recibe el migrante, considera las soluciones que genera en esa iteración y el migrante recibido.

  Para realizar la evaluación de las propuestas se consideran 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. Las pruebas se realizan sobre un cluster de PCs.

La calidad de las soluciones obtenidas y la iteración en la que obtienen 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. Se constata que las ejecuciones independientes requieren menor tiempo que el modelo de islas, esto resulta natural ya que no hay comunicación entre las ejecuciones independientes.

También se evalúa el efecto en el speedup y la eficiencia de 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 al 30 % para 3 procesadores. Se señala que el problema presenta algunas particularidades (el tamaño de las soluciones es variable) que pueden influir en los resultados obtenidos.

- **Multi-Level Parallel Framework (2005)**

  El artículo [15] es complementario del [14] y aborda la implementación de un framework piramidal siguiendo un modelo maestro-esclavo con la inclusión de submaestros. Incorpora un nivel de submaestros que actúan como concentradores de los esclavos que están bajo su órbita.

  El funcionamiento sigue las mismas pautas que [14], reduciendo la comunicación al maestro por la incorporación de los submaestros. Se obtienen mejoras para el speedup con más de 25 procesadores, logrando que este sea casi lineal.

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

  El artículo [10] presenta una de las escasas aplicaciones de ACO sobre un problema de bioinformática. El problema abordado es el de predicción de la estructura de proteínas (PSPP).

  Se plantean tres implementaciones paralelas. La primera consiste en un maestro-esclavo de grano fino. La segunda corresponde a una multicolonia con intercambio circular de migrantes aunque se implementa siguiendo un enfoque maestro-esclavo. En ese caso el maestro se encarga de manejar todas las matrices de feromona. La tercera implementación consiste en una multicolonia pero con matriz de feromona compartida. En ese caso también se sigue un enfoque maestro-esclavo para su implementación.

La evaluación de los resultados no es ortodoxa. Se utiliza como métrica la cantidad de tics de CPU necesarios para encontrar el óptimo. Las pruebas se realizan sobre 3, 4 y 5 procesadores. Utilizando la métrica antes mencionada, se observa que en general el mejor desempeño lo obtiene la multicolonia con intercambio de migrantes, seguida de la multicolonia con matriz de feromona compartida.

- **Comparing Parallelization of an ACO: Message Passing vs. Shared Memory (2005)**

  El artículo [18] es uno de los escasos trabajos que abordan la implementación de un ACO paralelo en una arquitectura con memoria compartida. La variante utilizada es ACS sobre el TSP tomando un enfoque similar al de [46] en lo que respecta a la aplicación del paralelismo.

  En un trabajo previo ([20]) algunos de los autores de este artículo plantearon una implementación paralela para una variante de ACO sobre un problema multiobjetivo de planificación. Esa implementación se corresponde en gran medida con la propuesta formulada en este artículo.

  La utilización de ACS acarrea complicaciones en un ambiente de memoria compartida debido a que la etapa de actualización local de feromona implicaría demasiadas sincronizaciones. Los autores realizaron un estudio del efecto que tiene sobre la calidad de las soluciones relajar este requisito, no encontrando efectos negativos.

  La implementación realizada utiliza dos regiones críticas para que solamente una hormiga a la vez realice la actualización local del rastro de feromona y la actualización de la mejor solución encontrada. Otra característica de la implementación es que sincroniza los procesos para realizar la actualización global del rastro de feromona.

  Para realizar la evaluación de su propuesta, los autores comparan el speedup y la eficiencia obtenidos por su implementación y por [46]. Utilizan entre 2 y 8 procesadores obteniendo mejores resultados que Randall y Lewis. Sin embargo, se debe considerar que: la definición utilizada de speedup en [46] no es la misma que la usada en [18]; Randall y Lewis intercambian mensajes para realizar la actualización local de feromona y en esta propuesta se omite; y los equipos utilizados no son iguales.

  Posteriormente evalúan el efecto que tiene el aumento en la cantidad de hormigas utilizada por proceso sobre el speedup y la eficiencia, obteniendo mejoras considerables a costa de disminuir la calidad de las

soluciones obtenidas. Realizan esta evaluación considerando una cantidad fija de iteraciones y otra dependiente de la cantidad de procesos (cada proceso *ciclos/procesos*). Utilizando una cantidad fija se logra paliar en parte la pérdida de calidad pero nunca se logra que sea igual a la obtenida utilizando una hormiga por procesador.

Los autores proponen una hibridización entre las técnicas de memoria compartida y de pasaje de mensajes. La idea consiste en tener una multicolonia en la que las colonias se comuniquen entre si mediante pasaje de mensajes y que el mecanismo de comunicación dentro de cada una de las colonias sea mediante memoria compartida. De este relevamiento surge que esa propuesta no se ha llevado a la práctica.

- **A Shared Memory Parallel Implementation of Ant Colony Optimization (2005)**

  El artículo [19] es complementario del artículo [18] manteniendo las principales características de la implementación. La única diferencia es la incorporación de una etapa de búsqueda local después de construir las soluciones.

  Realizan la evaluación sobre dos equipos diferentes, un IBM/P 1600 y un SGI Origin 3800. Se evalúa el speedup y la eficiencia para 1, 2, 4, 8 y 16 procesadores. El equipo IBM presenta sistemáticamente un mejor desempeño. Posteriormente, modifican el tipo de nodos utilizados por el equipo, pasando de un NH2 Power 3 a un Regatta Power 4, obteniendo importantes mejoras en el desempeño.

  Los autores señalan que las limitaciones en el desempeño no se deben únicamente a restricciones del algoritmo. Esta experiencia indica que la evolución de la tecnología de la computación paralela puede provocar mejoras en el desempeño de los algoritmos.

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

  El artículo [42] es el único trabajo que surge en este relevamiento en el cual se aborda la implementación de un ACO paralelo en un entorno Grid.

  La aplicación sobre la que trabajan es bastante novedosa y se conoce como 'Museos híbridos'. Los museos híbridos buscan maximizar la experiencia de un visitante de un museo con un tiempo limitado para recorrerlo. El visitante dispone de un dispositivo portátil (handheld) mediante el que solicita una visita automática indicando una obra de

arte origen, una obra de arte destino y un tiempo máximo para la recorrida. Cada obra de arte tiene asociado un puntaje que es el objetivo a maximizar con la restricción de tiempo del visitante. El dispositivo obtiene una solución que orienta paso a paso al visitante, en una visita guiada con información multimedia.

El problema presenta algunas dificultades ya que los museos suelen tener miles de obras de arte, por lo cual una solución centralizada podría implicar demoras inaceptables para el visitante. Esto también se refleja en el enfoque seguido por los autores al aplicar ACO sobre el problema en cuestión, ya que descartan que las hormigas o las colonias trabajen sobre todo el espacio de búsqueda.

Su enfoque se caracteriza por separar en clusters de acuerdo a la distancia euclidiana entre las obras, obtener soluciones parciales sobre los grafos clusterizados utilizando colonias de hormigas y unir las soluciones parciales para obtener una solución global. Es posible que al unir las soluciones parciales, la solución que se obtenga no sea factible, por lo cual se realizan búsquedas locales hasta factibilizar la solución.

La solución planteada requiere que exista un servicio centralizado que se encarga de separar en clusters, registrar los trabajadores (se utiliza una colonia por elemento de cómputo) y unir las soluciones parciales. La implementación realizada utiliza Web Services en .NET debido a que puede ser ejecutado tanto en PCs como en dispositivos móviles (handheld).

Para evaluar la propuesta, en primer lugar comparan la calidad de las soluciones sin realizar el clustering contra otras implementaciones de ACO, obteniendo resultados similares. La siguiente etapa consiste en ver la evolución al intentar obtener buenas soluciones en tiempos razonables. La instancia de prueba que utilizan es grande (tiene 1000 nodos) y utilizan hasta 32 PCs. El tiempo de ejecución del algoritmo propuesto decrece exponencialmente al aumentar la cantidad de colonias. En lo que respecta a la calidad de las soluciones obtenidas, se constata que al aumentar la cantidad de colonias se provoca una mejora en los resultados (los clusters quedan mejor repartidos). Aunque, a partir de las 32 colonias se comienza a degradar. La experiencia parece indicar que existe una fuerte dependencia entre el tamaño del problema a resolver y la cantidad de colonias (en definitiva la cantidad de clusters utilizados).

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

El artículo [29] aborda la paralelización de una variante poco difundida de ACO llamada Queen Ant Strategy (AS_{queen}) del cual existe muy poca información disponible (el artículo en el que se propone está publicado en japonés [30]). La variante se caracteriza por tener una reina y un grupo de agentes que generan soluciones (similar a las colonias), la reina a partir de las soluciones generadas por los agentes, les da directivas sobre si deben diversificar o explotar la búsqueda. Presenta grandes similitudes con una multicolonia con un control centralizado.

Se propone una implementación distribuida utilizando un espacio de objetos compartidos siguiendo el patrón de diseño trabajadores replicados ('replicated-worker pattern'). El intercambio de objetos se produce a través de un repositorio virtual de objetos 'JavaSpaces'.

Para evaluar su propuesta comparan la ejecución concurrente en un PC y en forma distribuida en un cluster heterogéneo de hasta 8 PCs. En lo que respecta a la calidad, no hay diferencias significativas para la cantidad de veces que obtienen el óptimo ni para el costo promedio de las soluciones obtenidas. En lo que respecta al tiempo promedio para encontrar el óptimo baja en forma considerable al aumentar la cantidad de grupos trabajando en forma distribuida.

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

  En el artículo [9] se trabaja con una multicolonia sobre una computadora masivamente paralela Dawn 2000. El problema utilizado es el TSP.

  En cada procesador se coloca una colonia y el intercambio de información se realiza en forma adaptativa de acuerdo al fitness de cada colonia (manejan la noción de fitness propia de los AEs). Siempre se intercambia la mejor solución y se proponen dos mecanismos para definir entre que colonias se produce el intercambio.

  La primera propuesta ordena las colonias de acuerdo al fitness promedio de la iteración. Dado el ordenamiento $1, 2, ..., p-1, p$ el intercambio se produce entre los pares de colonias que ocupan posiciones que sumadas dan $p+1$, es decir 1 y p, 2 y $p-1$ y así sucesivamente. Cada colonia actualiza el rastro de feromona considerando la calidad de la mejor solución del procesador y la solución recibida y ponderando según las posiciones obtenidas en el ranking.

  La segunda propuesta considera las diferencias entre las componentes de las mejores soluciones de cada colonia. La colonia i selecciona para el intercambio a la colonia j, si todavía no ha sido seleccionada y si es

la colonia con el máximo valor para la proporción entre la cantidad de componentes distintas que tengan las mejores soluciones de las colonias y el costo de la mejor solución de j. El rastro de feromona se actualiza proporcionalmente a la calidad de las soluciones.

También se presenta una formulación para el ajuste del intervalo de tiempo entre los intercambios en forma adaptativa a partir de la diversidad de las soluciones generadas en todas las colonias.

La propuesta requiere tener un proceso centralizado que realice el ordenamiento del fitness o el cálculo de las diferencias y el ajuste del intervalo de tiempo, ya que realizar las comunicaciones todos contra todos tendría demasiado overhead.

En primer lugar se evalúa la calidad de las soluciones y el tiempo de ejecución para los algoritmos ACO (no se especifica la variante utilizada), multicolonia con intercambio circular de la mejor solución local y las dos variantes propuestas. La calidad y el tiempo de ejecución son mejores para las propuestas adaptativas, siendo la variante de las diferencias superior a la del ordenamiento del fitness. Conviene señalar que resulta extraño que la implementación de multicolonia con intercambio circular tenga un mayor tiempo de ejecución que las variantes propuestas, ya que no requiere un proceso centralizado e inclusive podría ser implementada en forma asincrónica. En todo caso, la obtención de un peor tiempo por su parte, parece ser más atribuible a carencias en su implementación que a la propia variante.

En segundo lugar se evalúa el speedup sobre 5, 10, 15, 20, 25, 30 y 35 procesadores. El speedup obtenido por la propuesta es sublineal, siendo mejor para instancias grandes del problema y peor al aumentar la cantidad de procesadores.

En lo que respecta al mecanismo propuesto de ajuste del intervalo entre los intercambios obtiene soluciones de mejor calidad que mediante intervalos fijos pero tiene un mayor tiempo de ejecución.

- **Communication Strategies for Parallel Cooperative Ant Colony Optimization on Clusters and Grids (2005)**

  En el artículo [5] se trabaja sobre el VRP con una variante específica conocida como Savings based ACO. La variante se caracteriza por considerar para la actualización del rastro de feromona a la mejor solución y a un grupo de soluciones elitistas. Los autores señalan que realizan una implementación paralela para reducir los tiempos de ejecución y mejor la calidad de las soluciones obtenidas.

Para incorporar el paralelismo se plantean las tres estrategias siguientes: un modelo multicolonia, un modelo maestro-esclavo grano fino y un modelo híbrido.

En el modelo multicolonia se evalúan 6 estrategias de comunicación: sin comunicación; se intercambia la mejor solución; se intercambia la mejor solución y las soluciones elitistas; se intercambia la mejor solución y se reinician las matrices de feromona cuando se detecta estancamiento; se intercambia la mejor solución y las soluciones elitistas y se reinician las matrices de feromona cuando se detecta estancamiento; y la mejor colonia envía la matriz de feromona.

El modelo híbrido se estructura en forma jerárquica con una multicolonia en la que cada colonia está implementada con un modelo maestro-esclavo.

La evaluación se realiza sobre un cluster. Para el modelo maestro-esclavo se observa que el speedup mejora al trabajar con instancias más grandes del problema obteniendo para 8 procesadores un speedup de 6 y una eficiencia superior al 70 %. Si se aumenta la cantidad de procesadores se comienza a degradar el desempeño. Para el modelo híbrido se comprueba que el intercambio de la mejor solución y las soluciones elitistas presenta un buen balance entre tiempo de ejecución y calidad de las soluciones obtenidas. No se realizan pruebas independientes para el modelo multicolonia.

Finalmente, los autores discuten los aspectos tecnológicos de una posible implementación Grid mediante Web Services, pero no se implementa.

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

  En el artículo [22] se trabaja sobre el VRP con una variante específica conocida como D-Ant que se basa en la descomposición del problema en subproblemas más chicos. El algoritmo planteado tiene las siguientes etapas: generación de una solución inicial mediante la utilización de Savings based ACO; determinación de los centros de gravedad de cada ruta de la solución; obtención de clusters a partir de los centros de gravedad y los consumidores; y considerar como subproblemas cada uno de los clusters y resolverlos mediante distintas colonias que utilizan Savings based ACO.

  En este caso, cada colonia trabaja en forma independiente y posteriormente se unen las soluciones. A pesar de que cada colonia utiliza

diferentes pedazos de la matriz de feromona y realiza las actualizaciones 'locales', la matriz de feromona se maneja en forma global. Un proceso maestro se encarga de realizar la evaporación y el refuerzo de la mejor solución obtenida hasta el momento. En lo que respecta al paralelismo, es como utilizar un modelo maestro-esclavo ya que se ven todas las colonias como si fueran solamente una.

La evaluación la realizan sobre un cluster comprobando que se mantiene la calidad y se reduce el tiempo de ejecución. Para 8 procesadores se obtiene un speedup de 3,85 y una eficiencia de 0,48.

- **Parallel Cooperative Savings Based Ant Colony Optimization - Multiple Search and Decomposition Approaches (2006)**

  El artículo [23] es complementario de los artículos [5] y (FALTA REFERENCIA). Se evalúan las mismas propuestas formuladas en los dos artículos anteriores con pequeñas modificaciones. Se utilizan algunas instancias de los problemas de mayor complejidad obteniendo mejorías en el speedup y la eficiencia.

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

  En el artículo [38] abordan la paralelización del $\mathcal{MM}$AS sobre el TSP. Algunas características del algoritmo son levemente modificadas: no utilizan la reinicialización y la actualización del rastro de feromona se realiza con la mejor solución hasta el momento.

  Se implementan ejecuciones paralelas independientes y multicolonia con varias topologías: completamente conectado, hipercubo, anillo unidireccional y reemplazo del peor (solamente la colonia con la mejor solución envía a la colonia con la peor solución). Para las variantes multicolonia se utiliza un tiempo de comunicación fijo y se agrega la solución recibida, siempre y cuando sea mejor que la mejor solución hasta el momento.

  Se realiza la evaluación de las propuestas implementando versiones sincrónicas y asincrónicas para las variantes de multicolonia e incorporando una versión puramente secuencial. La evaluación se realiza en un cluster homogéneo, trabajando con 8 colonias sobre 8 procesadores.

  Se realiza solamente el estudio de la calidad de las soluciones halladas por las variantes constatándose que todas las variantes presentan mejores soluciones que la versión secuencial. La versión que obtiene los mejores resultados es la de ejecuciones paralelas independientes.

Los autores atribuyen este hecho a que los tiempos de ejecución son relativamente altos, teniendo las ejecuciones independientes múltiples búsquedas mientras que para las otras variantes las colonias convergen rápidamente a la misma solución. Comprueban esta afirmación reduciendo la frecuencia de la comunicación, obteniendo un impacto positivo sobre los resultados. Los autores establecen que para obtener mejores resultados, la comunicación tiene que ser más sofisticada y dependiente del tamaño del problema y del tiempo de ejecución.

Para obtener variantes más competitivas con la ejecución independiente sugieren utilizar mecanismos que eviten la convergencia prematura. En ese sentido proponen: incorporar mecanismos de reinicialización, utilizar criterios de aceptación de las soluciones migrantes (por ejemplo: cuando difieren menos de una cierta cantidad de componentes) y que cada colonia busque en diferentes regiones del espacio de búsqueda.

- **Two models of parallel ACO algorithms for the minimum tardy task problem(2007)**

  El artículo [1] es un trabajo continuación del artículo [2]. Se considera en este trabajo los modelos de ejecuciones paralelas independientes y el de islas con topologías de anillo y estrella bidireccional. La mecánica de funcionamiento de los modelos de islas es idéntico al descripto en [2].

  Las variantes fueron implementas utilizando la biblioteca MALLBA y evaluadas en un cluster de PCs. Para las pruebas se utilizaron 8 colonias y fueron utilizados 1, 2, 4 y 8 procesadores. Las variantes que requieren comunicación fueron implementadas en forma asincrónica.

  La calidad de los resultados obtenidos es un poco mejor para las variantes que involucran comunicación, siendo la cantidad de evaluaciones similar. La eficiencia es muy buena, obteniendo con 8 procesadores 98,5 % para ejecuciones independientes, 91 % para el modelo de islas con topología de anillo y 69,7 % para el modelo de islas con topología de estrella. También se aprecia que la cantidad de procesadores no impacta en la calidad de las soluciones y la influencia en la cantidad de evaluaciones es muy pequeña. Este hecho resulta esperable ya que la cantidad de colonias utilizadas es exactamente la misma.

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

  En el artículo [51] se aborda el problema Urban Bus Network Design (UBND) que consiste en la maximizar la densidad de viajes directos

en una red de ómnibus basado en la demanda. La variante de ACO utilizada calcula la cantidad de feromona a depositar por una solución considerando un término global y un término local que depende de la componente. En general se utiliza solamente un término global.

Se utiliza una multicolonia con topología de anillo unidireccional. Cada colonia utiliza su propia matriz de feromona y resuelven independientemente el mismo problema. Cada una cantidad fija de iteraciones se produce la migración de la mejor solución de cada colonia.

La evaluación se realiza en un cluster de 8 PCs. Se realizan estudios de la calidad y del tiempo de ejecución obteniendo importantes mejoras en ambas. No se realizan estudios de speedup ni de eficiencia.

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

  En el artículo [25] se evalúa una multicolonia de ACS (MACS). Se plantean algunas diferencias con otras implementaciones. Una diferencia menor es la definición de un modulo de intercambio que encapsula la comunicación entre las colonias. Cada colonia realiza su búsqueda en forma independiente y cada un intervalo fijo utiliza este módulo para comunicarse. La diferencia más importante es la utilización de un mecanismo adaptativo para la actualización global de la feromona. La actualización local es exactamente la misma que utiliza ACS, pero la global asigna pesos dependientes de la calidad de las soluciones obtenidas por una colonia con respecto al resto.

  La implementación de MACS es sincrónica y se utiliza como plataforma de las pruebas un cluster de PCs. Para su evaluación se plantean tres topologías que son estrella, hipercubo y anillo unidireccional. Previamente a la comparación con otras técnicas se comparan los resultados obtenidos al utilizar las distintas topologías, siendo la estrella la que obtiene los mejores resultados.

  Posteriormente se evalúa la calidad de las soluciones obtenidas por MACS (con topología de estrella) y ACS sobre el VRP. Para la comparación se utiliza un mismo esfuerzo prefijado, obteniendo muy buenos resultados MACS. Finalmente se realiza una comparación entre MACS y PACS (ver comentario del artículo [12]) sobre el TSP. Siendo mejor la calidad de las soluciones obtenidas por MACS.

Art	**Var**	**Prob**	**Modelo**	**Arq**
[48] (1998)	$\mathcal{MM}$AS	TSP	Ejecuciones paralelas independientes	SISD
[7] (1998)	AS	NA	Maestro-esclavo grano fino y multicolonia	NA
[49] (1999) y [50] (2001)	ANTabu	QAP	Maestro-esclavo grano fino	COW
[40] (2000)	AS	TSP	Multicolonia	ND
[41] (2002)	AS	TSP	Multicolonia	ND
[46] (2002)	ACS	TSP	Maestro-esclavo grano fino	Cluster???
[4] (2002)	ACS	TSP	Multicolonia	COW
[45] (2002)	ACO	SCP	Ejecuciones paralelas independientes y maestro-esclavo grano fino	COW
[44] (2002)	ACO	TSP	Multicolonia	Cray T3E????
[31] (2003)	ACO	MANET	ND	COW
[12] (2003)	ACS	TSP	Multicolonia	ND
[11] (2004)	ACS	TSP	Multicolonia	ND
[14] (2004)	ACO	TSP	Maestro-esclavo grano fino	Sun Fire 15K HPC con 48 procesadores???
[21] (2004)	AS_{rank}	VRP	Maestro-esclavo grano fino	COW
[2] (2005)	ACS	MTTP	Ejecuciones paralelas independientes, maestro-esclavo grano fino y multicolonia	COW
[15] (2005)	ACO	TSP	Maestro-esclavo grano fino con submaestros	Sun Fire 15K HPC con 48 procesadores???
[10] (2005)	ACO	PSPP	Maestro-esclavo grano fino y multicolonia	Cluster

Tabla 1: Resumen de aplicaciones de paralelismo sobre ACO

Art	**Var**	**Prob**	**Modelo**	**Arq**
[18] (2005)	ACS	TSP	Maestro-esclavo grano fino	Multiprocesador
[19] (2005)	ACS	TSP	Maestro-esclavo grano fino	Multiprocesador
[42] (2005)	ACO	OP	Multicolonia sin interacción (resuelven distinto problemas)	Grid
[29] (2005)	AS_{queen}	TSP	Multicolonia con control centralizado	COW
[9] (2005)	ACO	TSP	Multicolonia	MPP
[5] (2005)	ACO	VRP	Multicolonia, maestro-esclavo grano fino e híbrido	Cluster
[22] (2005)	ACO	VRP	Maestro-esclavo grano fino (resuelven distintos subproblemas)	Cluster
[38] (2006)	$\mathcal{MM}$AS	TSP	Ejecuciones paralelas independientes y multicolonia	COW
[23] (2006)	ACO	VRP	Multicolonia, maestro-esclavo grano fino, híbrido y maestro-esclavo grano fino (trabajan sobre diferentes subproblemas)	Cluster
[1] (2007)	ACS	MTTP	Ejecuciones paralelas independientes y Multicolonia	COW
[51] (2007)	ACO	UBND	Multicolonia	COW
[25] (2007)	ACS	VRP y TSP	Multicolonia	COW

Tabla 1: Resumen de aplicaciones de paralelismo sobre ACO

En la tabla 1 se presenta un resumen de las aplicaciones de paralelismo sobre ACO. En las columnas se indica: artículo y año de publicación (Art), variante de ACO utilizada (Var), problema abordado (Prob), estrategia de paralelismo utilizada (Modelo) y arquitectura utilizada (Arq).

El valor ACO en la columna Var indica que es una variante específica para el problema abordado o que no se especifica la variante utilizada. El valor NA en cualquier columna indica que no aplica o no corresponde. El valor ND en cualquier columna indica que no está disponible.

6. Otros trabajos sobre ACO

Los trabajos que se presentan en esta sección no son sobre paralelismo, pero por su temática incluyen aspectos interesantes. El primero de los trabajos es el primer artículo que aborda la utilización de un ACO multicolonia para resolver un problema. Los otros dos trabajos exploran ideas novedosas y no utilizadas habitualmente al trabajar con variantes de la clase multicolonia. Estos dos trabajos se incorporan por considerar que pueden ser de utilidad al diseñar nuevas implementaciones paralelas de ACO.

A continuación se presenta un resumen de esos artículos:

- **An Island Model Based Ant System with Lookahead for the Shortest Supersequence Problem (1998)**

  El artículo [39] aborda el problema Shortest Common Supersequence (SCS) mediante una variante de ACO específica para el problema. El algoritmo utiliza un modelo multicolonia pero no se establece que se realice una implementación paralela. En este caso, cada una cantidad fija de iteraciones las poblaciones intercambian las mejores soluciones. Al momento de realizar la actualización de la matriz de feromona se incorpora la mejor solución recibida hasta el momento siguiendo una estrategia elitista.

  Solamente se realizan estudios sobre la calidad de las soluciones obtenidas. En la comparación entre utilizar multicolonia o una única población, se observa que la calidad de las soluciones obtenidas por la primera es levemente superior.

- **Multiple Ant Colonies Algorithm Based on Colony Level Interactions (2000)**

  En el artículo [34] se presenta una propuesta de multicolonia en la cual la matriz de feromona de cada colonia puede tener efecto sobre las otras colonias. Cada colonia tiene su propia matriz y recibe efectos

positivos (aumentando la probabilidad de elegir una componente) o negativos (disminuyendo la probabilidad de elegir una componente) de las otras colonias. Evidentemente este mecanismo incorpora parámetros adicionales para reflejar la influencia de las distintas colonias.

Los autores evalúan la calidad de las soluciones obtenidas para el TSP por su propuesta contra AS, obteniendo su propuesta mejores resultados. En lo que respecta a los tiempos de ejecución, en instancias chicas los tiempos de ejecución de AS son mejores y en instancias grandes los tiempos de ejecución son similares. Estudian varias topologías con distintos efectos positivos y negativos, por ejemplo: jerárquico, todos contra todos y uno contra todos.

La posible paralelización de este algoritmo presenta algunas dificultades debido a que cada colonia tiene que conocer las matrices de feromona de las otras colonias. Una alternativa es realizar una implementación en memoria compartida para evitar el envío de las matrices.

- **Effective Diversification of Ant-Based Search Using Colony Fission and Extinction (2006)**

  En el artículo [28] se aborda una propuesta sobre multicolonia con la particularidad de que la cantidad de colonias varía en forma dinámica durante la ejecución del algoritmo.

  La propuesta se caracteriza por utilizar mecanismos de fisión de colonias, es decir a partir de una colonia se generan dos colonias, y de extinción, en los cuales a partir de dos colonias sobrevive solamente una.

  Si en una iteración t se mejora la mejor solución obtenida hasta el momento y difiere en una alta proporción a la previa, se produce la fisión en dos colonias, una con la información disponible en $t-1$ y otra con la información en t y la actualización de feromona correspondiente. La colonia nueva continúa normalmente su búsqueda, pero la colonia con información más vieja comienza una etapa de búsqueda dependiente de la nueva colonia hasta que encuentre una mejor solución que la que tenía. En esa etapa de búsqueda la colonia nueva influencia en forma negativa a la vieja para evitar que se produzcan soluciones similares.

  La extinción ocurre cuando dos colonias presentan mejores soluciones hasta el momento muy similares. El mecanismo propuesto consiste en eliminar la colonia con la solución de peor calidad, excepto que se haya creado en la iteración anterior o que las soluciones tengan la misma calidad.

Para la evaluación de la propuesta comparan la calidad de las soluciones encontradas contra las que obtienen mediante AS_{elite} y AS_{elite} con α negativo (es una variante muy poco difundida que incorpora aleatoriedad para evitar los óptimos locales). Evalúan la diversidad de las soluciones construidas y la calidad de las soluciones siendo en ambos casos ACO con fisión y extinción el mejor. A modo de ejemplo, para un problema de 70 ciudades llegan a existir 20 colonias trabajando a la vez para culminar con 5 soluciones (el promedio es 4.65 %). La tasa de solapamiento entre las 5 soluciones encontradas en inferior al 77 %.

Este artículo es novedoso porque propone un mecanismo dinámico para la creación y destrucción de colonias. Una posible implementación paralela de esta propuesta puede presentar algunas dificultades por ese motivo y por la inclusión de una etapa de búsqueda dependiente en la cual dos colonias actúan fuertemente acopladas.

Otros trabajos que utilizan un enfoque multicolonia pero que no realizan una implementación paralela son [27], [17] y [36]. En [27] se utiliza una multicolonia para resolver un problema multiobjetivo de ruteo de vehículos, cada colonia intenta minimizar uno de los objetivos. En [17] se aborda el problema de ubicar las paradas para líneas de ómnibus con una multicolonia en la cual cada colonia trabaja sobre una línea diferente en forma independiente. En [36] se incorpora una heurística para mejorar la mejor solución generada por las colonias en cada iteración al trabajar sobre un problema de asignación de armas a objetivos.

7. Conclusiones

Se ha realizado un relevamiento sobre las estrategias de paralelismo aplicado a ACO detectándose que existe un modelo utilizado ampliamente sobre otras metaheurísticas pero no ha sido utilizado sobre ACO. Por consiguiente se ha incorporado el modelo celular dentro de la taxonomía de estrategias de paralelismo aplicado a ACO. Tampoco existen propuestas de implementación del modelo celular. Dentro de las líneas de trabajo futuro está la implementación de un ACO paralelo siguiendo un modelo celular.

Se han realizado algunas correcciones menores tendientes a unificar categorías con la taxonomía de estrategias de paralelismo aplicado a AE. Esto puede permitir una aproximación más simple a la temática del paralelismo y ACO.

También se ha realizado un amplio relevamiento de trabajos en los cuales se implementan versiones paralelas y/o distribuidas de ACO. Es posible

apreciar un desarrollo desparejo entre las distintas categorías siendo multicolonia y maestro-esclavo grano fino los más utilizados. Como se ha señalado no existen implementaciones del modelo celular y los modelos híbridos son escasos. Tampoco existen implementaciones del modelo maestro-esclavo de grano más fino, lo cual pone en tela de juicio si realmente debe ser considerada como una categoría. También se ha experimentado con éxito con las ejecuciones paralelas independientes, pero no despierta ningún interés desde el punto de vista algorítmico. Parte de su éxito puede deberse a que con ejecuciones más largas es posible llegar a situaciones de estancamiento.

La mayor parte de las implementaciones ha sido desarrollada utilizando pasaje de mensajes sobre arquitecturas de tipo Cluster. Las experiencias en escenarios de equipos con memoria compartida son escasas.

El TSP ha sido el problema por excelencia para probar las diferentes propuestas. Recientemente ha comenzado a diversificarse el espectro de problemas abordados.

Los resultados de aplicar paralelismo sobre ACO pueden considerarse en general buenos. Algunas dudas han surgido sobre la conveniencia o no de utilizar estrategias sofisticadas cuando las ejecuciones paralelas independientes se han mostrado exitosas. Aunque como se ha señalado este hecho parece estar vinculado fuertemente a situaciones de estancamiento o pérdida de la diversidad de la búsqueda.

Bibliografía

[1] Enrique Alba, Guillermo Leguizamon, and Guillermo Ordoñez. Two models of parallel aco algorithms for the minimum tardy task problem. *International Journal of High Perfomance Systems Architecture*, 1(1):50–59, November 2007.

[2] Enrique Alba, Guillermo Leguizamón, and Guillermo Ordoñez. Analyzing the behavior of parallel ant colony systems for large instances of the task scheduling problem. In *19th International Parallel and Distributed Processing Symposium (IPDPS 2005), CD-ROM / Abstracts Proceedings, 4-8 April 2005, Denver, CA, USA*. IEEE Computer Society, 2005.

[3] Enrique Alba and Antonio J. Nebro. New Technologies in Parallelism. In Enrique Alba, editor, *Parallel Metaheuristics*, pages 63–78. Wiley, 2005.

[4] Benjamín Barán and Marta Almirón. Colonia de hormigas en un ambiente paralelo asíncrono. In *XXVIII Conferencia Latinoamericana de Informática CLEI - 2002*, Montevideo, Uruguay, 2002.

[5] Siegfried Benker, Karl Doerner, Richard F. Hartl, Guenter Kiechle, and Mária Lucká. Communication strategies for parallel cooperative ant colony optimization on clusters and grids. In Jens Gottlieb and Günther R. Raidl, editors, *Complimentary Proceedings of PARA'04 Workshop on State-of-the-Art in Scientific Computing, June 20-23, 2004*, pages 3–12, 2005.

[6] M. Bolondi and M. Bondaza. Parallelizzazione di un algoritmo per la risoluzione del problema del comesso viaggiatore (in Italian). Master's thesis, Politecnico di Milano, Italy, 1993.

[7] Bernd Bullnheimer, Gabriele Kotsis, and Christine Strauss. Parallelization Strategies for the Ant System. In R. De Leone, A. Murli, P. M. Pardalos, and G. Toraldo, editors, *High Performance Algorithms and Software in Nonlinear Optimization, Kluwer International Series in Applied Optimization*, volume 24, pages 263–308. Kluwer Academic Publisher, 1998.

[8] Erick Cantú-Paz. Theory of Parallel Genetic Algorithms. In Enrique Alba, editor, *Parallel Metaheuristics*, pages 423–445. Wiley, 2005.

[9] Ling Chen and Chunfang Zhang. Adaptive parallel ant colony algorithm. In L. Wang, K. Chen, and Y. Ong, editors, *Advances in Natural Computation, First International Conference, ICNC 2005, Changsha, China, August 27-29, 2005, Proceedings, Part II*, volume 3611 of *Lecture Notes in Computer Science*, pages 1239–1249. Springer, 2005.

[10] Daniel Chu, Michael Till, and Albert Y. Zomaya. Parallel ant colony optimization for 3d protein structure prediction using the hp lattice model. In *IPDPS '05: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) - Workshop 6*, page 193.2, Washington, DC, USA, 2005. IEEE Computer Society.

[11] Shu-Chuan Chu, John F. Roddick, and Jeng-Shyang Pan. Ant colony system with communication strategies. *Information Sciences*, 167(1-4):63–76, 2004.

[12] Shu-Chuan Chu, John F. Roddick, Jeng-Shyang Pan, and Che-Jen Su. Parallel ant colony systems. In N. Zhong, Z.W. Ras, S. Tsumoto, and E Suzuku, editors, *14th International Symposium on Methodologies for Intelligent Systems*, volume 2871 of *LNAI*, pages 279–284, Maebashi City, Japan, 2003. Springer.

[13] Teodor Gabriel Crainic and Nourredine Hail. Parallel Metaheuristics Applications. In Enrique Alba, editor, *Parallel Metaheuristics*, pages 447–494. Wiley, 2005.

[14] Mitica Craus and Laurentiu Rudeanu. Parallel framework for ant-like algorithms. In *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), 5-7 July 2004, Cork, Ireland*, pages 36–41. IEEE Computer Society, 2004.

[15] Mitica Craus and Laurentiu Rudeanu. Multi-level parallel framework. *International Scientific Journal of Computing*, 4, 2005.

[16] Van-Dat Cung, Simone L. Martins, Celso Ribeiro, and Catherine Roucairol. Strategies for the Parallel Implementation of Metaheuristics. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 263–308. Kluwer Academic Publisher, 2001.

[17] Jasper de Jong and Marco Wiering. Multiple ant colony system for the busstop allocation problem. In *BNAIC'01: Proceedings of the Thirteenth*

Belgium-Netherlands Conference on Artificial Intelligence, pages 141–148, 2001.

[18] Pierre Delisle, Marc Gravel, Michaël Krajecki, Caroline Gagné, and Wilson L. Price. Comparing parallelization of an aco: Message passing vs. shared memory. In M. J. Blesa, C. Blum, A. Roli, and M. Sampels, editors, *Hybrid Metaheuristics, Second International Workshop, HM 2005, Barcelona, Spain, August 29-30, 2005, Proceedings*, volume 3636 of *Lecture Notes in Computer Science*, pages 1–11. Springer, 2005.

[19] Pierre Delisle, Marc Gravel, Michaël Krajecki, Caroline Gagné, and Wilson L. Price. A shared memory parallel implementation of ant colony optimization. In *Proceedings of the 6th Metaheuristics International Conference (MIC'2005), Vienne, Autriche, August 22-26, 2005*, pages 257–264, 2005.

[20] Pierre Delisle, Michaël Krajecki, Marc Gravel, and Caroline Gagné. Parallel implementation of an ant colony optimization metaheuristic with openmp. In *In International conference of parallel architectures and complication techniques (PACT), Proceedings of the Third European workshop on OpenMP, Barcelona, Spain*, pages 8–12, September 8-9 2001.

[21] Karl Doerner, Richard F. Hartl, Guenter Kiechle, Mária Lucká, and Marc Reimann. Parallel ant systems for the capacitated vehicle routing problem. In Jens Gottlieb and Günther R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization, 4th European Conference, EvoCOP 2004, Coimbra, Portugal, April 5-7, 2004, Proceedings*, volume 3004 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2004.

[22] Karl Doerner, Richard F. Hartl, and Mária Lucká. A parallel version of the d-ant algorithm for the vehicle routing problem. In M.Vajtersic A.Uhl R.Trobec, P.Zinterhof, editor, *Proceedings of the International Workshop on Parallel Numerics 2005*, pages 109–118, Portoroz, Slovenia, April 20-23 2005.

[23] Karl F. Doerner, Richard F. Hartl, Siegfried Benkner, and Mária Lucká. Parallel cooperative savings based ant colony optimization - multiple search and decomposition approaches. *Parallel Processing Letters*, 16(3):351–370, 2006.

[24] Marco Dorigo and Thomas Stützle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, USA, 2004.

[25] Ismail Ellabib, Paul H. Calamai, and Otman A. Basir. Exchange strategies for multiple ant colony system. *Information Sciences*, 177(5):1248–1264, 2007.

[26] Michael Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9):948–960, 1972.

[27] Luca Maria Gambardella, Éric Taillard, and Giovanni Agazzi. Macs-vrptw: a multiple ant colony system for vehicle routing problems with time windows. pages 63–76, 1999.

[28] Akira Hara, Takumi Ichimura, Nobuyuki Fujita, and Tetsuyuki Takahama. Effective diversification of ant-based search using colony fission and extinction. In G. Yen, S.Lucas, G. Fogel, G. Kendall, R. Salomon, B. Zhang, C. Coello Coello, and T. Runarsson, editors, *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 1028–1035, Vancouver, BC, Canada, 16-21 July 2006. IEEE Press.

[29] Ichiro Iimura, Koji Hamaguchi, Toshiya Ito, and Shigeru Nakayama. A study of distributed parallel processing for queen ant strategy in ant colony optimization. In *PDCAT '05: Proceedings of the Sixth International Conference on Parallel and Distributed Computing Applications and Technologies*, pages 553–557. IEEE Computer Society, 2005.

[30] Ichiro Iimura, Takafumi Matsudome, and Shigeru Nakayama. Consideration on queen ant strategy in ant colony optimization. *IEICE Transactions on Information and Systems, Pt.1 (Japanese Edition)*, J88-D-1(10):1599–1602, 2005.

[31] Mohammad Islam, Parimala Thulasiraman, and Ruppa Thulasiram. A parallel ant colony optimization algorithm for all-pair routing in manets. In *17th International Parallel and Distributed Processing Symposium (IPDPS 2003), 22-26 April 2003, Nice, France, CD-ROM/Abstracts Proceedings*, page 259. IEEE Computer Society, 2003.

[32] Stefan Janson, Daniel Merkle, and Martin Middendorf. Parallel Ant Colony Algorithms. In Enrique Alba, editor, *Parallel Metaheuristics*, pages 171–201. Wiley, 2005.

[33] Alan H. Karp and Horace P. Flatt. Measuring parallel processor performance. *Commun. ACM*, 33(5):539–543, 1990.

[34] Hidenori Kawamura, Masahito Yamamoto, Keiji Suzuki, and Azuma 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.

[35] F. Krüger, M. Middendorf, and D. Merkle. Studies on a Parallel Ant System for the BSP Model. Unpublished manuscript, 1998.

[36] Zne-Jung Lee, Chou-Yuan Lee, and Shun-Feng Su. Parallel ant colonies with heuristics applied to weapon-target assignment problems. In *Proceedings of the 7h Conference on Artificial Intelligence and Applications, TAAI2002, Taichung, Taiwan, November 15, 2002*, pages 201–206, 2002.

[37] Gabriel Luque, Enrique Alba, and Bernabé Dorronsoro. Parallel Genetic Algorithms. In Enrique Alba, editor, *Parallel Metaheuristics*, pages 105–125. Wiley, 2005.

[38] Max Manfrin, Mauro Birattari, Thomas Stützle, and Marco Dorigo. Parallel ant colony optimization for the traveling salesman problem. In *Ant Colony Optimization and Swarm Intelligence, 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings*, volume 4150 of *Lecture Notes in Computer Science*, pages 224–234. Springer, 2006.

[39] René Michel and Martin Middendorf. An island model based ant system with lookahead for the shortest supersequence problem. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 692–701, London, UK, 1998. Springer-Verlag.

[40] Martin Middendorf, Frank Reischle, and Hartmut Schmeck. Information exchange in multi colony ant algorithms. In José D. P. Rolim, editor, *Parallel and Distributed Processing, 15 IPDPS 2000 Workshops, Cancun, Mexico, May 1-5, 2000, Proceedings*, Lecture Notes in Computer Science, pages 645–652. Springer, 2000.

[41] Martin Middendorf, Frank Reischle, and Hartmut Schmeck. Multi colony ant algorithms. *Journal of Heuristics*, 8:305–320, 2002.

[42] José Mocholí, Javier Martínez, and José Canós. A grid ant colony algorithm for the orienteering problem. In *Congress on Evolutionary Computation*, pages 942–949. IEEE, 2005.

[43] Martín Pedemonte. Ant colony optimization. Reporte técnico 07-11, InCo, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay, Agosto 2007.

[44] Douglas Piriyakumar and Paul Levi. A new approach to exploiting parallelism in ant colony optimization. In *International Symposium on Micromechatronics and Human Science (MHS)*, pages 237–243. Nagoya, Japan: n. n., January 2002.

[45] Malek Rahoual, Riad Hadji, and Vincent Bachelet. Parallel ant system for the set covering problem. In *ANTS '02: Proceedings of the Third International Workshop on Ant Algorithms*, pages 262–267, London, UK, 2002. Springer-Verlag.

[46] Marcus Randall and Andrew Lewis. A parallel implementation of ant colony optimization. *Journal of Parallel and Distributed Computing*, 62(9):1421–1432, 2002.

[47] Sartaj Sahni and Venkat Thanvantri. Parallel computing: Performance metrics and models. Research Report 96-008, University of Florida, Dept. Computer & Information Sci. & Engr., 1996.

[48] Thomas Stützle. Parallelization strategies for ant colony optimization. *Lecture Notes in Computer Science*, 1498:722–731, 1998.

[49] El-Ghazali Talbi, Olivier Roux, Cyril Fonlupt, and Denis Robillard. Parallel Ant Colonies for Combinatorial Optimization Problems. In J. Rolim, editor, *Workshop on Biologically Inspired Solutions to Parallel Processing Systems*. Springer-Verlag, 1999.

[50] El-Ghazali Talbi, Olivier Roux, Cyril Fonlupt, and Denis Robillard. Parallel Ant Colonies for the Quadratic Assignement Problem. *Future Generation Computer Systems*, 17(4):441–449, 2001.

[51] Zhongzhen Yang, Bin Yu, and Chuntian Cheng. A parallel ant colony algorithm for bus network optimization. *Computer-Aided Civil and Infrastructure Engineering*, 22(1):44–55, January 2007.