Comparación de dos medidas de la eficiencia de algoritmos paralelos

José Miguel León Blanco, José Manuel Framiñán Torres, Pedro L. González-R, Paz Pérez González, Rafael Ruiz-Usano

[1] Dpto. de Organización. Escuela Superior de Ingenieros. Universidad de Sevilla. Camino de los Descubrimientos s/n. 41092. Sevilla. miguel@esi.us.es, jose@esi.us.es, pedroluis@esi.us.es, pazperez@esi.us.es, usano@us.es

Resumen

Se comparan, en este trabajo, dos posibilidades para medir la eficiencia de un algoritmo paralelo, o su capacidad de aprovechar los recursos de un sistema informático paralelo. Por un lado, mediante la comparación de los tiempos de ejecución con uno y varios procesadores. Por otro, mediante la comparación de la calidad de las soluciones obtenidas cuando se emplea el mismo número de iteraciones con uno y con varios procesadores. Los resultados en cuanto a dispersión de los valores de eficiencia para diferentes tamaños de problema, llevan a plantear la aplicabilidad de ambas medidas a uno o a otro tipo de sistema paralelo.

Palabras clave: Algoritmo paralelo, eficiencia, metaheurística

1. Introducción

Uno de los parámetros más empleados para conocer la calidad de un algoritmo paralelo es su eficiencia, que mide el aprovechamiento que hace de los procesadores que conforman el sistema de computación. En este trabajo se comparan, para el problema de secuenciación de trabajos en flujo uniforme sin restricciones de capacidad, dos formas de medir la eficiencia que aparecen con frecuencia en la literatura sobre algoritmos paralelos: por un lado, el tiempo de ejecución y, por otro, el número de iteraciones que debe realizar el algoritmo paralelo en relación al algoritmo secuencial[*].

2. Eficiencia en algoritmos paralelos

La definición de eficiencia de un algoritmo paralelo mediante el tiempo de ejecución sigue la expresión (1):

$$e = \frac{t_1}{n \cdot t_n} \tag{1}$$

donde t_1 es el tiempo necesario para encontrar una solución cuando se emplea un solo procesador y t_n es el empleado en encontrar una solución de la misma calidad que la que se encontró con un procesador empleando ahora n procesadores. El primer objetivo de un algoritmo paralelo es mejorar el tiempo de resolución de problemas frente al algoritmo secuencial, para aprovechar la potencia que ofrecen varios procesadores trabajando de forma simultánea. De este modo, cuando la eficiencia se acerca a 1 (100%), el tiempo necesario con n procesadores es inversamente proporcional al número de los mismos y se está aprovechando casi toda la potencia de cálculo disponible.

[*] Este trabajo se deriva de la participación de sus autores en el proyecto de investigación financiado por CICYT con referencia DPI-2004-02902 titulado "Sistemas Avanzados de Secuenciación y Control".

Otros autores como Wodecki y Bozejko (2002) emplean el número de iteraciones que realiza cada uno de los procesos para contrastar la eficiencia del algoritmo. No se mide ahora el tiempo empleado por el algoritmo sino la calidad de las soluciones obtenidas. Si se obtiene una solución de mejor calidad con más procesadores pero el mismo número total de iteraciones, el algoritmo tiene una eficiencia superior a la unidad. Se puede decir que cada iteración, al emplear más procesadores, produce un mayor salto en la calidad que cuando se emplean menos procesadores, debido en la mayoría de los casos a la comunicación entre los distintos procesos.

Esta medida de la eficiencia tiene la limitación de que para mantener constante el número total de iteraciones, debe obligarse a que cada uno de los procesos realice un número cada vez menor de iteraciones. Cuando se emplean procedimientos no deterministas, es posible que en bastantes ejecuciones del algoritmo no se llegue a la calidad de soluciones obtenidas con un solo procesador. La ventaja de este método es limitar la experimentación a un número de iteraciones conocido de antemano, con lo que el método tendrá mayor validez en algoritmos que emplean gran número de iteraciones, típicamente algoritmos de grano fino.

3. Experimentación

En la experimentación, hemos empleado una versión en paralelo del algoritmo CLM de Ghosh y Sierksma (2002) aplicado al problema de la secuenciación de trabajos en flujo uniforme sin restricciones de capacidad, con el objetivo de minimizar el *makespan*. El esquema de funcionamiento del algoritmo paralelo es el presentado en el trabajo de León et al (2006), cuyo seudocódigo es el siguiente:

Proceso master:

1. Envío inicial: Para cada proceso esclavo
   1.1. Se genera una solución aleatoria y distinta para cada proceso
   1.2. Se añade al pool central
   1.3. Se envía la solución al proceso esclavo
2. Mientras el número total de iteraciones sea menor al fijado
   2.1. Se reciben las b mejores soluciones de un proceso esclavo
   2.2. Se almacenan en el pool central las que sean distintas de las existentes
   2.3. Se genera una nueva solución de partida para el proceso esclavo
   2.4. Se genera una nueva política de exploración para ese proceso
   2.5. Se envía la solución de partida junto con la política de exploración
3. Se envía una señal de parada a los procesos esclavos

Proceso esclavo:

1. Recibe la solución inicial del proceso master junto con la política de exploración
2. Se incluye en la lista LIVE
3. Mientras no se reciba la señal de parada
   3.1. Para un número de iteraciones dado y la lista LIVE no esté vacía
      3.1.1. Se toman k soluciones de la lista LIVE
      3.1.2. Se pasan a la lista DEAD
      3.1.3. Se actualiza el umbral de aceptación de soluciones
      3.1.4. Se generan soluciones vecinas de cada una de las anteriores, tales que estén dentro del umbral de aceptación
      3.1.5. Si no están en ninguna de las listas, se incluyen en la lista NEWGEN
      3.1.6. Si alguna mejora la mejor solución hasta el momento, se actualiza el valor del mínimo

3.1.7. Se pasan todas las soluciones de NEWGEN a LIVE

3.2. Postproceso: Se pasan todas las soluciones que queden en la lista LIVE a DEAD después de una búsqueda local

3.3. Se envían las B mejores soluciones de la lista DEAD al proceso master

3.4. Se vacían las listas DEAD y POOL

3.5. Recibe la siguiente solución de partida junto con la política de exploración y la señal de continuar o parar del master

3.6. Si no recibe la señal de parada, recibe también las soluciones de la lista central y las almacena en la lista POOL.

Para reducir el tiempo total de experimentación, las ejecuciones del algoritmo en un solo procesador se han realizado para un total de 10 iteraciones, con lo que obtenemos una desviación porcentual en promedio (ARPD) respecto a las ofrecidas por los experimentos de la batería de Taillard (1993) del 2,13%. Se han representado los valores de la eficiencia en promedio para cada tamaño de problema por 3 ejecuciones para cada uno de los problemas de tamaño 20x20 hasta 200x20. La eficiencia del algoritmo es muy reducida, ver tabla 1, aunque la convergencia sea muy rápida, ya que se está empleando un algoritmo paralelo de tipo síncrono.

3.2. Medición de la eficiencia según el tiempo de ejecución

Tabla 1. Eficiencia según tiempo de ejecución, de 3 a 12 procesadores, agrupada por tamaño de problema

	e3	e6	e9	e12
20x20	74,90%	39,70%	25,60%	17,90%
50x5	45,10%	25,30%	12,70%	12,40%
50x10	29,70%	19,50%	26,70%	20,20%
50x20	59,70%	29,60%	31,60%	20,00%
100x5	11,80%	5,90%	8,50%	7,70%
100x10	23,40%	11,70%	8,40%	14,80%
100x20	55,20%	27,58%	26,27%	24,58%
200x10	27,50%	13,80%	13,50%	11,10%
200x20	41,00%	42,50%	23,80%	22,60%

Se presentan, a continuación, en forma de gráfico (Figura 1), los resultados anteriores, para poner de manifiesto algunos puntos con anomalías, por ejemplo en el caso de la eficiencia medida con 9 procesadores, mayor que la obtenida con 6 para los problemas de 50 trabajos y 10 ó 20 máquinas. Para más claridad, se han agrupado por un lado los problemas de mayor dimensión, 100 y 200 trabajos (Figura 2) y por otro los de 50 o menos trabajos (Figura 1).

El mismo tipo de anomalías se observan (Figura 2) para algunos de los problemas de mayor dimensión.

3.3. Medición de la eficiencia según el número de iteraciones

En el caso de la medición de la eficiencia comparando el número de iteraciones, deben hacerse varias consideraciones relativas al método empleado para llegar a las cifras presentadas. Para contar el número total de iteraciones realizadas por todos los procesos, se han contado cada una de las veces que los procesos trabajadores terminan su exploración y se comunican con el proceso jefe.

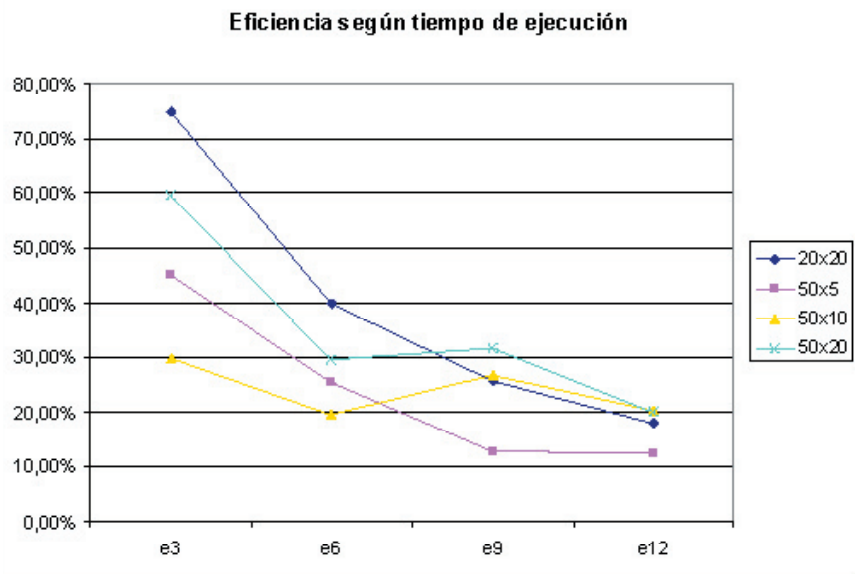

Figura 1. Eficiencia según el tiempo de ejecución para problemas de 20 y 50 trabajos

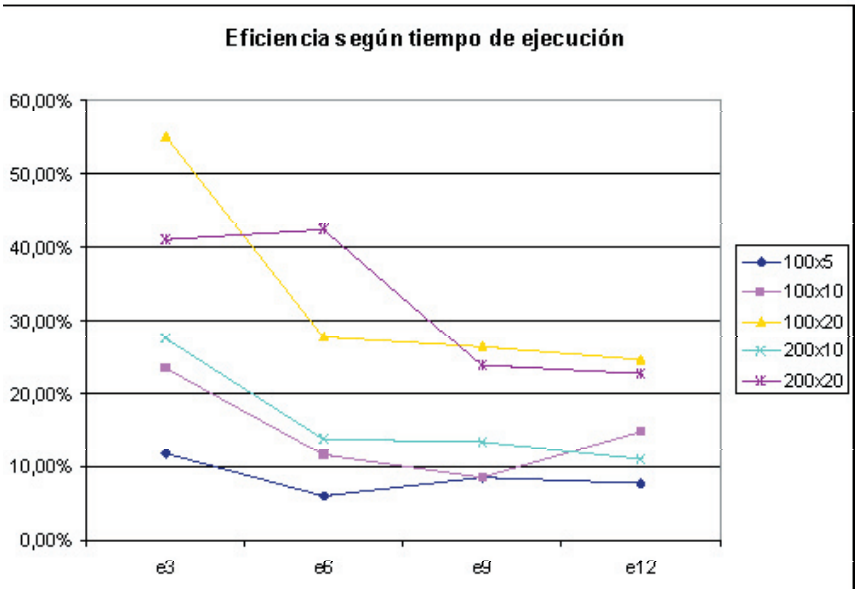

Figura 2. Eficiencia según el tiempo de ejecución para problemas de 100 y 200 trabajos

En general, las cifras obtenidas con este método son superiores a las obtenidas midiendo el tiempo de ejecución.

Por otro lado, no se ha empleado el concepto presentado por Wodecki y Bozejko (2002), que medían la calidad de las soluciones obtenidas con el mismo número de iteraciones, sino que se han comparado el total de iteraciones realizadas por todos los procesos con las 10 iteraciones empleadas por el algoritmo secuencial para llegar a la calidad de soluciones mostrada.

Tabla 2. Eficiencia según el número de iteraciones agrupada por tamaño de problema

	e3	e6	e9	e12
20x20	133,30%	50,50%	17,50%	23,70%
50x5	69,40%	25,50%	14,90%	32,80%
50x10	31,70%	15,10%	14,10%	8,50%
50x20	50,80%	19,60%	13,90%	14,30%
100x5	98,00%	18,30%	14,60%	12,80%
100x10	41,50%	19,50%	15,40%	7,90%
100x20	39,70%	23,40%	13,90%	9,10%
200x10	44,60%	15,30%	11,80%	9,10%
200x20	34,70%	27,10%	23,10%	15,00%

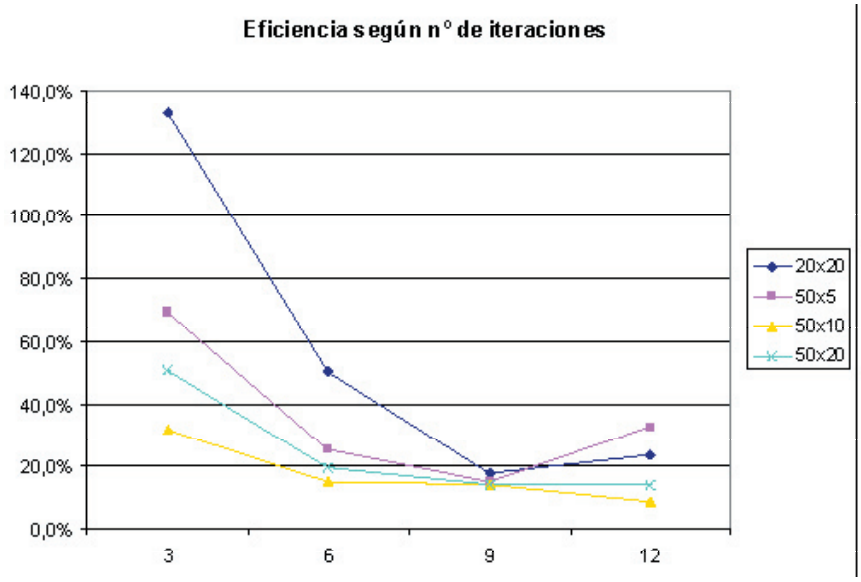

Figura 3. Eficiencia según el número de iteraciones para problemas de 20 y 50 trabajos

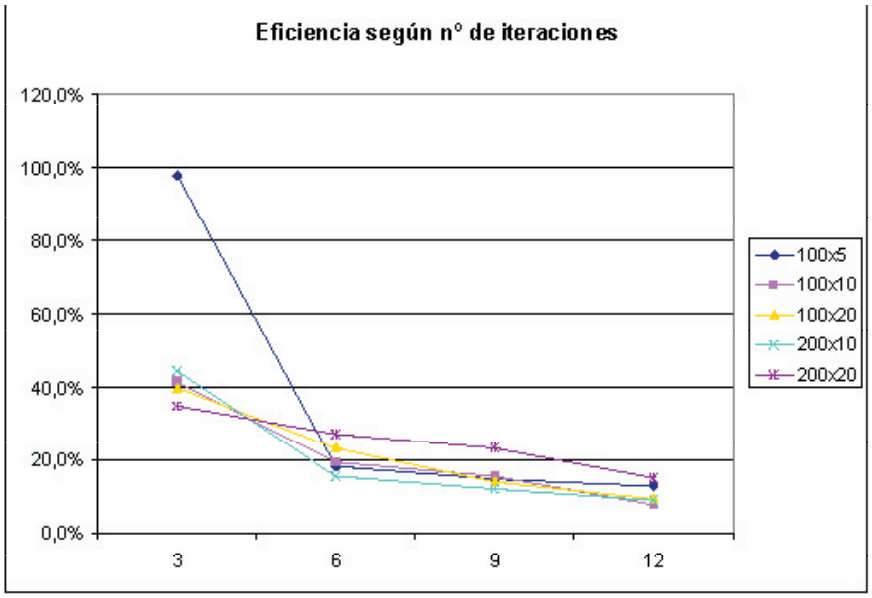

Figura 4. Eficiencia según el número de iteraciones para problemas de 100 y 200 trabajos

4. Conclusiones y ampliaciones del trabajo

Se observan resultados dispares entre ambas formas de medir la eficiencia del algoritmo. Aunque, en general, la eficiencia decae con el número de procesadores, los valores observados no siguen el mismo comportamiento en ambos casos. Por ejemplo, midiendo el tiempo de ejecución, la eficiencia obtenida con problemas de dimensión 100x20 (tabla 1 y figura 2) es mejor que la obtenida con problemas de dimensión 200x10. En cambio, contando el número de iteraciones (tabla 2 y figura 4), nos encontramos con valores de eficiencia prácticamente idénticos. Puede hacerse esta comparación para otras dimensiones de problema.

Por la disparidad de los resultados obtenidos, el empleo del número de iteraciones como medida de la eficiencia de un algoritmo paralelo se ha revelado como un procedimiento altamente inexacto comparado con el resultado de medir el tiempo de computación y debería restringirse a aquellos casos en los que no se disponga de otros elementos para juzgar la eficiencia del algoritmo. En general

En equipos con otra tipología como las redes de transputers, podrían emplearse más iteraciones, con lo que posiblemente se acercarían los resultados de ambas mediciones, y por tanto con dispersión en los resultados se tendría una buena medida de la eficiencia. De este modo puede aprovecharse la sencillez de comparar eficiencias mediante el número de iteraciones frente al tiempo de ejecución. El problema ahora se plantearía en relación con el tipo de algoritmo y las posibilidades de emplearlo en sistemas de bajo coste formados por redes de ordenadores tipo PC, más fácilmente disponibles.

Una extensión a la investigación realizada sería el análisis estadístico de los resultados, que nos daría una medida cuantitativa de la dispersión de los mismos y de la diferencia entre las dos medidas que se están comparando.

Referencias

Alba, E. (2005). Parallel Metaheuristics. A new class of Algorithms. John Wiley & Sons.

Bozejko, W.; Wodecki, M. (2004). The New Concepts in Parallel Simulated Annealing Method. L. Rutkowski et al. (Eds.): ICAISC 2004, LNAI 3070, pp. 853–859.

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-R., P.L.; Pérez, P. (2006). Influencia de la granularidad en las prestaciones de algoritmos paralelos. X Congreso de Ingeniería de Organización.

Taillard, E. (1993). Benchmarks for basic scheduling problems. European Journal of Operational Research, Vol. 64, pp. 278-285.

Wodecki, M.; Bozejko, W. (2002). Solving the Flow Shop Problem by Parallel Simulated Annealing. R. Wyrzykowski et al. (Eds.): PPAM 2001, LNCS 2328, pp. 236–244.