Domination analysis for scheduling on non preemptive uniformly related machines

Idan Eisner[1†] and Alek Vainshtein[2]

[1]*Department of Mathematics, University of Haifa, Israel*
[2]*Department of Mathematics and Department of Computer Science, University of Haifa, Israel*

Let S be a problem of non-preemptive scheduling on p uniformly related machines, with total completion time (*Makespan*) to be minimized:
Consider the set $N = \{1, 2, \dots, n\}$, where each $i \in N$ is assigned a positive weight $\sigma(i)$. For a subset $A \subseteq N$ let $\sigma(A) = \sum_{i \in A} \sigma(i)$. A p-partition of N is a p-tuple $\mathcal{A} = (A_1, A_2, \dots, A_p)$ of subsets of N such that $A_1 \cup A_2 \cup \ldots \cup A_p = N$, and $A_i \cap A_j = \emptyset$, for all $1 \le i < j \le p$.

For a set $U = \{u_1, u_2, \dots, u_p\}$, with $u_j > 0$ for each $j = 1, \dots, p$, define

$$T(\mathcal{A}) = \max_{1 \le j \le p} \frac{\sigma(A_j)}{u_j}.$$

The scheduling problem S is, given a triple (N, σ, U) with $|U| = p$, find a p-partition that minimizes $T(\mathcal{A})$ over all p-partitions $\mathcal{A} = (A_1, A_2, \dots, A_p)$ of N.

The problem S is a known NP-complete problem. The special case of two identical machines $U = \{1, 1\}$ is the well known partition problem, which is known to be NP-complete. We therefore settle for polynomial time heuristics, that produce suboptimal solutions. These are sometimes called approximation algorithms, and are usually compared by their performance ratio. Another way of evaluating approximation algorithms is *Domination Analysis* (DA):

The domination number (ratio) of an algorithm $\mathcal{H}$ for a combinatorial optimization problem P is the maximum number (fraction) of all feasible solutions that are not better than the solution found by $\mathcal{H}$ for any instance of P of size n. An algorithm has *Asymptotic Domination Ratio One* (ADRO) if it is of polynomial time complexity, and the limit of its domination ratio when $n \to \infty$ is 1.

Gutin, Jensen and Yeo [Domination analysis for minimum multiprocessor scheduling, Discrete Appl. Math., 154(18):2613-2619, 2006] proved that the minimum multiprocessor problem (that is, the special case of $u_1 =, \dots, = u_j$) admits an ADRO algorithm. We adjust their algorithm to the any $\{u_1, \dots, u_j\}$, and prove that this algorithm $\mathcal{H}$ also has ADRO.

Let s denote the size of the instance (N, σ, U). The algorithm $\mathcal{H}$ is as follows:
If $s \ge p^n$ then we simply solve S optimally. If $s < p^n$, then sort the elements of the sequence $\sigma(1), \sigma(2), \dots, \sigma(n)$. For simplicity of notation, assume that $\sigma(1) \ge \sigma(2) \ge \cdots \ge \sigma(n)$. Compute

[†]Email: eisner@math.haifa.ac.il

$r = \lceil \log n / \log p \rceil$, and solve S for $(\{1, 2, \ldots, r\}, \sigma, U)$ to optimality. Suppose we have obtained a p-partition $\mathcal{A} = (A_1, \ldots, A_p)$ of $\{1, 2, \ldots, r\}$. Now for i from $r + 1$ to n, add i to the set A_j of the current p-partition $\mathcal{A}$ with the smallest $\sigma(A_j \cup \{i\})/u_j$.

Define $V = \sum_{j=1}^{p} u_j$ (i.e, V is the sum of all speeds). Let $\tilde{\sigma} = \frac{\sigma(N)}{V}$. We say that a p-partition $\mathcal{A} = (A_1, A_2, \ldots, A_p)$ of N is *balanced* if

$$\text{for all} j \in \{1, \ldots, p\} \quad \sigma(A_j) < u_j \tilde{\sigma} + \frac{u_j(p-1)}{V},$$

Proposition 1 *Let $\mathcal{P}$ be a scheduling problem on p machines with instance (N, σ, U) and $\sigma(1) \geq \sigma(2) \geq \cdots \geq \sigma(n) = 1$. Then any p-partition that is better than the one obtained by the algorithm $\mathcal{H}$ is a balanced p-partition.*

Proposition 2 *Let $\mathcal{P}$ be a scheduling problem on p machines with instance (N, σ, U) and $\sigma(1) \geq \sigma(2) \geq \cdots \geq \sigma(n) = 1$. Then The number g of balanced p-partitions is less than*

$$p^n \times \left(\sqrt{\frac{8p}{n\pi}} \right)^{p-1} \sqrt{\frac{4}{\pi}}.$$

Using Propositions (1) and (2) we state that:

Theorem 3 *The algorithm $\mathcal{H}$ for S has*

$$\lim_{n \to \infty} domr(\mathcal{H}, n) = 1.$$

Remark. If the number of processors p and the speeds $\{u_j\}$ are not fixed, and depend on the number of jobs n, this can not work. We can give an example that shows that the number g can be bigger than $(n-2)^n$. In this case, we can not apply the same methods for proving $\mathcal{H}$ has ADRO. Note that this does not imply that $\mathcal{H}$ is not an ADRO algorithm. As a matter of fact, in this special example $\mathcal{H}$ gives an optimal solution. But we can not use the same tools to analyze the domination ratio of $\mathcal{H}$.