IX Congreso de Ingeniería de Organización
Gijón, 8 y 9 de septiembre de 2005

Algoritmo paralelo basado en agentes inteligentes aplicado al problema de secuenciación de trabajos en flujo uniforme*

José Miguel León Blanco[1], Pedro Luis González Rodríguez[1], Rafael Ruiz Usano[1]

[1] Dpto. de Organización Industrial y Gestión de Empresas. Escuela Superior de Ingenieros de Sevilla. 41092 Sevilla. miguel@esi.us.es, jose@esi.us.es, pedroluis@esi.us.es, usano@esi.us.es

Palabras clave: Algoritmo auto adaptativo, paralelo, secuenciación

1. Introducción

Existen en la actualidad numerosos algoritmos que tratan de hacer frente al problema de secuenciación de trabajos. Algunos de ellos son versiones en paralelo de algoritmos de búsqueda local. Estas versiones en paralelo tratan de alcanzar en un primer momento, menores tiempos de computación para ofrecer soluciones de buena calidad en un tiempo razonable. La segunda mejora, y más importante en la actualidad, es la posibilidad de obtener versiones de los mismos algoritmos que sean más tolerantes a variaciones en las condiciones externas del problema, algoritmos más robustos o menos sensibles. Esto se consigue ajustando, tras gran número de pruebas, los parámetros de la búsqueda. Si cada uno de los procesos que componen el algoritmo paralelo, son capaces de ajustar sus parámetros de búsqueda, se conseguirá optimizar el uso del sistema informático que se emplea.

2. Prototipo para implementación de un algoritmo auto-adaptativo

2.1. Algoritmo de búsqueda local

Un método de búsqueda local que ha dado buenos resultados es el propuesto por Ghosh y Sierksma (2002), llamado *complete local search with memory* o CLM. Este método combina las prestaciones de la búsqueda tabú y el recocido simulado para conseguir buenos resultados en su aplicación al problema del viajante. El algoritmo realiza una búsqueda partiendo de una solución encontrada con otra heurística y mantiene listas con las soluciones evaluadas. Para aceptar soluciones, emplea un parámetro umbral que varía en cada iteración del algoritmo. Este finaliza bien cuando se llena la memoria (el total de soluciones guardadas en las listas) o bien cuando se alcanza determinado número de iteraciones sin haber obtenido mejora.

2.2. Versión en paralelo del algoritmo

El algoritmo se ha implementado en paralelo, León *et al* (2004), siguiendo un esquema maestro esclavo. En este, uno de los procesos controla la ejecución del algoritmo CLM por cada uno de los procesos trabajadores. El proceso principal mantiene una lista central de soluciones y envía la solución inicial junto con los parámetros de la búsqueda a los procesos esclavos. Este algoritmo proporciona buenos resultados en su aplicación a un problema NP-Completo, como el problema de secuenciación de trabajos en un entorno de flujo uniforme.

* Este trabajo se deriva de la participación de sus autores en el proyecto de investigación financiado por CICYT con referencia DPI-2001-3110 titulado “Sistemas Híbridos para un control Integrado de la Producción”.

Una de las cuestiones más complejas en la experimentación con este algoritmo ha sido el afinamiento de los parámetros del algoritmo CLM para obtener soluciones de calidad en un espacio de tiempo razonable. Este ajuste conlleva un plazo largo de experimentación.

2.3. Versión en paralelo basada en agentes inteligentes

En nuestro caso, proponemos como mejora al algoritmo paralelo, una versión basada en agentes inteligentes. Se cambiará la forma de trabajo de cada uno de los procesos. Para ello, se pasará de una versión maestro esclavo, en la que un proceso se encarga únicamente de coordinar al resto a una versión en la que los procesos observan la zona del espacio de soluciones que van explorando, y adaptan su comportamiento a las características de esa zona. De esta forma, se consigue minimizar el número de experimentos necesarios para alcanzar el ajuste de parámetros que ofrece la mejor relación tiempo de proceso/makespan.

Las características de la zona del espacio son las que se recogen en el trabajo de Reeves y Eremeev(2004) en el que los valores de la función objetivo para soluciones próximas entre sí según una métrica, son también cercanos y por tanto se pueden observar valles que conducen al mejor valor de la función objetivo.

Los parámetros que deben adaptar cada uno de los procesos son heredados de la versión inicial del algoritmo: tamaño de memoria, umbral de aceptación de soluciones, número inicial de soluciones exploradas en cada búsqueda y número máximo de iteraciones sin obtener mejora. Un aumento en el tamaño de memoria, junto con el aumento en el número máximo admitido de iteraciones sin mejora, implican una búsqueda en una zona más amplia del espacio de soluciones y a un mayor tiempo de búsqueda. Esta es la fase de diversificación, que se complementa con un valor alto para el umbral de aceptación de soluciones. La fase de intensificación en la búsqueda se implementa reduciendo el umbral de aceptación.

El mayor tamaño de memoria y el mayor número de iteraciones admitido implica normalmente, salvo estancamientos en óptimos locales, una mayor calidad en las mejores soluciones obtenidas, pero como contrapartida, el tiempo de ejecución del algoritmo se alarga enormemente para mejoras en el makespan apenas apreciables. Esto se hace patente en los problemas de mayor dimensión. Por otro lado, la disminución en el tamaño de memoria y número de iteraciones sin mejora puede dar lugar a que el tiempo empleado en el algoritmo CLM, llegue a ser incluso inferior al empleado en la construcción de la solución de partida o al empleado en el postproceso de las soluciones que el algoritmo CLM no ha explorado.

En la anterior versión en paralelo de este algoritmo, un proceso se encargaba de fijar los parámetros de búsqueda de los demás en función de las características de la zona del espacio donde se había encontrado un mínimo local.

Para conseguir que los parámetros se adapten correctamente al comportamiento óptimo, se dota de inteligencia basada en red neuronal a los procesos para que se puedan entrenar con la experiencia obtenida con el algoritmo paralelo inicial.

En esta versión del algoritmo paralelo, no existe un proceso principal. A un número prefijado de iteraciones, los procesos interrumpen su búsqueda para intercambiar los valores de las mejores soluciones encontradas, de forma que una parte de los procesos intensifiquen la búsqueda en zonas prometedoras y otra parte pase a zonas sin explorar. Todos los procesos

adaptan sus parámetros de búsqueda a las características de la zona del espacio de soluciones donde se encuentran.

La variación de algunos parámetros tiene una influencia más evidente que la de otros sobre el comportamiento del algoritmo. Así, un aumento en la cantidad de memoria del algoritmo CLM conducirá a soluciones de mayor calidad, además de un tiempo mayor de exploración. En el mismo sentido influye un incremento en el número de iteraciones permitidas sin obtener mejora en el makespan. La influencia de parámetros como la ley de disminución del umbral de aceptación de soluciones, gobernada por los parámetros alfa y beta, el número de soluciones que se toma de la lista LIVE antes de pasar a otra exploración, agresividad de la búsqueda según las palabras de Gosh y Sierksma, tienen una influencia menos evidente tanto en el tiempo de exploración como en la calidad de las soluciones encontradas.

2.3. Descripción del algoritmo

El primer paso, consiste en la ejecución del algoritmo CLM por cada uno de los procesos, partiendo de una solución proporcionada por la heurística NEH. El uso de una semilla distinta para la generación seudoaleatoria de la solución inicial garantiza sin demasiada pérdida de tiempo que las soluciones de partida son diferentes para cada uno de los procesos/agentes. Esto no quiere decir que no se encuentren próximas en el espacio. Por tanto será necesario implementar mecanismos que garanticen la diversificación por parte de los procesos. Por otro lado, se ha observado que el tiempo necesario para generar una solución de partida es muy amplio con el actual método, sobre todo en los problemas de dimensión más elevada. Es preciso encontrar mecanismos que aceleren la consecución de una solución de partida suficientemente buena en un tiempo razonable.

Una vez mejorada la solución con el algoritmo CLM y unos parámetros tomados de la versión maestro esclavo del algoritmo, los procesos intercambian tanto las B mejores soluciones encontradas, de forma similar a como trabajan otros algoritmos paralelos de exploración, cada proceso entrenará su red neuronal con los resultados de todos los procesos, con lo que se dispone en cada uno de los criterios para variar adecuadamente los parámetros de búsqueda antes de comenzar una nueva exploración.

2.3. Adaptación de los parámetros de búsqueda mediante una red neuronal

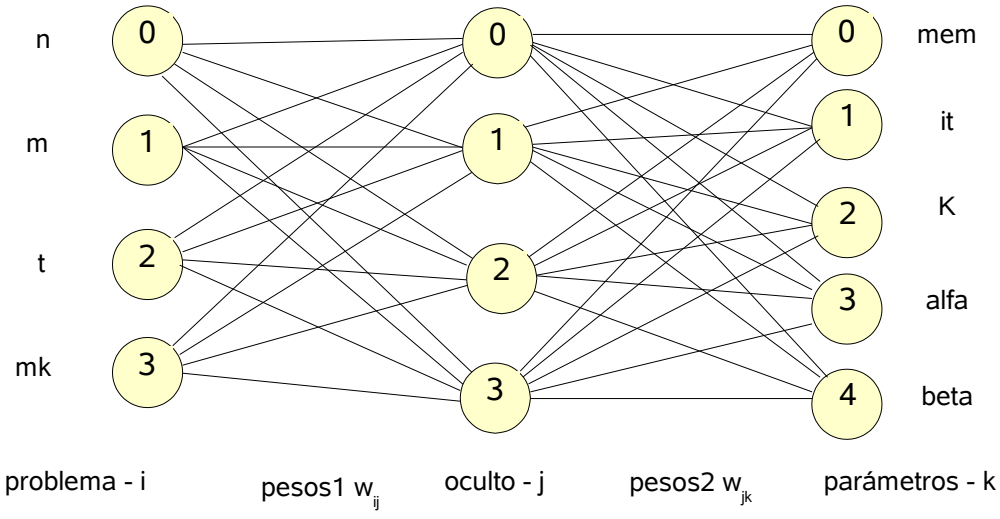

Figura 1. Arquitectura de la red neuronal

El diseño de la red neuronal para que cada proceso decida o no cambiar los parámetros de búsqueda es el de un perceptrón con una sola capa oculta, que se presenta a continuación (ver figura 1).

Cada una de las neuronas i de la primera capa representa las características del problema en cuanto a complejidad, número de trabajos n y número de máquinas m, y en cuanto al tiempo de resolución y calidad de la solución alcanzada en ese tiempo: mk.

Las neuronas de salida k representan los parámetros que conducen a la resolución del problema en el tiempo obtenido. Se trata de conocer a priori, dadas las características del problema, los parámetros que darán como resultado un tiempo de resolución y calidad preestablecida. Por tanto, una vez entrenada la red, será posible prever los parámetros que habría que utilizar para mejorar la calidad de las soluciones tanto en valor de makespan como en tiempo de ejecución.

La función de activación g(x) empleada es la tangente hiperbólica (1)

$$g(x) = \tanh(x) \tag{1}$$

Los resultados que obtienen las diferentes combinaciones de parámetros se evalúan en la neurona de salida como la suma de las activaciones de las diferentes neuronas multiplicadas por sus pesos. De esta forma, en la capa inicial, la función de activación viene dada por (2):

g(entrada) = tanh (entrada) (2)

La respuesta de cada una de las neuronas de la capa oculta se muestra en la expresión (3):

$$salida_j = \sum_{i=0}^{3} w_{ij} \cdot \tanh(entrada_i) \quad para\, i = 0 \ldots 3, \;\; j = 0 \ldots 3 \tag{3}$$

La respuesta de cada una de las neuronas de la capa de salida será, de forma análoga (4):

$$salida_k = \sum_{j=0}^{3} w_{jk} \cdot \tanh(salida_j) \quad para\, k = 0 \ldots 4, \;\; j = 0 \ldots 3 \tag{4}$$

Para el entrenamiento supervisado de la red mediante la lista de soluciones obtenida por cada proceso, se emplea el algoritmo de propagación hacia atrás, con lo que los nuevos pesos vendrán dados por las expresiones (5) para los pesos entre la capa oculta y la final y (8) para los pesos entre la capa de entrada y la capa oculta.

$$w_{jk}^{nuevo} = w_{jk}^{viejo} + ap * \delta_k * salida_k \tag{5}$$

siendo *ap* el coeficiente de aprendizaje y δ viene dado por (6):

$$\delta_k = (1 - salida^2{}_k) * error \tag{6}$$

habiendo calculado el error (7) como la diferencia en valor absoluto entre la salida obtenida por la red y la salida deseada,

$$error = |salida_k - real_k| \tag{7}$$

De la misma forma,

$$w_{ij}^{nuevo} = w_{ij}^{viejo} + ap * \delta_j * salida_j \tag{8}$$

siendo ahora δ dada por (9)

$$\delta_j = (1 - salida^2{}_j) * \sum_{k=0}^{4} w_{jk} * \delta_k \tag{9}$$

El criterio de finalización del entrenamiento es la no disponibilidad de más soluciones en la primera ejecución del algoritmo CLM.

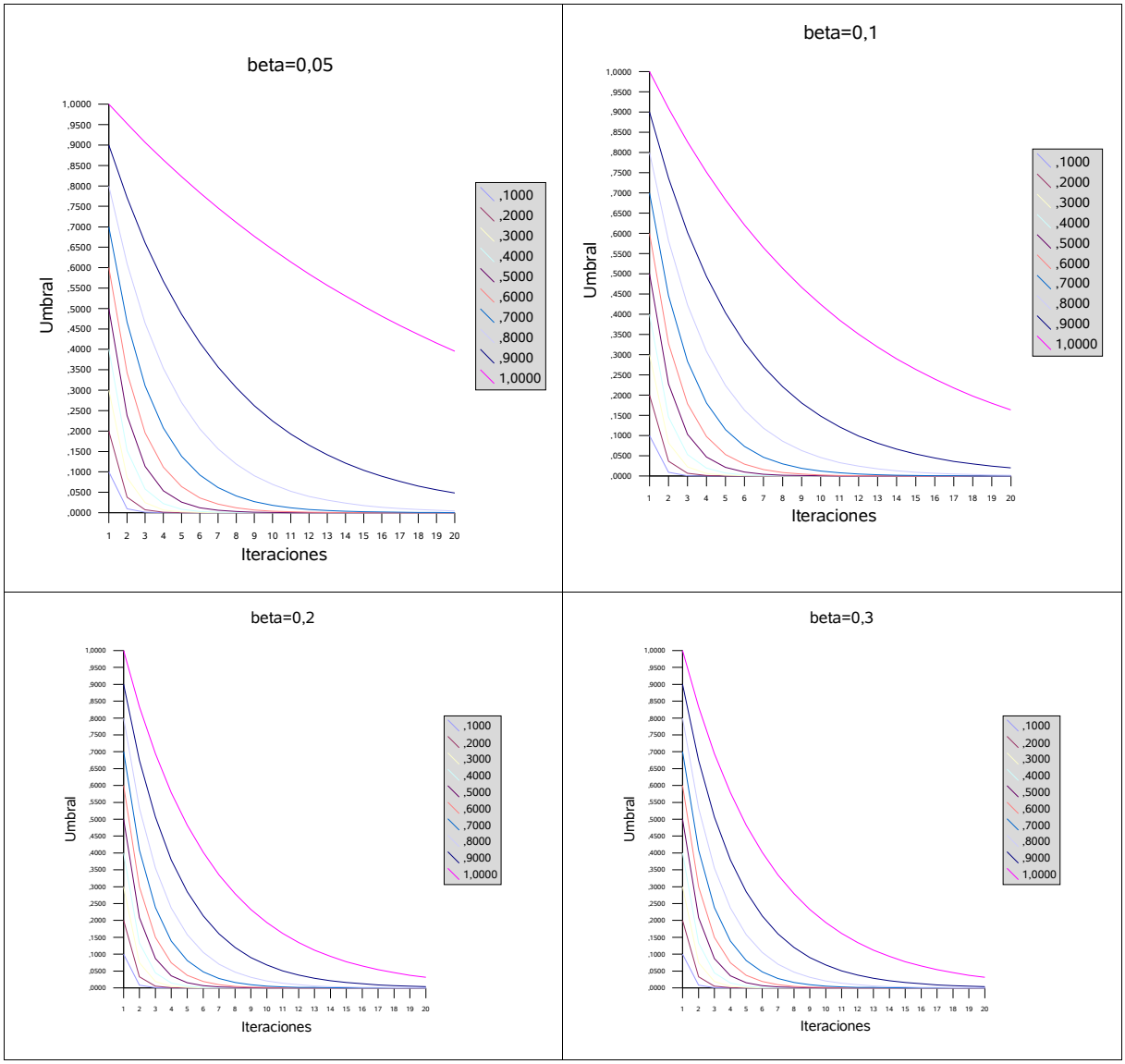

Figura 2: Evolución de los parámetros del umbral

Las variaciones en el umbral de aceptación de soluciones están gobernadas por los parámetros alfa y beta (figura 2).

El entrenamiento de la red neuronal, consiste en valorar si el uso de unos parámetros u otros conduce a una mejora o no en las prestaciones del algoritmo, basándose en la relación entre la mejora porcentual en el valor de la función objetivo, en nuestro caso, el valor de makespan, dividido por el empeoramiento porcentual en el tiempo de resolución.

Una vez entrenada la red, ante una dimensión de problema, y conocidos los parámetros empleados en la exploración anterior, será capaz de decidir si aumentar o no los valores de los parámetros de exploración.

3. Situación actual y líneas de investigación

Las mejoras al algoritmo paralelo deben aún probarse en un entorno multiprocesador. El punto de interés principal consiste en comprobar si la mejora obtenida en calidad de soluciones es suficientemente robusta ante variaciones en los parámetros iniciales del problema. Se estudiará la sensibilidad ante variaciones en los tiempos de proceso y en la complejidad del problema medida por el número de máquinas y el número de trabajos. Una vez comprobada la robustez del algoritmo, se mejorará haciendo que no solo se adapten los parámetros de la búsqueda sino también el número total de procesos-agentes que se ponen en marcha o hacia qué procesador dirigirlos.

La siguiente modificación trata de acercar realmente el algoritmo propuesto a la teoría de agentes, en concreto, dotar de mayor autonomía de funcionamiento a cada proceso, así como definir claramente las relaciones entre los mismos, de forma que se mejore la colaboración en la búsqueda de soluciones.

Otras mejoras al algoritmo son a más largo plazo la experimentación con otras heurísticas para la construcción de la solución de partida. Hasta ahora, se ha empleado la heurística NEH de Nawaz et al., 1983, que históricamente ha dado buenos resultados al encontrar rápidamente una solución de calidad como partida para otros algoritmos en la resolución del problema de secuenciación de trabajos en flujo uniforme. Esta heurística sin embargo, puede llevar a que todos los procesos comiencen a explorar una zona relativamente próxima del espacio de soluciones.

Referencias

Ghosh, D.; Sierksma, G.; (2002). Complete Local Search with Memory. *Journal of Heuristics*, No. 8, pp. 571-584. Kluwer.

León, J.M.; Framiñán, J.M.; González, P.L.; Ruiz Usano, R. (2004). Implementación de meta-heurísticas en paralelo mediante LAM-MPI: Situación actual y perspectivas de futuro. Actas del VIII Congreso de Ingeniería de Organización. pp.121-126.

Nawaz, M.; Enscore Jr., E.; Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA, No. 11(1). pp. 91-95.

Reeves, C.R.; Eremeev, A.V.(2004). Statistical analysis of local search landscapes. Journal of the Operational Research Society. n. 55. pp. 687-693.