3rd International Conference on Industrial Engineering and Industrial Management
XIII Congreso de Ingeniería de Organización
Barcelona-Terrassa, September 2nd-4th 2009

Algoritmos cooperativos multicore para el problema del taller de flujo de permutación[1]

Gema Escrivá, Eva Vallada

Instituto Tecnológico de Informática. Universidad Politécnica de Valencia. Cno. de Vera s/n, 46022.Valencia.
gema.escriva@gmail.com , evallada@eio.upv.es

Keywords: taller de flujo, tardanza, algoritmos cooperativos

1. Introducción

En el problema del taller de flujo de permutación o *Permutation Flowshop* tenemos un conjunto de N trabajos, $N = \{1,\ldots, n\}$, que son procesados en un conjunto de M máquinas, $M = \{1,\ldots, m\}$. Cada trabajo debe procesarse en todas las máquinas, siendo la secuencia de proceso la misma para todos los trabajos. Además, el orden de los trabajos es el mismo en todas las máquinas por lo que tendremos un total de $n!$ secuencias posibles. El objetivo de optimización más estudiado en la literatura es la minimización del tiempo máximo de finalización de las máquinas o *Makespan.* Sin embargo, cada vez está más extendido el estudio de otros objetivos más realistas, especialmente los relacionados con fechas de entrega de los trabajos. De entre todos ellos, destaca la minimización del retraso o tardanza total en el taller de flujo de permutación, denotado como $F/prmu/\sum T_j$, donde $T_j=\max(0,C_j-d_j)$, siendo d_j la fecha de entrega establecida para el trabajo j y C_j su tiempo de finalización.

En la literatura podemos encontrar métodos muy eficaces y eficientes para el problema definido. Sin embargo, una de las alternativas más prometedoras para mejorar la eficacia de un algoritmo es implementar su versión paralela o cooperativa. En los últimos tiempos la computación paralela vuelve a estar en auge debido a las nuevas tecnologías que nos permiten conectar ordenadores de una manera barata, sencilla y rápida. Por otro lado, las nuevas generaciones de ordenadores personales ya incorporan como mínimo doble núcleo (dual core) en sus procesadores, por lo que es posible paralelizar un algoritmo utilizando un solo ordenador. En este trabajo, se propone un algoritmo genético cooperativo multicore para el problema descrito que aprovecha toda la potencia de las últimas tecnologías de los ordenadores.

2. Algoritmo genético cooperativo multicore

Los algoritmos genéticos pueden verse como técnicas de búsqueda de soluciones basadas en la teoría de la evolución. Este tipo de algoritmos son paralelizables de una manera sencilla, ya que ciertas operaciones son independientes de otras. Si el objetivo a alcanzar es la mejora de

[1] Este trabajo está parcialmente subvencionado por el Ministerio de Ciencia e Innovación, bajo los proyectos "OACS - Optimización Avanzada de la Cadena de Suministro" y "SMPA - Secuenciación Multiobjetivo Paralela Avanzada: Avances Teóricos y Prácticos" con referencias IAP-020100-2008-11 y DPI2008-03511/DPI, respectivamente.

la eficacia del algoritmo y no de su eficiencia, como suele ser habitual en computación paralela, la paralelización consiste en el desarrollo de algoritmos cooperativos. Este tipo de algoritmos se ejecutan de manera distribuida en distintos ordenadores, compartiendo información entre ellos mediante el paso de mensajes. Aprovechando las últimas tecnologías de los ordenadores, que ya incorporan varios núcleos o cores los algoritmos genéticos cooperativos multicore se ejecutan en todos los cores de los ordenadores, de manera independiente y permitiendo la comunicación entre ellos. El algoritmo genético cooperativo multicore propuesto está basado en los trabajos previos de Vallada y Ruiz (2009) y Vallada y Ruiz (2009b). Como características principales destacan la aplicación de búsqueda local acelerada, control de diversidad de la población y aplicación de la técnica Path Relinking.

3. Experimento computacional y resultados

El algoritmo genético cooperativo multicore propuesto se ha comparado con su correspondiente versión serie y con el algoritmo voraz iterativo propuesto en Ruiz y Stützle (2007). Todos los métodos han sido implementados en Delphi 2007. En cuanto al criterio de parada, se ha optado por el tiempo máximo transcurrido de CPU, según la expresión $n \cdot (m/2) \cdot t$, donde $t = 90$ ms. El conjunto de instancias utilizado es el mismo que en Vallada y Ruiz (2009) y Vallada y Ruiz (2009b) donde n varía entre 50 y 350 trabajos y m entre 10 y 50 máquinas. Todos los algoritmos han sido ejecutados en 2, 4, 8 y 12 ordenadores core 2 duo, es decir, se han ejecutado en 4, 8, 16, 20 y 24 cores. En el caso de los algoritmos serie, a los que denotamos como GA (Vallada y Ruiz, 2009b) e IG (Ruiz y Stützle, 2007), se ejecutan de manera aislada en los diferentes cores. En cuanto al algoritmo cooperativo multicore, al que denotamos como CMGA, se ejecutará en los diferentes cores permitiendo comunicación entre ellos. En la tabla 1 podemos observar los resultados obtenidos para el caso de 24 cores, calculando la desviación porcentual relativa (RPD) como $((Método_{sol} - Mejor_{sol})/Mejor_{sol}) *100$.

Tabla 1. Desviación Porcentual Relativa Media para los métodos evaluados (24 cores)

Método	RPD
CMGA	0,76
GA	1,35
IG	1,41

Podemos observar en la Tabla 1, que el algoritmo cooperativo multicore (CMGA) ejecutado en 24 cores obtiene unos resultados un 77% mejor que su correspondiente versión serie ejecutada en 24 cores de manera aislada. Por tanto, dicha mejora solo puede ser atribuida a la comunicación existente entre los algoritmos cooperativos.

Referencias

Ruiz, R; Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research, Vol. 177, pp. 2033-2049 .

Vallada, E.; Ruiz, R. (2009). Genetic algorithms with path relinking for the minimum tardiness permutation flowshop problem. En segunda revisión en OMEGA, the International Journal of Management Science.

Vallada, E.; Ruiz, R; (2009b). Cooperative metaheuristics for the permutation flowshop scheduling problem. European Journal of Operational Research, Vol. 193, pp. 365-376.