Estrategias cooperativas paralelas en la solución de problemas de optimización

Carlos Cruz Corona
Grupo de Modelos de Decisión y Optimización
Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Granada, España
carloscruz@decsai.ugr.es

J. Marcos Moreno Vega
Grupo de Computación Inteligente
Departamento de Estadística, I.O. y Computación
Escuela Técnica Superior de Ingeniería Informática
Universidad de La Laguna, España
jmmoreno@ull.es

Resumen

Los métodos cooperativos paralelos parecen ser una buena herramienta para resolver problemas de optimización combinatoria de gran tamaño y alta complejidad, no solo porque mejoran el factor de aceleración en la búsqueda de las soluciones, si no también porque alcanzan soluciones de muy alta calidad y comportamientos robustos. En este trabajo se tratan los fundamentos de la cooperación partiendo de algunos ejemplos tomados de la naturaleza y su uso en el contexto de las heurísticas. Se hace un estudio de implementaciones cooperativas paralelas de diferentes metaheuristicas, haciendo énfasis en métodos cooperativos centralizados y descentralizados.

Palabras claves: Metaheurísticas, estrategias cooperativas, optimización, paralelismo.

1. Introducción

El ser humano constantemente se enfrenta a situaciones que lo obligan a tomar decisiones ante una infinidad de problemas en los que no basta el sentido común y entonces hay que acudir a la ayuda de la ciencia. Una categoría importante y compleja de estos problemas son aquellos en que se desea maximizar o minimizar una cantidad específica, que puede ser llamada objetivo, y que depende de un número finito de variables de entrada. Estas variables pueden ser independientes unas de otras o pueden estar relacionadas por una o más restricciones.

Para modelar estos problemas, lo primero es identificar las posibles decisiones que pueden tomarse y esto puede hacerse a través de las variables del problema concreto que son generalmente de carácter cuantitativo, buscándose los valores de las mismas que optimizan el objetivo. Posteriormente se define qué decisiones resultan admisibles, lo que conduce a un conjunto de restricciones teniendo presente la naturaleza del problema. Se calcula el coste/beneficio asociado a cada decisión

admisible lo que supone determinar una función objetivo que asigna un valor de coste/beneficio a cada conjunto posible de valores para las variables que conforman una decisión. El conjunto de todos estos elementos define un problema de optimización.

Así, un problema de *Optimización* puede formularse como:

$$optimizar_{X \in S} f(X)$$

donde $f : S \rightarrow \Re$ es una función que a cada $X \in S$ asocia un número real, y S es un conjunto finito o infinito de puntos. Al conjunto S se le conoce como *espacio solución* o *región factible* y a la función f por *función de costo* o *función objetivo*. Por optimizar se entiende minimizar o maximizar la función objetivo f sobre el espacio solución. Sin pérdida de generalidad, puede suponerse que el problema es de minimización, ya que maximizar una función f, equivale a minimizar la función cambiada de signo, $-f$.

Existen multitud de problemas reales que se pueden formular como un problema de minimización. Algunos de estos problemas son conocidos desde hace muchos años, y para ellos se han desarrollado procedimientos de solución específicos. Estos procedimientos son exactos o aproximados. Dado que muchos de los problemas que surgen son NP-completos, la teoría de la NP-completitud [35] indica que éstos no pueden ser resueltos de forma óptima con una cantidad razonable de recursos. Por ello, se ha dedicado un gran esfuerzo investigador a la obtención de heurísticas para cada problema particular.

Un inconveniente de la gran mayoría de los métodos heurísticos aparecidos en la literatura es su dependencia de la estructura del problema considerado, y su falta de habilidad para adaptarse a nuevas situaciones o modificaciones del problema de partida. Así, usan propiedades de la región factible y/o de la función objetivo o información a priori sobre las soluciones óptimas que hacen que los procedimientos sean válidos sólo bajo esas condiciones. Sin embargo, es frecuente encontrar problemas en los que la función objetivo no cumple las condiciones para las que han sido desarrollados los procedimientos conocidos, o para los que no se posee ningún tipo de información sobre las soluciones óptimas. Además, el diseño de una buena heurística para un problema exige muchas veces un conocimiento matemático del problema que no poseen las personas que las usarían en su toma de decisiones. En estos casos es necesario disponer de procedimientos que sean *robustos* (poco sensibles a pequeñas alteraciones del modelo), *generales* (ampliamente utilizables), *sencillos* (fáciles de implementar) y *efectivos*.

Las Metaheurísticas son procedimientos que cumplen con estas características y, además en muchos casos, son efectivos. Constan de un número conocido de elementos que son fácilmente adaptables a cualquier problema. Además, pueden incorporar conocimiento específico para la resolución del problema en forma de heurísticas dependientes del mismo. La efectividad de las metaheurísticas ha sido probada en la resolución de numerosos problemas de indudable interés práctico.

Aunque se ha intuido desde hace tiempo que la cooperación entre metaheurísticas podría mejorar el desempeño que se obtiene con cada una de ellas por separado, no ha sido hasta hace poco cuando se ha empezado a estudiar esta opción. La idea inicial de compartir información entre programas de búsqueda de soluciones se hizo para acelerar la exploración del proceso de búsqueda. Sin embargo varios estudios han demostrado que la cooperación también modifica profundamente los patrones de búsqueda de los métodos y se obtienen rendimientos muy altos [19, 17]. Aún así hay pocos estudios realizados que permitan desarrollar un modelo comprensivo de la cooperación entre procesos de búsqueda.

Por ello, en el presente trabajo se recopilan varias de las estrategias de cooperación para un amplio grupo de metaheurísticas, así como sus métodos de paralelización, y se enfatiza en las implementaciones paralelas cooperativas centralizadas y descentralizadas, de manera que sirva como punto de inicio a la profundización de estos temas.

El trabajo está organizado como sigue: la Sección 2 trata sobre el concepto de cooperación y cómo se manifiesta en la naturaleza, la Sección 3 describe la cooperación en el contexto de las metaheurísticas, la Sección 4 recoge las estrategias de paralelización de las metaheurísticas, en la Sección 5 se presenta un estado del arte de diferentes metaheurísticas y sus esquemas paralelos cooperativos centralizados y descentralizados. Las conclusiones se presentan en la Sección 6.

2. Cooperación

Cooperar es, según la Real Academia Española de la Lengua, *obrar juntamente con otro u otros para un mismo fin.*

Esto se manifiesta a cualquier nivel, desde sistemas abstractos a los organismos biológicos mas simples. Al existir variedad ya aparecen dos actitudes en la naturaleza de las cosas: la cooperación y la lucha, sin olvidar la dinámica intrínseca a todo esto, de forma que en determinados momentos los que hoy luchan entre sí pueden cooperar mañana, y viceversa. Por ejemplo, se dice que la célula primitiva y la mitocondria, que hoy cooperan, hace millones de años fueron enemigos irreconciliables. Según la teoría endosimbiótica [52, 53], el mitocondrio, una célula procariota capaz de obtener energía a partir del oxígeno, se fusionó en un momento de la evolución con las células eucariotas, suministrándoles la mayor parte de la energía necesaria para la actividad celular, aprovechando el aumento de la concentración de oxígeno en la atmósfera terrestre. Es decir, sus movimientos devinieron en cooperativos y provocaron un cambio fundamental en la naturaleza.

Nos referimos en este trabajo a la cooperación entre individuos no relacionados por parentesco, o sea que no puede ser entendida directamente mediante una explicación estrictamente darwinista, y donde cada sistema consiste en muchas unidades diferenciadas que actúan de forma simultánea, sin ningún proceso regulador global responsable de sus comportamientos individuales. Las acciones de cada unidad dependen de los estados y de las acciones de un número limitado de otras unidades con las que interaccionan, y la evolución global del sistema queda determinada por la coordinación entre las unidades según relaciones estructurales.

La cooperación en entornos naturales desde este punto de vista se puede dividir en tres grandes métodos [30]:

- *Mutualismo por producto:* Ocurre cuando dos organismos interactúan produciendo un beneficio mutuo. Un ejemplo de esto ocurre en la polinización de las plantas por vectores biológicos. Se crea una asociación en la que la planta ofrece algún tipo de recompensa al vector y éste a su vez transporta el polen. Cuanto más atractiva resulte una planta a los vectores, aumenta el índice de visitas y por lo tanto la producción de semillas.
- *Reciprocidad:* Los individuos con más capacidad hacen "favores" con la esperanza de ser recompensados más adelante en una situación desfavorable. Un sistema de "memoria" o reconocimiento de los favorecedores pasados es necesario para este método de cooperación. Los individuos con un pasado "altruista", respecto al donante, son ayudados con mas frecuencia que aquellos que fueron "egoístas". Un ejemplo de este tipo ocurre en los vampiros (Desmodus rotundus) [87], donde los individuos que fueron exitosos obteniendo alimento durante la noche entregan (regurgitan) sangre a individuos que no lograron obtener alimento.
- *Selección de grupos:* En este método la selección no opera sobre la unidad de estudio estándar del modelo darwinista (el individuo), sino sobre el grupo como unidad evolutiva. En este modelo son necesarios al menos dos grupos de individuos y las habilidades cognitivas de reconocimiento de estos para que ocurra la cooperación. Un ejemplo de ello es el comportamiento de la urraca yucateca [1]. Estas aves muestran un importante desarrollo de sistemas sociales cooperativos, al grado de que la unidad social es la bandada. Construyen y cuidan su nido comunitariamente. La hembra que incuba sus huevos es alimentada por otro pájaro, en intervalos irregulares de tiempo. Incluso en algunas ocasiones la hembra llega a pedir a gritos, aleteando y levantando la cola, que le lleven comida, lo cual hacen solícitamente los demás miembros del grupo. Los pájaros que no incubaron visitan frecuentemente el nido, aunque no lleven comida. Incluso limpian de excrementos el interior del nido, pero no los tiran en lugares cercanos, se los llevan lejos o bien se los comen rápidamente. Quizá esto se haga para evitar la aparición de parásitos en el nido, puesto que también se les ha visto comerse a unas pequeñas larvas que crecen en el mismo.

Estos modelos cooperativos son adecuados para aplicarlos a contextos donde la población sea un conjunto de agentes que operan sobre el espacio de soluciones y donde las mejoras en la búsqueda no se traducen necesariamente en mecanismos de representación de soluciones que pasan de una generación a la siguiente. Así, a continuación se

presenta la cooperación desde el punto de vista de las heurísticas.

3. Heurísticas paralelas cooperativas

En el contexto de las Heurísticas, el objetivo que se persigue con la cooperación es mejorar la calidad de la solución dada a un problema o disminuir la cantidad de recursos necesarios para obtener dicha solución. Dicho de otro modo, cooperando se pretende, fundamentalmente, aumentar la eficiencia o la eficacia del algoritmo de búsqueda.

La cooperación puede producirse entre varias heurísticas, por ejemplo cuando un método de Búsqueda por Entornos Variables intercambia información con un método de Búsqueda Tabú, o entre elementos de una misma heurística, por ejemplo cuando en un método Multiarranque, la información que suministran los óptimos locales se emplea para dirigir la búsqueda hacia subregiones prometedoras de la región factible.

Por su naturaleza la cooperación entre varias heurísticas es mas explícita que entre los elementos de una misma, por ello en esta última resulta complejo definir con claridad los patrones de la cooperación.

Los esquemas cooperativos pueden clasificarse atendiendo a diversas características. Una de ellas es teniendo en cuenta la forma en que se produce la cooperación entre los módulos del sistema, en la cual aparecen dos tipos: centralizada y descentralizada. Considerando que un esquema cooperativo está formado por diferentes módulos que cumplen una función determinada dentro del sistema, tendríamos un esquema centralizado, cuando la cooperación entre los *módulos de búsqueda* se realiza a través de un *módulo coordinador*; y en un esquema descentralizado, la cooperación se realiza directamente entre los *módulos de búsqueda*.

Estos esquemas cooperativos generalmente, y últimamente con mas frecuencia, se implementan en un entorno paralelo, sin embargo, el uso del paralelismo no es imprescindible en el diseño, implementación y análisis de estrategias cooperativas. Este trabajo se enfoca sólo a los esquemas cooperativos paralelos.

El paralelismo surge como una necesidad de lograr un mayor poder computacional en una sociedad tecnológica cada vez mas abrumada por la ingente cantidad de operaciones necesarias para el análisis de la realidad que la envuelve, y en la que no se busca paralelizar como meta en sí, el objetivo es obtener mayores rendimientos que permitan resolver problemas cada vez más grandes en tiempos de computación aceptables.

Referido a la paralelización de las metaheurísticas se distinguen tres tipos de métodos atendiendo a la fuente de paralelismo [19]:

- *Paralelismo del Tipo 1 o de Bajo Nivel*: La fuente de paralelismo se encuentra normalmente dentro de una iteración del método de búsqueda. El paralelismo se obtiene al ejecutar o evaluar concurrentemente los movimientos presentes en una iteración del método. El paralelismo tipo 1 se emplea para reducir el tiempo de cómputo.
- *Paralelismo del Tipo 2 o Descomposición del Dominio*: Se obtiene al descomponer la región factible en subregiones disjuntas (descomponiendo, por ejemplo, el conjunto de variables de decisión en subconjuntos disjuntos). A continuación se ejecuta una heurística particular en cada subregión, considerando fijas el resto de subregiones. Esta estrategia se implementa generalmente en un modelo maestro-esclavo, en el que el proceso maestro divide las variables de decisión. Durante la búsqueda, el maestro puede modificar esta división. Estas modificaciones, que se hacen a intervalos, se pueden fijar antes o durante la ejecución, o ser ajustadas en el momento de reiniciar el método. Los esclavos exploran de forma concurrente e independiente sólo su subregión. Las otras variables se mantienen fijas y no se ven afectadas por los movimientos realizados, aunque se tenga acceso a todo el conjunto de variables. Cuando los esclavos tienen acceso a todo el vecindario, el maestro debe realizar operaciones complejas de combinación de las soluciones parciales obtenidas de cada subconjunto para formar una solución completa del problema.
- *Paralelismo del Tipo 3 o Búsqueda Múltiple*: Se lleva a cabo ejecutando concurrentemente varias búsquedas (o hilos) en el espacio de soluciones. Cada hilo concurrente puede o no ejecutar el mismo método, puede empezar por las mismas o diferentes soluciones iniciales, etc. Los hilos pueden comunicarse durante la búsqueda o sólo al fi-

nal para identificar la mejor de todas las soluciones. La primera estrategia se denomina método cooperativo de búsqueda múltiple y la segunda es conocida como método de búsqueda independiente. La comunicación puede realizarse de manera sincrónica o asincrónica, y puede ser manejada por eventos o ejecutada en momentos decididos dinámicamente o predeterminados de antemano. Esta estrategia se usa frecuentemente para llevar a cabo una mayor exploración del espacio de búsqueda. Varios estudios [18, 17] han mostrado que las técnicas multihilos proporcionan mejores soluciones que su correspondiente contraparte secuencial, incluso cuando el tiempo disponible de ejecución para cada hilo es menor que el de la computación secuencial. Dichos estudios han mostrado también que la combinación de distintos hilos, implementando cada una de ellos diferentes estrategias, incrementa la robustez de la búsqueda global en relación con las variaciones de las características de las instancias del problema.

En [84, 85, 21] se clasifican las estrategias de paralelización en dos categorías, métodos de un único camino y métodos de múltiples caminos, atendiendo al número de trayectorias que simultáneamente se investigan en la región factible. Como se verá a continuación, esta clasificación coincide, en gran medida, con la propuesta anterior de Crainic y Toulouse [19].

La paralelización de un único camino se emplea, principalmente, para aumentar la velocidad del recorrido secuencial al explorar eficientemente el espacio de búsqueda. La tarea cuya ejecución se paraleliza puede ser la evaluación de la función de coste, la exploración del entorno de la solución actual o cualquier otra que requiera un uso intensivo de recursos computacionales. La paralelización de la función de coste no altera la trayectoria seguida por la implementación secuencial. Para la exploración eficiente del entorno de la solución actual, se descompone éste en subconjuntos disjuntos que son distribuidos entre varios procesadores. De esta forma, puede explorarse una mayor porción del entorno de una solución, en aquellos problemas en los que éstos constan de un gran número de soluciones vecinas, o analizar completamente todo el entorno en menos tiempo. Es decir, se obtiene un aumento de la eficiencia o un mayor nivel de exploración que conducirá, posiblemente, a mejores soluciones.

La paralelización de múltiples caminos se caracteriza por la investigación en paralelo de múltiples trayectorias, cada una en un procesador diferente. En cada hebra o hilo de búsqueda se puede ejecutar el mismo algoritmo, el mismo algoritmo con diferentes parámetros, o diferentes algoritmos. Además, los hilos pueden cooperar o ejecutarse de forma independiente. En las estrategias independientes, los hilos pueden partir de soluciones o poblaciones iniciales diferentes, o descomponer la región factible en subregiones disjuntas que son asignadas a cada hilo. Se obtienen así, respectivamente, trayectorias que se intersectan y trayectorias que no.

En la literatura pueden encontrarse otras clasificaciones de las metaheurísticas paralelas, referidas, en muchos casos, a un tipo específico de metaheurísticas. Así, Greening [40] divide las estrategias paralelas del Templado Simulado de acuerdo al grado de seguridad en la evaluación de la función de costo asociada con un movimiento. Los algoritmos que brindan una evaluación libre de error las identifica como sincrónicos, y como asincrónicos el resto. A su vez, los sincrónicos se dividen en aquellos que mantienen las propiedades de convergencia del método secuencial, y aquellos que generan una patrón de búsqueda que difiere del patrón de búsqueda del algoritmo secuencial.

Cantú-Paz [9] propone una clasificación para los Algoritmos Genéticos paralelos. La primera categoría llamada paralelización global es idéntica a la paralelización del tipo 1. Las otras dos categorías clasifican los Algoritmos Genéticos de acuerdo al tamaño de su población, así están los de grano grueso y los de grano fino. Hay también una clase de Algoritmos Genéticos paralelos híbridos, que enmarca, por ejemplo, a aquellos de paralelización global que tienen subpoblaciones de algoritmos de grano fino.

Habría que señalar que el uso de las medidas clásicas de desempeño para estas metaheurísticas en paralelo es en cierto modo problemático [18]. El desempeño de los algoritmos paralelos está fuertemente vinculado al ambiente de programación en el que nos encontremos, así como de factores cuya homogeneidad en el campo secuencial se da por entendida: diseño de la red de comunicaciones, aspectos de sincronización, latencia de los mensajes, etc. Además, en el caso de haber cooperación los hilos interactúan asincrónicamente, de forma que las distintas ejecuciones pueden producir distintas salidas para la misma entrada. De

esta manera, el tema de comparación de estas estrategias en paralelo con sus correspondientes secuenciales es un campo abierto y activo.

Para calificar la medida clásica del factor de aceleración se está usando la noción de calidad de la solución [18], o sea comparar cuál método encuentra la mejor solución, aunque esto tampoco permite una comparativa clara debido a lo complejo de lograr recursos computacionales similares.

Toulouse y Crainic [20] defienden que el comportamiento global de un esquema paralelo cooperativo centralizado depende del valor que toman los siguientes cuatro parámetros: información recopilada e intercambiada, tipo de algoritmos que cooperan, frecuencia con que se produce el intercambio de información y número de algoritmos que cooperan. Estos parámetros pueden usarse también para caracterizar un esquema cooperativo descentralizado. Sin embargo, para ese caso, debemos especificar cómo se comunican entre sí los *hilos de búsqueda.*

1. *Información que se intercambia.* En general, la información que se intercambia es la mejor solución encontrada o los atributos que forman parte de esa solución. No obstante, existe otro tipo de características que pueden ser igualmente útiles para mejorar el comportamiento del sistema. Destacamos el número y valor de los óptimos locales encontrados o la distancia entre ellos. La información debe ser significativa, en el sentido de que debe ser útil para el proceso de decisión de la búsqueda en cada uno de ellos. Debe ser información que brinde indicaciones correctas acerca del estado actual de la búsqueda global. Estudios previos [16, 20] demuestran que estos métodos cooperativos con acceso no restringido a la información compartida pueden experimentar serios problemas de convergencia. También resaltan el hecho de que el simple intercambio de información, por ejemplo el mejor valor, puede dirigir la búsqueda a regiones no esperadas sin tener en cuenta una lógica en el proceso de optimización.

2. *Tipo de algoritmos que cooperan.* Pueden distinguirse esquemas homogéneos y heterogéneos. En los primeros, los algoritmos que cooperan se obtienen ejecutando una misma heurística con diferentes valores de los parámetros, desde soluciones de inicio diferentes o restringiendo la búsqueda a subregiones disjuntas del espacio de soluciones. En los esquemas heterogéneos, la cooperación se produce entre heurísticas difererentes. Algunos trabajos [51][16][57] respaldan la idea de que el uso de esquemas heterogéneos suministra métodos eficientes para resolver diversos problemas.

3. *Frecuencia con que se produce el intercambio de información.* Los esquemas pueden ser síncronos, asíncronos, con intercambio frecuente, etc.

4. *Número de algoritmos que cooperan.*

5. *Tipo de cooperación.* Se distingue entre cooperación centralizada y cooperación descentralizada. Para esta última, los *hilos de búsqueda* pueden cooperar actualizando el valor de aquellas variables que se emplean para dirigir la búsqueda.

4. Cooperación centralizada y descentralizada

En la literatura pueden encontrarse varias propuestas éxitosas de técnicas heurísticas cooperativas aplicadas a ciertos problemas. Una referencia a varios de estos trabajos aparecen en [21, 75].

- *Algoritmos Genéticos, AG* [64, 4, 25]: Son una familia de modelos computacionales inspirados en los mecanismos de herencia y evolución natural. Mediante una imitación del mecanismo genético de los organismos biológicos, los algoritmos genéticos exploran el espacio de soluciones asociado a un determinado problema. Se establece una codificación apropiada de las soluciones del espacio de búsqueda y una forma de evaluar la función objetivo para cada una de estas codificaciones. Las soluciones se identifican con individuos que pueden formar parte de la población de búsqueda. La codificación de una solución se interpreta como el cromosoma del individuo compuesto de un cierto número de genes a los que les corresponden ciertos alelos. El gen es la característica del problema; alelo, el valor de la característica; genotipo, la estructura y fenotipo, la estructura sometida al problema. Cada cromosoma es una estructura de datos que representa una de las posibles soluciones del espacio de búsqueda del problema. Los cromosomas

son sometidos a un proceso de evolución que envuelve evaluación, selección, recombinación sexual o cruce y mutación. Después de varios ciclos de evolución la población deberá contener individuos más aptos.

Se consideran dos operaciones básicas: la mutación y el cruce. La mutación de un individuo consiste en modificar uno o varios genes cambiando al azar el alelo correspondiente. El cruce de dos individuos, llamados padres, produce un individuo hijo tomando un numero k (elegido al azar) de genes de uno de los padres y los $t - k$ del otro. La población evoluciona de acuerdo a las estrategias de selección de individuos, tanto para las operaciones como para la supervivencia.

Se puede entender el mecanismo de selección si se plantea cada iteración del Algoritmo Genético como un proceso en dos etapas. En la primera, se aplica algún mecanismo de selección sobre la "población actual" para crear una "población intermedia"; y en la segunda etapa, a partir de esta población, se aplica cruce y mutación para generar una "nueva población".

Ya en 1976 Bethke [6] describe implementaciones paralelas centralizadas para los AG donde hace un análisis detallado de la eficiencia del uso de la capacidad de procesamiento. Grefenstette en 1981 [41] propone varios prototipos y en algunos de ellos un proceso maestro hace la selección y aplica cruce y mutación a los individuos, y luego se envían a los esclavos para ser evaluados. Abramson [2] implementó un método sobre una máquina de memoria distribuida, en el que un maestro almacena la población para facilitar la aplicación de los operadores genéticos, envía los individuos a los esclavos para su evaluación, recolecta los resultados y aplica los operadores para producir la próxima generación. Lis [59] aplica un modelo en el que un proceso maestro controla toda la población y envía el mismo conjunto de los mejores individuos a los procesos esclavos, cada uno de los cuales tiene una probabilidad de mutación diferente.

También entre los prototipos propuestos por Grefenstette [41], puede encontrarse un método descentralizado en el que los mejores individuos son enviados en cada generación al resto de procesadores. Tanese en 1987 [82] propuso, entre los primeros trabajos de grano grueso, una topología de hipercubo 4D para comunicar individuos de una subpoblación a otra. Desde sus inicios la mayoría de las implementaciones de grano grueso usan las topologías disponibles de sus computadores, como hipercubo [11] y anillos [39], para comunicar individuos de una subpoblación a otra. Los métodos de grano fino dividen la población en un gran número de pequeños subconjuntos que son asignados a los procesadores disponibles. Cada procesador es conectado a otros en su vecindario. Por ello, es común que sus implementaciones se hagan en computadoras paralelas masivas con determinadas topologías de redes [79, 60, 3]. En [66] se presentan trabajos que resumen el paralelismo de grano fino, así como su hibridación con otras estrategias.

Otros trabajos interesantes son los relacionados al uso de algoritmos evolutivos en sistemas coevolutivos, que aparecen a partir de 1991 [5]; aunque son mas frecuentes las aplicaciones competitivas que cooperativas. De estas últimas están los trabajos empíricos de Potter y De Jong [71] indicando que hay un posible enlace entre la interactividad variable y la selección de la colaboración. Wiegand et al. [86] realizan un análisis empírico de varios tipos de mecanismos de colaboración y dan algunos consejos acerca de cómo seleccionar el mecanismo apropiado para un problema particular.

- *Búsqueda por Entornos Variables, (Variable Neighborhood Search, VNS)* [44, 45, 46, 43, 42], es una metaheurística que está basada en un principio simple: cambiar sistemáticamente de estructura de entornos dentro de la búsqueda para escapar de los mínimos locales.

  El VNS básico obtiene una solución del entorno de la solución actual, ejecuta una búsqueda monótona local desde ella hasta alcanzar un óptimo local, que reemplaza a la solución actual si ha ocurrido una mejora y modifica la estructura de entorno en caso contrario.

  Se le han hecho extensiones que constituyen mejoras prácticas de la VNS que han permitido resolver con éxito problemas muy grandes: la Búsqueda Descendente por Entornos Variables *(Variable Neighbourhood Descent, VND)*[46], aplica una búsqueda monótona, o sea de mejora por entornos, cambiando de forma sistemática la estructura de entornos cada vez que se alcanza un

mínimo local. La búsqueda local de la VNS puede ser sustituida por la VND y forma lo que se conoce como Búsqueda de Entorno Variable General *(General Variable Neighbourhood Search, GVNS)* que ha dado lugar a muchas aplicaciones exitosas [46]. También están la Búsqueda de Entorno Variable Sesgada *(Skewed Variable Neighbourhood Search, SVNS)* y la Búsqueda de Entorno Variable Paralela *(Parallel Variable Neighborhood Search, PVNS)*, así como hibridación con otras metaheurísticas como Tabú, Multi-arranque y GRASP.

Algunos casos de cooperación centralizada con VNS los presentan García-López et al. [33] resolviendo la p-mediana y donde aplican mecanismos de cooperación sincrónico en el que un proceso maestro ejecuta la VNS secuencialmente y la solución actual se envía a los esclavos que la usan para obtener la solución inicial desde la cual se inicia la búsqueda local. Las soluciones obtenidas son pasadas al maestro que selecciona la mejor y continua el algoritmo. Crainic et al. [17] proponen un método cooperativo multi-búsqueda basado en un mecanismo de memoria central resolviendo instancias muy grandes de la p-mediana.

- *Algoritmos Meméticos, AM*[12, 13, 65] Metaheurísticas derivadas de los algoritmos genéticos y desarrolladas por Moscato en 1989, y que surgen de combinar los algoritmos genéticos con búsquedas locales. La denominación "memético" surge del término inglés "meme", acuñado por R. Dawkins (1976) como el análogo del gen en el contexto de la evolución cultural.

  Un AM mantiene en todo momento una población de diversas soluciones al problema considerado, que se denominan agentes. Esta denominación es una extensión del término individuo. Estos agentes se interrelacionan entre sí en un marco de competición y de cooperación, de manera muy semejante a lo que ocurre en la Naturaleza entre los individuos de una misma especie a través de la selección y el reemplazo.

  Digalakis [27] propone un modelo cooperativo paralelo centralizado en el cual la población se distribuye entre los procesadores del sistema, de forma que cada uno tiene su propia subpoblación y una tabla local de localización que indica donde pueden encontrarse los cromosomas no locales. Un procesador maestro está a cargo de la parte secuencial del algoritmo y mantiene una tabla de operación que refleja los cromosomas seleccionados para sobrevivir, ser mutados y/o tomar parte en el cruce. Esta tabla es enviada a cada procesador esclavo al inicio de cada generación. Este modelo se prueba ante un conjunto de 10 diferentes problemas.

- *Colonias de Hormigas (Ant Colony Optimization, ACO)*[28, 24, 8]. Esta metaheurística emplea estrategias inspiradas en el comportamiento de las colonias de hormigas para descubrir fuentes de alimentación, al establecer el camino más corto entre éstas y el hormiguero y transmitir esta información al resto de sus compañeras.

  Cada hormiga construye una solución, o un componente de esta, comenzando en un estado inicial seleccionado de acuerdo a criterios dependientes del problema. Mientras construye su propia solución colecciona información sobre las características del problema y sobre su actuación y usa esta información para modificar la representación del problema.

  El algoritmo secuencial de este método tiene un alto grado de paralelismo natural, ya que puede verse como un conjunto de agentes cooperativos y un conjunto de reglas que determinan la generación, actualización y uso de información local y global para encontrar las soluciones. Bullnheimer et al. [7] desarrollan dos estrategias centralizadas paralelas para resolver el TSP, una sincrónica y otra parcialmente asincrónica. En estas un proceso maestro crea un conjunto de esclavos, uno para cada hormiga y le envía la información inicial del problema. Cada esclavo prepara el camino y calcula la longitud de la travesía para su hormiga. Estos resultados son enviados al maestro que actualiza los niveles del rastro y chequea el mejor camino encontrado. Los niveles de rastro actualizados son enviados a cada hormiga. En caso de usar búsqueda local para mejorar las soluciones [29] se usa un maestro para actualizar las estructuras de datos, construir las soluciones iniciales para la búsqueda local y enviar las soluciones a los esclavos cuando estas son mejoradas. El maestro recolecta estos óptimos locales y al alcanzar un número suficiente de éstos, actualiza la matriz de rastro antes de construir otras soluciones. Thomas

Stützle [81] propone un método en el cual un procesador central mantiene las estructuras de datos para la matriz de rastro y la actualiza. Otros procesadores usan esta matriz para construir soluciones que son enviadas a su vez a otros procesadores en otro nivel para que realicen la búsqueda local de estas soluciones. Las soluciones que son mejoradas son enviadas al procesador central.

- La Búsqueda Tabú (*Tabu Search, TS*) [38] [36], es una de las estrategias que tratan de utilizar la memoria del proceso de búsqueda para mejorar su rendimiento evitando que la búsqueda se concentre en una misma zona del espacio. Hace menos énfasis a la aleatoriedad, y generalmente se usa en un modo altamente restringido, con el supuesto de que la búsqueda inteligente debería estar basada en formas más sistemáticas de dirección. Por consiguiente, muchas de sus implementaciones son en gran parte deterministas [38].

  La Búsqueda Tabú se caracteriza por dos aspectos principales: Un mecanismo de generación de vecinos modificado que evita la exploración de zonas del espacio de búsqueda que ya han sido visitadas: generación de entornos tabú restringidos, y mecanismos para mejorar la capacidad del algoritmo para la exploración-explotación del espacio de búsqueda.

  Se han hecho varias aplicaciones centralizadas donde un proceso maestro ejecuta un método tabú secuencial y despacha en cada iteración los posibles movimientos en el vecindario de la solución actual a ser evaluados en paralelo por los esclavos. Los esclavos pueden decidir en evaluar los movimientos del conjunto recibido desde el maestro o usar movimientos mas allá del conjunto recibido. El maestro recibe y procesa la información resultante de las operaciones en los esclavos y selecciona e implementa el proximo movimiento. También recoge toda la información generada durante la exploración, actualiza las memorias y decide si activa estrategias diferentes de búsqueda o detener el proceso. Malek et al. [62] presentaron una estrategia cooperativa sincrónica en la que un proceso maestro controla la cooperación y los esclavos ejecutan métodos tabú secuenciales con diferentes parámetros. Rego et al. [73] implementan un método paralelo multihilo independiente donde cada esclavo ejecuta una búsqueda tabú secuencial completa y el maestro recoge las soluciones encontradas por los hilos, selecciona la mejor y reinicia los hilos para una nueva búsqueda. En [15] proponen una estrategia con memoria central para la comunicación inter-hilos aplicada al MLAP (multicommodity location-allocation problems) con muy buenos resultados que luego también aplican Crainic et al. [14] en una estrategia asincrónica multihilo para problemas de diseño de redes capacitadas.

  Un ejemplo de implementación descentralizada lo presentan De Falco et al. [23] con un modelo donde cada proceso ejecuta una búsqueda local desde su mejor solución. Entonces se sincronizan los procesos y las mejores soluciones se intercambian entre aquellos procesos que se ejecutan en procesadores vecinos. Las mejores soluciones se reemplazan si las soluciones importadas son de mayor calidad. En [76] se usa una variante basada en memoria adaptable que aplican al problema de ruta de vehículos, donde un conjunto de soluciones parciales se distribuye entre todos los procesadores; y en la que los elementos de las mejores soluciones encontradas por cada hilo de búsqueda son emitidos a todos para asegurar que cada búsqueda tiene aún acceso a toda la información cuando se construyen nuevas soluciones.

- *Templado Simulado (Simulated Annealing, SA)*, [22] [47] [63] es conocida como la primera de las metaheurísticas y uno de los primeros algoritmos con una estrategia explícita para escapar de óptimos locales. Un modo de evitar que la búsqueda local finalice en óptimos locales, hecho que suele ocurrir con los algoritmos tradicionales de búsqueda local, es permitir que algunos movimientos se produzcan hacia soluciones peores. Pero si la búsqueda está avanzando realmente hacia una buena solución, estos movimientos de escape de óptimos locales deben realizarse de un modo controlado. Esto aquí se realiza controlando la frecuencia de los movimientos de escape mediante una función que hará disminuir la probabilidad de estos movimientos hacia soluciones peores conforme avanza la búsqueda, y por tanto se está más cerca, previsiblemente, del óptimo local.

El fundamento de este control se basa en el trabajo de Metropolis en el campo de la termodinámica estadística realizado en 1953, quien modeló el proceso de enfriamiento, simulando los cambios energéticos en un sistema de partículas conforme decrece la temperatura hasta que converge a un estado estable (congelado).

Felten et al. [32] la aplican al TSP (Travelling Salesman Problem) para 64 ciudades donde un proceso central genera aleatoriamente un "tour" inicial y lo divide en p subconjuntos de ciudades adyacentes que son asignadas a p procesos esclavos. Janaki et al. [72] aplica varias versiones paralelas de este método al TSP y al JSS (Job Shop Scheduling). Usa un método de agrupamiento en el cual un nodo central se encarga de generar diferentes estados iniciales que envía a los nodos esclavos y espera por los resultados de estos. Los esclavos hacen varias iteraciones del algoritmo del Templado Simulado e intercambian soluciones parciales entre ellos y con la mejor solución obtenida cada trabajador ejecuta de nuevo el algoritmo esta vez de forma independiente y envía el resultado final al nodo central. En el otro método presentado se mantiene el mismo código para los trabajadores, no así para el nodo central, el cual para generar las soluciones se usa un algoritmo genético. En [58] se desarrollan métodos paralelos sincrónicos y asíncrónicos que aplican satisfactoriamente a problemas de redes de bases de datos distribuidas en los que en el primer caso un proceso maestro mantiene al algoritmo del Templado Simulado y los esclavos solo realizan movimientos locales que envían al maestro y en el segundo caso los esclavos también hacen movimientos locales pero los guardan en buffers hasta que el maestro lo requiera.

Algunas aplicaciones paralelas descentralizadas son las siguientes: Kliewer [49] propone una biblioteca de programas para el Templado Simulado que brinda, entre sus métodos de paralelización, una combinación de un único camino con múltiples caminos, en la que cada procesador comienza evaluando su propio espacio de búsqueda. Según decrece el nivel de aceptación, los procesadores se asocian en grupos o clusters. Cada cluster trabaja sobre una única subcadena y todos los procesadores tienen una solución común en cada paso del método. El sistema global está compuesto por varios grupos que se comunican entre sí después de que cada nivel de temperatura alcanza su fin. En la fase final todos los procesadores se asocian en un gran grupo. Esto se aplica al problema de la asignación de flota. En [10] se presenta un esquema de paralelización en el que la distribución de Boltzman se muestrea en paralelo por un grupo de procesadores en el que todos ejecutan el algoritmo de Metropolis usando cadenas de Markov. Se controla las estadísticas obtenidas simultáneamente por todos los procesadores, y se mezclan estados a intervalos para asegurar la distribución de energía. Esto se aplica a la determinación de parámetros de un conjunto de ecuaciones diferenciales ordinarias. Otros métodos, como en [56, 61] incrementan el grado de paralelismo usando metodologías cercanas a los algoritmos genéticos.

- Optimización con enjambre de partículas *(Particle Swarm Optimization, PSO)* es una metaheurística evolutiva inspirada en el comportamiento social de las bandadas de pájaros o bancos de peces desarrollada por Eberhart y Kennedy en 1995 [31].

  Comparte similaridades con técnicas de la computación evolutiva, como los Algoritmos Genéticos. Aquí el sistema es inicializado con una población de soluciones aleatorias y hace búsquedas del óptimo actualizando las generaciones. Sin embargo no tiene operadores de cruce ni mutación. En PSO las soluciones, llamadas partículas, se "echan a volar" en el espacio de búsqueda guiadas por la partícula que mejor solución ha encontrado hasta el momento y que hace de líder de la bandada. Cada partícula evoluciona teniendo en cuenta la mejor solución encontrada en su recorrido y al líder.

  El procedimiento también tiene en cuenta el mejor valor alcanzado por alguna de las partículas en su entorno. En cada iteración, las partículas modifican su velocidad hacia la mejor solución de su entorno teniendo en cuenta la información del líder.

  Algunas ejemplos de implementaciones centralizadas son: a partir del algoritmo PSO paralelo sincrónico propuesto por Schutte et al. [78] en el que se actualiza las posiciones y velocidades de las partículas al final de cada iteración del optimizador usando una sincronización global, se presenta por Koh et al. [50] un algoritmo paralelo asincrónico

en el cual un proceso maestro mantiene una cola de partículas listas para enviar a los esclavos. En cuanto un esclavo completa la función de evaluación envía su valor al maestro que lo recibe, chequea la convergencia y actualiza la posición y velocidad de las partículas que es enviada de nuevo a los esclavos para su evaluación. Jin y Rahma-Samii [48] presentan una implementación paralela del PSO para el diseño de antenas de comunicaciones inalámbricas en el cual un proceso maestro sigue las partículas y los resultados de la simulación del diseño y un grupo de nodos esclavos donde las partículas se distribuyen para evaluar la bondad del método de dominio del tiempo de diferencias finitas (FDTD). Las posiciones y velocidades de las partículas se trasmiten desde el maestro y los archivos de configuración del FDTD son generados en los esclavos. En [77] presentan un método de búsqueda global en el cual un nodo maestro asigna diferentes vectores de configuración a los esclavos, encargados de la evaluación del costo que luego reportan al maestro. Esto lo aplican con buenos resultados a un problema de identificación de sistemas biomecánicos.

- *Búsqueda Dispersa (Scatter Search, SS)* [37, 54, 55]. Este método se basa en el principio de que la información sobre la calidad o el atractivo de un conjunto de reglas, restricciones o soluciones pueden ser utilizados mediante la combinación de éstas en lugar de usarlas aisladamente.

  Contempla el uso de un conjunto de soluciones, denominado conjunto de referencia, de buenas soluciones dispersas que sirve, tanto para conducir la búsqueda mejorando las herramientas para combinarlas adecuadamente para crear nuevas soluciones que mejoren las anteriores, como para mantener un grado satisfactorio de diversidad. Por ello, se clasifica como evolutivo; pero no está fundamentado en la selección aleatoria sobre un conjunto relativamente grande de soluciones sino en elecciones sistemáticas y estratégicas sobre un conjunto pequeño.

  Integra la combinación de soluciones con la búsqueda local, que aunque puede tener una estructura de memoria, en la mayoría de los casos se usa una búsqueda lineal convencional.

  Esta es una metaheurística que desarrolla una cooperación descentralizada implícita. Así, en [34] se proponen tres estrategias de paralelización de este método que son evaluados sobre datos del problema de la p-mediana. En [74] se resuelve el problema de rutas de vehículos con ventanas de tiempo. En el diseño se incluyen diferentes algoritmos de resolución específicos de este problema, que hacen que el método propuesto sea robusto y competitivo frente a otros métodos previamente propuestos. Hay otros ejemplos de Búsqueda Dispersa en combinación con otros métodos: en [67] se propone un procedimiento metaheurístico basado en *Scatter Search* para resolver dos problemas de localización: el problema del p-centro y el problema del máximo recubrimiento. El algoritmo incorpora diferentes estrategias: búsquedas locales, *GRASP* y *Path Relinking*. El algoritmo propuesto obtiene resultados similares a los obtenidos por métodos recientes para el problema del p-centro, pero de manera más eficiente. Los datos obtenidos para el problema del máximo cubrimiento son también competitivos; en [26] los autores proponen y evalúan el comportamiento de un *Scatter Search*, un *Path Relinking* y un híbrido entre ambos métodos para el problema de la p-mediana con capacidades. Los experimentos computacionales indican que el método híbrido supera a los otros dos.

- *Reencadenamiento de Trayectorias, (Path Relinking, PR):* Este método [37] trata de volver a unir dos buenas soluciones mediante un nuevo camino. Así, si en el proceso de búsqueda se ha encontrado dos soluciones x e y con un buen valor de la función objetivo, se puede considerar el tomar x como solución inicial e y como solución final e iniciar un nuevo camino desde x hasta y. Para seleccionar los movimientos no se considera la función objetivo o el criterio que se haya estado utilizando sino que van incorporando a x los atributos de y hasta llegar a ésta. En algunas implementaciones se ha considerado el explorar el entorno de las soluciones intermedias para dar más posibilidad al descubrimiento de buenas soluciones.

  En [70] se propone un Path Relinking para el problema del p-hub mediano y se comparan los resultados obtenidos con los obtenidos por una Búsqueda Tabú propuesta anteriormente. La experiencia computacional muestra la mayor eficiencia del método propuesto, sobre todo en problemas grandes. En [80] se presenta un esquema

híbrido entre Búsqueda Tabú y Path Relinking para el problema de rutas de vehículos que es superior a la Búsqueda Tabú pura para el mismo problema. El método híbrido se compara también con otros métodos aparecidos previamente en la literatura. Los resultados obtenidos muestran la validez de la propuesta.

- *Heurísticas basadas en Soft Computing*: Basándose en técnicas de la Soft-Computing, que aprovecha la tolerancia asociada a la imprecisión, la incertidumbre, las verdades parciales y la vaguedad implícita en la definición y funcionalidad de las metaheurísticas, se han desarrollado algoritmos de optimización robustos y adaptables en una gran variedad de situaciones [83] consiguiendo tratabilidad, robustez, soluciones de bajo costo y mejores representaciones de la realidad. Ejemplo de ello son las estrategias paralelas cooperativas multihilos propuestas por Pelta et al. en [68, 69] donde un conjunto de agentes cooperan entre sí a través de un coordinador cuya toma de decisiones es diseñada a través de reglas difusas.

5. Conclusiones

Teniendo en cuenta los estudios presentados anteriormente se concluye que:

- El objetivo de la cooperación en el contexto de las heurísticas es mejorar la calidad de la solución dada a un problema o disminuir la cantidad de recursos necesarios para obtener dicha solución. O sea, aumentar la eficiencia o la eficacia del algoritmo de búsqueda.

- La cooperación puede producirse entre varias heurísticas, o entre elementos de una misma heurística.

- Los esquemas cooperativos pueden clasificarse atendiendo a diversas características. Entre ellas se destaca la forma en que se produce la cooperación, ya sea de forma centralizada o descentralizada.

  En un esquema centralizado, la comunicación entre los *hilos de búsqueda* se realiza a través de un *módulo central o coordinador*; y en un esquema descentralizado, la comunicación se realiza directamente entre los *hilos de búsqueda*. Ya sea en un esquema u otro los mecanismos de cooperación varían de una metaheurística a otra. Prácticamente no existe una comparativa entre métodos cooperativos centralizados y descentralizados en cuanto a sus resultados.

  En la literatura pueden encontrarse varias propuestas éxitosas de técnicas heurísticas cooperativas aplicadas a muchos problemas.

- Los esquemas cooperativos pueden implementarse en un entorno paralelo y esto es lo más común; sin embargo, el uso del paralelismo no es imprescindible en el diseño, implementación y análisis de estrategias cooperativas.

- De entre todos ellos, los métodos paralelos cooperativos multihilos se presentan como los más prometedores no solo porque mejoran la velocidad y el factor de aceleración, sino también la calidad de las soluciones, la eficiencia computacional y la robustez de la búsqueda.

- Últimamente, vinculado a estos métodos, se intenta también aprovechar las ventajas del uso de la memoria, ya sea global para toda la estrategia cooperativa y/o particular de cada algoritmo. Además se han comenzado a usar técnicas de la Soft Computing en la modelación de comportamientos y decisiones más cercanas a la realidad.

- Se podría aprovechar toda la experiencia que brinda la naturaleza, acumulada y refinada durante millones de años tanto en esquemas cooperativos muy simples como en los mas complejos. Hay un campo abierto a la experimentación de estas técnicas y de sus mecanismos de cooperación.

Agradecimientos

El primer autor ha sido financiado parcialmente por los proyectos TIN-2005-08404-C04-01 (HeuriFuzzy) y TIC-00129-JA (MINAS). El segundo autor ha sido parcialmente financiado por el Ministerio de Ciencia y Tecnología (proyecto TIN2005-08404-C04-03 (70 % son fondos FEDER)) y por el Gobierno de Canarias (proyecto PI042004/088).

Referencias

[1] La urraca yucateca. http://www.yucatan.com.mx/especiales/faunaenextincion/urraca.asp.

[2] D. Abramson, G. Mills, and S. Perkins. Parallelisation of a genetic algorithm for the computation of efficient train schedules. In *Proceedings of the 1993 Parallel Computing and Transputers Conference*, pages 139–149, 1993.

[3] E.J. Anderson and Ferris M.C. A genetic algorithm for the assembly line balancing problem. Tr 926, Computer Science Department, University of Wisconsin-Madison, 1990.

[4] T. Bäck, D. Fogel, and Z. Michalewicz, editors. *Handbook of Evolutionary Computation*. Institute of Physics Publishing and Oxford University Press, 1997.

[5] R.K. Belew and L.B. Booker, editors. *Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling*. Morgan Kaufmann, 1991.

[6] A. D. Bethke. Comparison of genetic algorithms and gradient-based optimizers on parallel processors: Eficiency of use of processing capacity. Technical Report 197, University of Michigan, Logic of Computers Group, Ann Arbor, 1976.

[7] B. Bullnheimer, G. Kotsis, and C. Strauss. Parallelization strategies for the ant system, 1997.

[8] B. Bullnheimer, Hartl R.F., and C. Strauss. A new rank based version of the ant system — a computational study. Technical report, University of Viena, Institute of Management Science, 1997.

[9] E. Cantú-Paz. A summary of parallel genetic algorithms. *Calculateurs Parallèls, Réseaux et Systèmes répartis*, 10:141–170, 1998.

[10] K. Chu, Y. Deng, and J. Reinitz. Parallel simulated annealing algorithms by mixing states. *Journal of Computational Physics*, 148:646–662, 1999.

[11] J. Cohoon, W. Martin, and D. Richards. A multi-population genetic algorithm for solving the k-partition problem on hyper-cubes. In R.K. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991.

[12] C. Cotta and P. Moscato. *Handbook of Metaheuristics*, chapter A Gentle Introduction to Memetic Algorithms. Kluwer Academic, 2003.

[13] C. Cotta and P. Moscato. Una introducción a los algoritmos meméticos. *Revista Iberoamericana de Inteligencia Artificial*, 19:131–148, 2003.

[14] Teodor Gabriel Crainic and Michel Gendreau. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8(6):601–627, 2002.

[15] Teodor Gabriel Crainic, Michel Toulouse, and Michel Gendreau. Toward a taxonomy of parallel tabu search heuristics. *INFORMS Journal on Computing*, 9(1):61–72, 1997.

[16] T.G. Crainic and M. Gendreau. Cooperative parallel tabu search for capacitated network design. *Journal of Heuristics*, 8:601–627, 2002.

[17] T.G. Crainic, M. Gendreau, P. Hansen, and N. Mladenovic. Cooperative parallel variable neighborhood search for the p-median. *Journal of Heuristics*, 10:293–314, 2004.

[18] T.G. Crainic and M. Toulouse. *Fleet Management and Logistics*, chapter Parallel Metaheuristics. Kluwer Academic, 1998.

[19] T.G. Crainic and M. Toulouse. *Parallel Strategies for Metaheuristics*, volume Handbook of Metaheuristics, pages 475 – 513. Kluwer Academic Publisher, 2003.

[20] T.G. Crainic, M. Toulouse, and B. Sansó. Systemic behavior of cooperative search algorithms. *Parallel Computing*, 30:57–79, 2004.

[21] V. Cung, S.L. Martins, C. Ribeiro, and C. Roucairol. *Essays and Surveys in Metaheuristics*, chapter Strategies for the Parallel Implementation of Metaheuristics, pages 263–308. Kluwer Academic Publisher, 2001.

[22] B.A. Díaz and K.A. Dowsland. Diseño de heurísticas y fundamentos del recocido simulado. *Revista Iberoamericana de Inteligencia Artificial*, 19:93–102, 2003.

[23] I. De Falco, R. Del Balio, E. Tarantino, and R. Vaccaro. Improving search by incorporating evolution principles in parallel tabu search. In *Proc. Int. Conf. on Machine Learning*, pages 823–828, 1994.

[24] G. Di Caro, L.M. Gambardella, and M. Dorigo. Ant algorithms for discrete optimization. *Artificial Life*, 5:137–172, 1999.

[25] A. Diaz, F. Glover, H. Ghaziri, J. Gonzalez, M. Laguna, P. Moscato, and F. Tseng. *Optimización Heurística y Redes Neuronales*. Ed. Paraninfo, 1996.

[26] Fernández E. Díaz, J.A. Hybrid scatter search and path relinking for the capacitated p-median problem. *European Journal of Operational Research*, 169:570–585, 2006.

[27] J. Digalakis and K. Margaritis. Performance comparison of memetic algorithms,. *Journal of Applied Mathematics and Computation*, 158:237–252, October 2004. http://www.sciencedirect.com/science/article/B6TY8-4B66CNR-1/2/52dae0363c6e521507c08c0c304bd60e.

[28] M. Dorigo and T. Stützle. *Handbook of Metaheuristics*, chapter Ant Colony Optimization Metaheuristic. Kluwer Academic, 2003.

[29] Marco Dorigo and Luca Maria Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, April 1997.

[30] L. A. Dugatkin and H. K. Reeve, editors. *Game theory and animal behavior*. Oxford University Press, 2000.

[31] R.C. Eberhart and J. Kennedy, editors. *Swarm Intelligence*. Academic Press, 2001.

[32] E. Felten, S. Karlin, and S. W. Otto. The traveling salesman problem on a hypercubic, mimd computer. In *Proc. 1985 Parallel Processing Conf.*, pages 6–10, 1985.

[33] F. Garcia, B. Melian, J.A. Moreno, and J.M. Moreno. The parallel variable neighborhood search for the p-median problem. *Journal of Heuristics*, 8:375–388, 2002.

[34] F. Garcia, B. Melian, J.A. Moreno, and J.M. Moreno. Parallelization of the scatter search for the p-median problem. *Parallel Computing*, 29:575–589, 2003.

[35] Johnson D.S. Garey, M.R., editor. *Computers and Intractability: A Guide to the Theory of NP- Completeness*. Freeman, 1979.

[36] F. Glover and M. Laguna, editors. *Tabu Search*. Kluwer Academic, 1997.

[37] F. Glover and R. Martí. *Handbook of Metaheuristics*, chapter Scatter Search and Path Relinking: Advances and applications. Kluwer Academic, 2003.

[38] F. Glover and B. Melián. Búsqueda tabú. *Revista Iberoamericana de Inteligencia Artificial*, 19:29–48, 2003.

[39] V. Gordon and D. Whitley. Serial and parallel genetic algortihms as function optyimizers. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 177–183. Morgan Kaufmann Publishers, 1993.

[40] D.R. Greening. Parallel simulated annealing tecniques. *Physica D*, 42:293–306, 1990.

[41] J.J. Grefenstette. Parallel adaptive algorithms for function optimization. CS-81-19 14, Computer Science Department, Vanderbilt University, 1981.

[42] P. Hansen and N. Mladenovic. An introduction to variable neighborhood search. In S. Voss, S. Martello, I. Osman, and C. Roucairol, editors, *Metaheuristics: Advances and Trends in Local Search Procedures for Optimization*, pages 433–458. Kluwer, 1999.

[43] P. Hansen and N. Mladenovic. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.

[44] P. Hansen and N. Mladenovic. *Handbook of Applied Optimization*, chapter Variable Neighborhood Search. Oxford University Press, 2002.

[45] P. Hansen and N. Mladenovic. *Handbook of Metaheuristics*, chapter Variable Neighbourhood Search. Kluwer Academic, 2003.

[46] P. Hansen, Mladenovic N., and J.A. Moreno. Búsqueda de entorno variable. *Revista Iberoamericana de Inteligencia Artificial*, 19:77–92, 2003.

[47] D. Henderson, S.H. Jacobson, and A.W Johnson. *Handbook of Metaheuristics*, chapter The theory and practice of simulated annealing. Kluwer Academic, 2003.

[48] Nanbo Jin and Yahya Rahmat-Samii. Parallel particle swarm optimization and finite-difference time-domain (pso/fdtd) algorithm for multiband and wide-band patch antenna designs. *IEEE Transactions on Antennas and Propagation*, 53(11), 2005.

[49] Georg Kliewer and Stefan Tschöke. A general parallel simulated annealing library and its application in airline industry. In *Proceedings of the 14th International Parallel and Distributed Processing Symposium (IPDPS 2000), Cancun, Mexico*, pages 55–61, 2000.

[50] Byung-Il Koh, B. J. Fregly, A. D. George, and R. T. Haftka. Parallel asynchronous particle swarm for global biomechanical optimization. http://www.mae.ufl.edu/ fregly/pdfs/iscsb2005a.pdf, 2005.

[51] N. Krasnogor. Self-generating metaheuristics in bioinformatics: The proteins structure comparison case. *Genetic Programming and Evolvable Machines*, 5(2), 2004.

[52] Sagan L. On the origin of mitosing cells. *J. Theor. Biol.*, 14:225–274, 1967.

[53] Sagan L. *Origin of Eukaryotic Cells*. Yale University Press, 1970.

[54] M. Laguna and R. Martí. Scatter search: Diseño básico y estrategias avanzadass. *Revista Iberoamericana de Inteligencia Artificial*, 19:123–130, 2003.

[55] M. Laguna and R. Martí, editors. *Scatter Search. Methodology and Implementations in C*. Kluwer Academic Publisher, 2003.

[56] P.S. Laursen. *Parallel Problem Solving from Nature III. Lecture Notes in Computer Science 866*, chapter Problem Independent Parallel Simulated Annealing Using Selection and Migration, pages 408–417. Springer Verlag, 1994.

[57] A. Le Bouthillier and T.G. Crainic. A cooperative parallel meta-heuristic for the vehicle routing problem with time windows. *Computers and Operations Research*, 32(7):1685–1708, 2003.

[58] Fu-Hsieng Allisen Lee. Parallel simulated annealing on a message-passing multicomputer. Technical report, Utah State University Logan, Utah, 1995.

[59] J. Lis. Parallel genetic algorithm with the dynamic control parameter. In *IEEE 1996 Int. Conf. on Evolutionary Computation*, pages 324–328, 1996.

[60] B. Maderick and P. Spiessens. Fine-grained parallel genetic algortihms. In D. Schaffer, editor, *Proceedings of the Third International Conference on Genetic Algoritms*. Morgan Kauffmann Publisher, 1989.

[61] S.W. Mahfoud and D.E. Goldberg. Parallel recombinative simulated annealing. a genetic algorithm. *Parallel Computing*, 21:1–28, 1995.

[62] M. Malek, M Guruswamy, M. Pandya, and H. Owens. Serial and parallel simulated annealing and tabu search algorithms for the traveling salesman problem. *Annals of Operations Research*, 21:759–84, 1989.

[63] O. Martin and S.W. Otto. Combining simulated annealing with local search heuristics. *Annals of Operations Research*, 63:57–75, 1996.

[64] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 1999.

[65] P. Moscato. *Handbook of Applied Optimization*, chapter Memetic Algorithms. Oxford University Press, 2002.

[66] H. Muhlenbein. *Foundations of Genetic Algortihms*, chapter Evolution in time and space-the parallel genetic algorithm. Morgan Kaufmann Publisher.

[67] Casado S. Pacheco, J. Solving two location models with few facilities by using a hybrid heuristic. a real health resources case. *Computers & Operations Research*, 32:3075–3091, 2005.

[68] D. Pelta, C. Cruz, A. Sancho-Royo, and J.L. Verdegay. *Fuzzy Applications in Industrial Engineering*, chapter Fuzzy Sets based Cooperative Heuristics for Solving Optimization Problems. Springer, 2006.

[69] D. Pelta, C. Cruz, A. Sancho-Royo, and J.L. Verdegay. Using memory and fuzzy rules in a co-operative multi-thread strategy for optimization. *Information Sciences*, 176:1849–1868, 2006.

[70] Almeida F. Moreno-Vega J.M Pérez, M. On the use of path relinking for the p-hub median problem. *Lecture Notes in Computer Science*, 3004:155–164, 2004.

[71] Mitchell A. Potter and Kenneth A. De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer-Verlag, 1994.

[72] D. Janaki Ram, T. H. Sreenivas, and K. Ganapathy Subramaniam. Parallel simulated annealing algorithms. *Journal of Parallel Distrib. Comput.*, 37(2):207–212, 1996.

[73] C. Rego and C. Roucairol. *Meta-Heuristics: Theory and Applications*, chapter A parallel tabu search algorithm using ejection chains for the vehicle routing problem, pages 661–675. Kluwer Academic Publishers, 1996.

[74] Chiang W.C. Russell, R.A. Scatter search for the vehicle routing problem with time windows. *European Journal of Operational Research*, 169:606–622, 2006.

[75] P.M. Pardalos S. Duni Ekisoglu and M.G.C. Resende. *Models for Parallel and Distributed Computation - Theory, Algorithmic Techniques and Applications*, chapter Parallel metaheuristics for combinatorial optimization. Kluwer Academic, 2002.

[76] J. Schulze and T. Fahle. A parallel algorithm for the vehicle routing problem with time window constraints. *Annals of Operations Research*, 86:585–607, 1999.

[77] J. F. Schutte, Byung-Il Koh, J. A. Reinbolt, R. T. Haftka, A. D. George, and B.J. Fregly. Evaluation of a particle swarm algorithm for biomechanical optimization. *Journal of Biomechanical Engineering*, 127(3):465–474, 2005.

[78] J. F. Schutte, J. A. Reinbolt, B. J. Fregly, R. T. Haftka, and A. D. George. Parallel global optimization with the particle swarm algorithm. *Int. J. Numer. Meth. Engng*, 61:2296–2315, 2004.

[79] M. Schwehm. *Parallel Computing and Transputer Applications*, chapter Implementation of genetic algorithms on various interconnection networks, pages 195–203. Amsterdam. IOS Press, 1992.

[80] Gendreau M. Sin, C.H. Path relinking for the vehicle routing problem. *Journal of Heuristics*, 12:55–72, 2006.

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

[82] R. Tanese. Parallel genetic algorithms for a hypercube, booktitle = Proceedings of the Second International Conference on Genetic Algorithms and Their Applications, pages = 177–183, publisher = Hillsdale, NJ, USA. Lawrence Erlbaum Associates, year = 1987,.

[83] J.L. Verdegay, editor. *Fuzzy Sets based Heuristics for Optimization.* Studies in Fuzziness and Soft Computing. Physica-Verlag, 2003.

[84] M. Verhoeven and E. Aarts. Parallel local search. *Journal of Heuristics*, 1:43–65, 1995.

[85] M. Verhoeven and M. Severen. Parallel local search for steiner trees in graphs. *Annals of Operations Research*, 90:185–202, 1999.

[86] R. Wiegand, W. Liles, and K. De Jong. An empirical analysis of collaboration methods in cooperative coevolutionary algorithms.

[87] G.W. Wilkinson. Reciprocal food sharing in the vampire bat. *Nature*, 308:181–184, 1984.