Parallel Ant Colony Optimization

Martín Pedemonte y Héctor Cancela

{mpedemon,cancela}@fing.edu.uy INCO, Facultad de Ingeniería, UdelaR

Abstract. The application of parallel programming techniques to Ant Colony Optimization is incipient and has not completed its maturation. Although the first proposals date back to ACO's origin, research in this area has grown in recent years. This article highlights applications of parallel ACO in order to provide a summary of the proposals in the literature. In addition to the survey itself, we propose a new taxonomy of parallelization strategies for ACO that improves some aspects of the previously existing ones, incluiding the cellular model that has not been applied to ACO yet.

1. Introducción

Ant Colony Optimization (ACO) es una de las técnicas metaheurísticas bien desarrolladas y conocidas en la comunidad [17], sin embargo el estudio sistemático de la aplicación de técnicas de programación paralela a esta técnica es reciente y varios autores coinciden en que no se ha realizado un trabajo exhaustivo [17,20]. Incluso Dorigo y Stützle [17] señalan que aunque existe 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 con respecto a las versiones secuenciales.

Este trabajo aborda el estudio de las taxonomías de estrategias para la paralelización de ACO propuestas y el relevamiento de aplicaciones de paralelismo a ACO, centrado en trabajos realizados sobre clusters de PCs y equipos de memoria compartida. El resto del artículo se estructura de la siguiente forma. Las taxonomías que han sido presentadas en la literatura existente para la paralelización de ACO se describen en la sección 2; en esa misma sección se discuten algunas carencias de esas taxonomías, y se presenta una nueva propuesta, que incluye la identificación de un modelo de paralelismo no aplicado aún a ACO. Posteriormente, en la sección 3 se reseñan las principales aplicaciones de paralelismo a ACO, organizadas de acuerdo a la taxonomía propuesta. Finalmente, en la sección 4 se presentan las conclusiones de este trabajo.

2. Taxonomía de paralelismo aplicado a ACO

Randall y Lewis [28] clasificaron las estrategias de paralelismo aplicado a ACO existentes en cinco categorías. *Colonias de hormigas paralelas independientes* corresponde a la ejecución paralela e independiente de un algoritmo ACO

secuencial sobre un conjunto de procesadores. *Colonias de hormigas paralelas con interacción* consiste en la ejecución en paralelo de un algoritmo ACO secuencial. Las colonias con una cierta frecuencia sincronizan sus matrices de feromona con la de la mejor colonia hasta el momento. *Hormigas paralelas* corresponde a una colonia que trabaja en paralelo usando un modelo de comunicación maestro-esclavo. El proceso maestro maneja la matriz de feromona, mientras que cada proceso esclavo construye una solución. La comunicación involucra el envío de las soluciones 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* corresponde a la ejecución secuencial de una colonia en la que cada hormiga evalúa en paralelo las componentes que puede incorporar a la solución. El modelo de comunicación es maestro-esclavo. Finalmente, *Combinación de hormigas paralelas y evaluación de componentes de soluciones* cuenta con los dos niveles de paralelismo maestro-esclavo descritos previamente.

Janson et al. [20] presentan algunos aspectos relevantes sobre la paralelización de ACO. En primer lugar, proponen clasificar las propuestas distinguiendo si el paralelismo es realizado sobre un algoritmo ACO estándar o si el algoritmo ACO paralelo está diseñado especialmente y si el enfoque es centralizado o descentralizado. Sin embargo, estos criterios no permite clasificar a todas las propuestas, ya que en la práctica estos aspectos están también interrelacionados. Finalmente, comentan las particularidades del *Modelo multicolonia* (varias colonias de hormigas con matrices de feromona independientes que intercambian información con un cierto patrón predeterminado). Los aspectos señalados como relevantes para el diseño de un algoritmo multicolonia son: la estructura de comunicación y topología, el tipo de información intercambiada entre las colonias, la forma de utilización de la información recibida de otras colonias, la cadencia de la comunicación y si los parámetros son homogéneos o heterogéneos.

El enfoque presentado por Janson y sus coautores para el *Modelo multicolonia* es más amplio que el de Randall y Lewis para las *Colonias de hormigas paralelas con interacción*, ya que permite agrupar una mayor cantidad de propuestas en la categoría al flexibilizar la restricción de sincronizar las matrices de feromona.

Es posible señalar algunas carencias en la taxonomía de Randall y Lewis. Si se compara con la taxonomía de paralelismo aplicado a Evolutionary Algorithms aceptada por la comunidad académica [21], es posible detectar la ausencia de categorías similares al *Modelo celular* (una población estructurada en vecindades pequeñas con interacciones limitadas) y *Modelo híbrido* (con características de más de un modelo). Por otro lado, la restricción de sincronizar las matrices de feromona en la categoría *Colonias de hormigas paralelas con interacción* restringe el agrupamiento de una mayor cantidad de propuestas similares. Finalmente, el relevamiento de implementaciones de ACO paralelos reveló la existencia de un modelo maestro-esclavo que no integraba la taxonomía original.

En base a estas observaciones, en este trabajo se propone una taxonomía de estrategias para implementaciones de ACO paralelos que corrige las carencias detectadas. Esta nueva taxonomía cuenta con las siguientes categorías: *Ejecuciones paralelas independientes*, *Modelo maestro-esclavo*, *Modelo multicolonia*,

Modelo celular y *Modelo híbrido*. A continuación describimos en detalle estas categorías.

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

Modelo maestro-esclavo: es posible dividirla en tres subcategorías. La primer subcategoría es un modelo maestro-esclavo (M-E1) que corresponde a una descomposición de dominio del problema. Los esclavos resuelven subproblemas en forma independiente, mientras que el maestro maneja la información global del problema, encargándose de construir una solución completa a partir de las soluciones parciales. La segunda subcategoría es un modelo maestro-esclavo (M-E2) en el que los esclavos se encargan de construir soluciones completas. El maestro se encarga del manejo de la matriz de feromona y cada uno de los esclavos comunica su(s) solución(es) al maestro. La subcategoría anteriormente descrita es más amplia que la categoría *Hormigas paralelas* de Randall y Lewis. La tercera subcategoría (M-E3) se corresponde con la categoría *Evaluación en paralelo de componentes de soluciones* de Randall y Lewis.

Modelo multicolonia: se corresponde con el modelo multicolonia descrito por Janson y sus coautores. *Colonia de hormigas paralelas con interacción* de Randall y Lewis está comprendida en este modelo.

Modelo celular: trabaja sobre una única población estructurada en vecindades pequeñas con solapamientos. Cada hormiga interactúa solamente con las hormigas vecinas, teniendo cada vecindad su propia matriz de feromona. La actualización en cada matriz de feromona considera solamente las soluciones de su vecindad. Actualmente no existen implementaciones de este modelo.

Modelo híbrido: corresponde a implementaciones paralelas que usan más de un modelo. La categoría *Combinación de hormigas paralelas y evaluación de componentes de soluciones* de Randall y Lewis es un caso particular de este modelo.

3. Aplicaciones de paralelismo a ACO

En esta sección se presenta un resumen de artículos relevantes sobre la implementación de ACO paralelos, organizados de acuerdo a las categorías de la taxonomía propuesta en la sección precedente.

3.1. Trabajos pioneros

La primera implementación de un algoritmo ACO paralelo se atribuye a Bolondi y Bondaza [5] y consistía en colocar una hormiga en cada procesador. Por este motivo, el algoritmo no escalaba al aumentar la cantidad de procesadores considerados. Posteriormente, Bullnheimer et al. [6] evaluaron el efecto de las comunicaciones en versiones sincrónica y parcialmente asincrónica del

modelo maestro-esclavo M-E2. El estudio fue realizado mediante una simulación, obteniendo la versión parcialmente asincrónica un mejor desempeño que la sincrónica. Finalmente, Stützle [29] evaluó la conveniencia de realizar ejecuciones paralelas independientes o una única ejecución de mayor tiempo. Las ejecuciones paralelas independientes obtuvieron mejores soluciones que el algoritmo secuencial en todos los casos.

3.2. Modelo maestro-esclavo

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

- **[30]:** Talbi y sus coautores estudiaron un maestro-esclavo M-E2 sincrónico. La propuesta tiene la particularidad que los esclavos aplican un operador de búsqueda local. La evaluación fue realizada sobre un cluster de 10 PCs. Los autores solamente evaluaron la calidad de las soluciones obtenidas, mostrando resultados superiores a otras técnicas evaluadas.
- **[28]:** Randall y Lewis evaluaron un maestro-esclavo M-E2 con la particularidad que el maestro realiza para cada hormiga una actualización local del rastro de feromona. Cada esclavo envía al maestro cada componente qué incluye en la solución mientras que el maestro responde con la actualización local correspondiente. La evaluación realizada en un cluster de 8 PCs mostró un speedup pobre, logrando la eficiencia máxima al usar dos procesadores. Los malos resultados obtenidos muestran que el enfoque usado para la actualización local no es un esquema eficiente.
- **[10,11]:** Craus y Rudeanu [10] implementaron un framework para la ejecución de algoritmos secuenciales en ambientes paralelos siguiendo un modelo maestro-esclavo. Los esclavos solicitan intercambios de datos con el maestro por turnos en forma asíncrona. Solamente se intercambia la información modificada entre las solicitudes del esclavo. El framework fue evaluado instanciando un ACO sobre un equipo de memoria compartida con 48 procesadores. Se comprobó que el speedup era casi lineal hasta con 25 procesadores, degradándose al aumentar la cantidad de procesadores. Posteriormente, los mismos autores [11] implementaron un framework piramidal que reduce la comunicación al maestro por la incorporación de submaestros que actúan como concentradores de los esclavos que están bajo su órbita. Se comprobó que el speedup se mantenía casi lineal con más de 25 procesadores.
- **[15]:** Doerner y sus coautores evaluaron un modelo maestro-esclavo M-E2 con más de una hormiga por procesador. La evaluación se realizó sobre un cluster de 32 procesadores. Los resultados obtenidos indicaron que el speedup fue sublineal y la eficiencia decreció al aumentar la cantidad de procesadores. Los autores señalaron que un buen compromiso entre el desempeño y la cantidad de procesadores se da al usar 8 procesadores.
- **[12,13]:** Delisle et al. [12] siguieron un enfoque similar al de Randall y Lewis [28] sobre un equipo de memoria compartida de 16 procesadores. Los autores

comprobaron que su implementación logra un mejor desempeño computacional que la de Randall y Lewis, manteniendo la calidad de las soluciones obtenidas. Sin embargo, los trabajos no son totalmente comparables, ya que su implementación solamente sincroniza periódicamente las actualizaciones locales de los rastros de feromona. Posteriormente, los mismos autores [13] ampliaron su evaluación, comprobando que el desempeño del algoritmo con una misma cantidad de procesadores variaba al usar distintos equipos y tipos de nodos. Esta experiencia muestra que pueden existir limitaciones en el desempeño que sean atribuibles a razones tecnológicas y no al algoritmo.

- **[16]:** Doerner et al. evaluaron un maestro-esclavo M-E1 en el que la matriz de feromona se maneja en forma global, realizando el maestro la evaporación y el refuerzo de la mejor solución hasta el momento. La evaluación se realizó en un cluster de 16 procesadores, comprobándose que es posible mantener la calidad de las soluciones reduciendo el tiempo de ejecución.
- **[25]:** Mocholí y sus coautores implementaron un maestro-esclavo M-E1. La implementación fue desarrollada en un entorno grid con los servicios de alto nivel provistos por web services. La evaluación en un cluster de 32 PCs mostró que al aumentar la cantidad de colonias decrece exponencialmente el tiempo de ejecución mientras que mejoran las soluciones obtenidas.
- **[19]:** Iimura y sus coautores propusieron un maestro-esclavo M-E2 que funciona como un conjunto de colonias con un control centralizado. Los esclavos trabajan independientemente sobre el mismo problema, enviando la mejor solución encontrada al maestro. El maestro, a partir de la calidad de las soluciones obtenidas, envía directivas a los esclavos para diversificar o explotar la búsqueda. La evaluación experimental fue realizada en un cluster de 9 PCs mostrando que un aumento en la cantidad de esclavos provoca mejoras leves en las soluciones con una reducción en el tiempo de ejecución.
- **[32]** Tsutsui y Liu propusieron un modelo maestro-esclavo M-E2 en el que el maestro construye soluciones y los esclavos las mejoran ejecutando una búsqueda local. La evaluación se realizó en dos PCs con cuatro núcleos cada una, mostrando que es posible lograr mejoras significativas en el tiempo de ejecución, si el overhead introducido por las comunicaciones es bajo.

3.3. Modelo multicolonia

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

- **[23,24]:** Michel y Middendorf [23] propusieron el modelo multicolonia en el que cada colonia usa su propia matriz de feromona. Las colonias intercambian las mejores soluciones con una frecuencia fija. Al producirse la actualización de la matriz de feromona, cada colonia incorpora la mejor solución recibida hasta el momento. El estudio comparativo mostró que el modelo multicolonia obtiene soluciones levemente superiores a las de una única colonia. Posteriormente, Middendorf et al. [24] mostraron la conveniencia de usar un modelo

multicolonia ante una multicolonia sin intercambio y una colonia. Los autores analizaron la frecuencia de intercambio, afirmando que debe existir un equilibrio para no degradar la calidad de las soluciones obtenidas.

- **[3]:** Barán y Almirón estudiaron una implementación asincrónica del modelo multicolonia con topología de conexión completa. La evaluación fue realizada en un cluster de 4 PCs logrando un speedup lineal con resultados levemente superior a los del algoritmo serial.
- **[26]:** Piriyakumar y Levi estudiaron un modelo multicolonia sincrónico en el que cada colonia actualiza localmente el rastro de feromona y con una frecuencia fija se realiza una actualización global en forma conjunta. La evaluación experimental realizada en una computadora masivamente paralela mostró que es posible obtener resultados de buena calidad manteniendo acotados el tiempo ocioso de los procesadores y el tiempo de comunicación.
- **[9]:** Chu y sus coautores estudiaron un modelo multicolonia en el que cada colonia actualiza independientemente su matriz de feromona y en las iteraciones en que recibe soluciones de otras colonias incorpora una actualización adicional. La evaluación mostró que el modelo multicolonia obtiene mejores resultados que algunos algoritmos ACO secuenciales.
- **[7]:** Chen y Zhang propusieron un modelo multicolonia que intercambia las mejores soluciones entre pares de colonias en forma adaptativa ajustando la frecuencia de los intercambios según la bondad de las soluciones obtenidas. La evaluación experimental fue realizada en una computadora masivamente paralela constatándose que el uso del mecanismo adaptativo permite obtener soluciones de mejor calidad que usando intervalos fijos, aunque a costa de un mayor tiempo de ejecución. El estudio mostró que el speedup era sublineal, mejorando al aumentar el tamaño de las instancias consideradas.
- **[33]:** Yang y sus coautores evaluaron un modelo multicolonia con una topología de interconexión de anillo unidireccional. La evaluación realizada en un cluster de 8 PCs mostró que en un menor tiempo de ejecución se obtuvieron soluciones de calidad superior a las obtenidas con una sola colonia.
- **[18]:** Ellabib y sus coautores plantearon un modelo multicolonia con un mecanismo adaptativo para la actualización global de feromona que asigna pesos dependientes de la calidad de las soluciones obtenidas por cada colonia con respecto al resto. Se evaluaron implementaciones sincrónicas con distintas topologías de interconexión diferentes sobre un cluster de 8 PCs. La evaluación permitió comprobar que el modelo paralelo obtiene resultados superiores a algunos algoritmos ACO secuenciales y paralelos.

3.4. Más de un modelo implementado

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

- **[27]:** Rahoual y sus coautores estudiaron implementaciones de ejecuciones paralelas independientes y del modelo maestro-esclavo M-E2. La evaluación

fue realizada en un cluster de 40 PCs. Las ejecuciones paralelas independientes lograron un speedup casi lineal, debido a la escasa comunicación entre procesos. La eficiencia del modelo maestro-esclavo mostró una gran variabilidad dependiendo del tamaño de las instancias consideradas.

- **[1,2]:** Alba et al. [1] estudiaron implementaciones de: ejecuciones paralelas independientes, multicolonia con topología de interconexión de anillo en versiones sincrónica y asincrónica, y maestro-esclavo M-E2 en versiones sincrónica y asincrónica. La evaluación se realizó en un cluster de 3 PCs obteniendo soluciones similares para todas las implementaciones. Los tiempos de ejecución de las variantes asincrónicas fueron menores que los de las variantes sincrónicas, mientras que las ejecuciones paralelas independientes requirieron el menor tiempo por la ausencia de comunicación. Posteriormente, Alba et al. [2] ampliaron su estudio para los modelos de ejecuciones paralelas independientes y multicolonia asincrónico con topologías de interconexión de anillo y estrella bidireccional. La evaluación experimental se realizó sobre un cluster de 8 PCs comprobándose que la calidad de las soluciones obtenidas es mejor para las variantes que involucran comunicación. El speedup es casi lineal para las ejecuciones independientes y el modelo multicolonia con topología de anillo, y sublineal para el modelo multicolonia con topología de estrella.
- **[8]:** Chu y sus coautores estudiaron los siguientes modelos de paralelismo: maestro-esclavo M-E2, multicolonia con intercambio circular de soluciones y multicolonia con matriz de feromona compartida. La evaluación experimental fue realizada en un cluster de 5 PCs, usando como métrica la cantidad de tics de CPU necesarios para encontrar el óptimo. Los resultados indicaron que el mejor desempeño lo obtuvo la multicolonia con intercambio de soluciones seguida de la multicolonia con matriz de feromona compartida.
- **[4,14]:** Benker et al. [4] estudiaron un maestro-esclavo M-E2 y un modelo híbrido jerárquico (una multicolonia en la que cada colonia es un maestro-esclavo). La evaluación realizada en un cluster de 64 procesadores mostró que el maestro-esclavo mejora su eficiencia al trabajar con instancias de tamaño creciente del problema. A pesar de ello, al considerarse más de 8 procesadores la eficiencia comienza a degradarse. Para el modelo híbrido, solamente se estudió la calidad de las soluciones y el tiempo de ejecución. Posteriormente, Doerner et al. [14] ampliaron su estudio incorporando un maestro-esclavo M-E1. La evaluación experimental comprobó que se producen mejoras en la eficiencia al crecer el tamaño de las instancias.
- **[22]:** Manfrin y sus coautores estudiaron los modelos de ejecuciones paralelas independientes y multicolonia en versiones sincrónicas y asincrónicas. La evaluación experimental se realizó en un cluster de 8 PCs constatándose que las implementaciones estudiadas obtuvieron soluciones superiores a la versión secuencial, destacándose las ejecuciones paralelas independientes por obtener los mejores resultados. Los autores atribuyeron este hecho a que el tiempo de ejecución es relativamente alto, teniendo las ejecuciones independientes múltiples búsquedas mientras que en las multicolonias, todas las colonias convergen rápidamente a la misma solución. Para comprobar esa afirmación, redujeron la frecuencia de comunicación entre las colonias obte-

niendo mejoras en los resultados. A partir de este experimento, establecieron que la comunicación entre las colonias debería ser dependiente del tamaño del problema y del tiempo de ejecución. Otras mejoras sugeridas para evitar la convergencia prematura en el modelo multicolonia son: incorporar mecanismos de reinicialización, usar otros criterios de aceptación de las soluciones y que cada colonia explore diferentes regiones del espacio de búsqueda.

- **[31]:** Tsutsui evaluó implementaciones de los siguientes modelos: maestro-esclavo, multicolonia con topología de conexión completa y ejecuciones paralelas independientes. La evaluación se realizó sobre un equipo con 4 núcleos. El estudio mostró que los modelos paralelos obtuvieron mejores soluciones que el algoritmo secuencial, destacándose el modelo maestro-esclavo por presentar los mejores resultados.

3.5. Resumen del relevamiento

La Tabla 1 presenta un resumen del relevamiento realizado, incluyendo la cantidad de trabajos que implementan algoritmos en cada categoría de la taxonomía propuesta, así como las citas a los trabajos en sí mismos. De dicha tabla, surge claramente que los modelos maestro-esclavo y multicolonia son los que más han sido explorados en la literatura. La categoría ejecuciones paralelas independientes no presenta una literatura tan nutrida; quizás por ser la más trivial en su implementación, no ha resultado un desafío conceptual interesante, si bien los resultados obtenidos con este modelo tan simple han sido relativamente muy buenos en comparación con los otros modelos. A pesar de que en varios trabajos se plantea la posibilidad de implementar modelos híbridos solamente uno de los trabajos implementa dicho modelo. Como se ha hecho notar anteriormente, la categoría modelo celular por el momento no ha sido explorada en la literatura, si bien pensamos puede ser una alternativa interesante.

Modelo	**#**	**Citas**
Ejec. paralelas independientes	6	[1,2,22,27,29,31]
Modelo maestro-esclavo	18	[1,4,6,8,10,11,12,13,14,15,16,19,25,27,28,30,31,32]
Modelo multicolonia	13	[1,2,3,7,8,9,18,22,23,24,26,31,33]
Modelo celular	0	No hay trabajos que lo implementen.
Modelo híbrido	1	[4]

Tabla 1. Resumen del relevamiento de algoritmos ACO paralelos

4. Conclusiones

En este artículo se presentó una reseña de estrategias de paralelización de ACO. Si bien hay una buena cantidad de trabajos y aplicaciones de paralelismo para ACO, no existe una taxonomía ampliamente reconocida por la comunidad

académica. La taxonomía de Randall y Lewis no es del todo satisfactoria, por lo cual se propuso una ampliación que subsana algunos problemas de la misma e incorpora una categoría novedosa, el modelo celular.

Se realizó un amplio relevamiento de trabajos que implementan versiones paralelas de ACO, centrado en trabajos realizados sobre clusters de PCs y equipos de memoria compartida. A partir del relevamiento realizado, es posible concluir que la aplicación de paralelismo a ACO es promisoria, existiendo una cantidad considerable de trabajos que muestran que es posible obtener mejoras en la calidad de las soluciones obtenidas en comparación con los algoritmos secuenciales, reducir el tiempo de ejecución y obtener speedups casi lineales.

En particular, los modelos que han sido más ampliamente usados en la práctica y que han mostrado su potencial son el maestro-esclavo y el multicolonia. Por otro lado, las ejecuciones paralelas independientes se han mostrado como una alternativa sencilla para la implementación del paralelismo, obteniendo speedup casi lineales y mejoras en la calidad de las soluciones obtenidas en comparación con los algoritmos secuenciales. Algunos autores han planteado dudas sobre la conveniencia o no de utilizar estrategias sofisticadas, cuando el modelo más simple permite alcanzar buenos resultados. Sin embargo, el éxito parece estar vinculado a que en el modelo de ejecuciones paralelas independientes se realizan varias exploraciones cortas del espacio de búsqueda, evitando así situaciones que se pueden producir en recorridas largas, como son alcanzar situaciones de estancamiento o de pérdida de diversidad de la búsqueda.

Es posible identificar dos líneas de trabajo futuro interesantes. La primera de ellas consiste en la implementación de un modelo celular para ACO, tarea en la que nos encontramos trabajando actualmente. La segunda línea de trabajo corresponde al estudio comparativo de las ejecuciones paralelas independientes respecto a otros modelos con el fin de entender los motivos de su éxito.

Referencias

1. Alba, E., Leguizamon, G. y Ordoñez, G.: Analyzing the Behavior of Parallel Ant Colony Systems for Large Instances of the Task Scheduling Problem. Proceedings of the 19th IPDPS, 14, IEEE Computer Society, 2005.
2. Alba, E., Leguizamon, G. y Ordoñez, G.: Two Models of Parallel ACO Algorithms for the Minimum Tardy Task Problem. IJHPSA, vol. 1, no. 1, 50–59, 2007.
3. Barán, B. y Almirón, M.: Colonia de Hormigas en un Ambiente Paralelo Asíncrono. XXVIII Conferencia Latinoamericana de Informática - CLEI 2002, 2002.
4. Benker, S., Doerner, K., Hartl, R., Kiechle, G. y Lucká, M: Communication Strategies for Parallel Cooperative ACO on Clusters & Grids. Proc. PARA04, 3–12, 2005
5. Bolondi, M. y Bondaza, M.: Parallelizzazione di un Algoritmo per la Risoluzione del Problema del Comesso Viaggiatore. Tesis de Maestría, Pol. di Milano, 1993.
6. Bullnheimer, B., Kotsis, G. y Strauss, C.: Parallelization Strategies for the Ant System. Applied Optimization, v. 24, 263–308, 1998.
7. Chen, L. y Zhang, C.: Adaptive Parallel Ant Colony Algorithm. Proceedings of the ICNC 2005, Part II, LNCS, vol. 3611, 1239–1249, Springer, 2005.
8. Chu, D., Till, M. y Zomaya, A.: Parallel Ant Colony Optimization for 3D Protein Structure Prediction using the HP Lattice Model. Proc. of 19th IPDPS, 193b, 2005.

9. Chu, S-C., Roddick, J.F. y Pan, J-S.: Ant Colony System with Communication Strategies. Information Sciences, vol. 167, no. 1-4, 63–76, 2004.
10. Craus, M. y Rudeanu, L.: Parallel Framework for Ant-Like Algorithms, Proceedings of the 3rd ISPDC and 3rd HeteroPar, 36–41, IEEE Computer Society, 2004.
11. Craus, M. y Rudeanu, L.: Multi-Level Parallel Framework. International Scientific Journal of Computing, vol. 4, no. 1, 2005.
12. Delisle, P., Gravel, M., Krajecki, M., Gagné, C. y Price, W.: Comparing Parallelization of an ACO: Message Passing vs. Shared Memory. Proceedings of the 2nd HM, LNCS, vol. 3636, 1–11, Springer, 2005.
13. Delisle, P., Gravel, M., Krajecki, M., Gagné, C. y Price, W.: A Shared Memory Parallel Implementation of Ant Colony Optimization. Proc. 6th MIC, 257–264, 2005
14. Doerner, K., Hartl, R., Benkner, S. y Lucká, M.: Parallel Cooperative Savings Based Ant Colony Optimization - Multiple Search and Decomposition Approaches. Parallel Processing Letters, vol. 16, no. 3, 351–370, 2006.
15. Doerner, K., Hartl, R., Kiechle, G., Lucká, M. y Reimann, M.: Parallel AS for the Capacitated Vehicle Routing Problem. Proc. EvoCOP, LNCS v. 3004, 72–83, 2004.
16. Doerner, K., Hartl, R.F. y Lucká, M.: A Parallel Version of the D-Ant Algorithm for the Vehicle Routing Problem. Proceedings of ParNum05, 109–118, 2005.
17. Dorigo, M. y Stützle, T.: Ant Colony Optimization. MIT Press, 2004.
18. Ellabib, I., Calamai, P.H. y Basir, O.A.: Exchange Strategies for Multiple Ant Colony System. Information Sciences, vol. 177, no. 5, 1248–1264, 2007.
19. Iimura, I., Hamaguchi, K., Ito, T. y Nakayama, S.: A Study of Distributed Parallel Processing for Queen Ant Strategy in ACO. Proc. of the 6th PDCAT, 553–557, 2005
20. Janson, S., Merkle, D. y Middendorf, M.: Parallel Ant Colony Algorithms. En E. Alba (editor): Parallel Metaheuristics, 171–201. Wiley, 2005.
21. Luque, G., Alba, E. y Dorronsoro, B.: Parallel Genetic Algorithms En E. Alba (editor): Parallel Metaheuristics, 105–125. Wiley, 2005.
22. Manfrin, M., Birattari, M., Stützle, T. y Dorigo, M.: Parallel Ant Colony Optimization for the TSP. Proc. of the 5th ANTS, LNCS, vol. 4150, 224–234, 2006
23. Michel, R. y Middendorf, M.: An Island Model Based AS with Lookahead for the Shortest Supersequence Problem. Proc. of 5th PPSN, LNCS, v. 1498, 692–701, 1998.
24. Middendorf, M., Reischle, F. y Schmeck, H.: Multi Colony Ant Algorithms. Journal of Heuristics, vol. 8, issue 3, 305–320, 2002.
25. Mocholí, J., Martínez, J. y Canós, J.: A Grid Ant Colony Algorithm for the Orienteering Problem. Proceedings of the 2005 IEEE CEC, 942–949, 2005.
26. Piriyakumar, D. y Levi, P.: A New Approach to Exploiting Parallelism in Ant Colony Optimization. Proceedings of MHS 2002, 237–243, 2002.
27. Rahoual, M., Hadji, R. y Bachelet, V.: Parallel Ant System for the Set Covering Problem. Proceedings of the 3rd ANTS, LNCS, vol. 2463, 262–267, Springer, 2002.
28. Randall, M. y Lewis, A.: A Parallel Implementation of Ant Colony Optimization. Journal of Parallel and Distributed Computing, vol. 62, n. 9, 1421–1432, 2002.
29. Stützle, T.: Parallelization Strategies for Ant Colony Optimization. Proceedings of the 5th PPSN, LNCS, vol. 1498, 722–731, Springer, 1998.
30. Talbi, E-G., Roux, O., Fonlupt, C. y Robillard, D.: Parallel Ant Colonies for the Quadratic Assignement Problem. FGCS, vol. 17, no. 4, 441–449, 2001.
31. Tsutsui, S.: Parallel Ant Colony Optimization for the Quadratic Assignment Problem with Symmetric Multi Processing. MEDAL Rept. 2008006, U. of Missouri, 2008.
32. Tsutsui, S. y Liu, L.: Cunning Ant System for Quadratic Assignment Problem with Local Search and Parallelization. MEDAL Rept. 2007006, Univ. of Missouri, 2007.
33. Yang, Z., Yu, B. y Cheng, C.: A Parallel Ant Colony Algorithm for Bus Network Optimization. Comp.-Aided Civil & Infrastructure Eng., vol. 22, no. 1, 44–55, 2007.