Scheduling malleable tasks for mean flow time criterion

M.Caramia[1], M.Drozdowski[2]

1 Introduction

In this presentation we study scheduling malleable tasks with limited parallelism, for mean flow time criterion. Malleable tasks can be executed by more than one processor at the same time, while the number of used processors can be changed during execution of the task. More formally, the scheduling problem studied here can be formulated as follows. Set $\mathcal{T}$ of n tasks is to be executed on m parallel identical processors. Each task $j \in \mathcal{T}$ is defined by the parameters: processing requirement p_j, maximum number of processors δ_j that can be used in parallel, ready time r_j, and deadline d_j which cannot be exceeded in any feasible schedule. Tasks can be interrupted, and restarted without cost. A task can migrate to a different processor, increase or decrease the number of used processors without cost, but the number of simultaneously used processors cannot exceed δ_j. Task j is finished when the sum of the lengths of the intervals occupied by task j on all processors is at least p_j. The completion time of task j will be denoted by c_j. The objective is the minimization of the sum of completion times: $\sum_{j=1}^{n} c_j$.

The problem we study can be justified by parallel computer applications. Parallel applications may use more than one processor. The number of processors exploited in parallel in real time depends on the load of the computer system. The number of processors simultaneously used may be limited to avoid overuse of the processor resource by a single program. The $\sum_{j=1}^{n} c_j$ criterion reflects the mean task waiting time. According to the naming conventions of [4] malleable tasks may change the number of exploited processors during the runtime. These should be distinguished from moldable tasks for which the number of used processors must be selected before starting a task. The concept of scheduling tasks using many processors in

[1]Istituto per le Applicazioni del Calcolo, CNR, V.le del Policlinico, 137 - 00161 Rome, Italy.

[2]Institute of Computing Science, Poznan University of Technology, Piotrowo 3A, 60-965 Poznan, Poland. fax: (4861) 8771525, phone: (4861) 6652981. Email: `Maciej.Drozdowski@cs.put.poznan.pl`. The research of this author has been partially supported by Polish Committee for Scientific Research. Corresponding Author.

parallel is explained in more detail in [2, 4]. Problem $P|pmtn, r_j|\sum c_j$ is NP-hard [3], so NP-hard is our problem. Here we present two polynomially solvable cases, and an approximation result.

2 Fixed sequences

When the order of r_j's, d_j's, and c_j's with the respect to each other is known, our problem can be formulated as a linear program. Let $l \leq 3n$ be the number of these events, and let τ_{i-1}, τ_i denote the endpoints of an interval determined by two consecutive events, for $i = 2, \ldots, l$. Let x_{ij} be the amount of processing that task j receives in the interval $[\tau_{i-1}, \tau_i]$. We have the following linear programming formulation:
minimize $\sum_{i=1}^{l} \tau_i$
subject to:

$$x_{ij} \leq \delta_j(\tau_i - \tau_{i-1}) \qquad i = 2, \ldots, l,\ j = 1, \ldots, n \tag{1}$$

$$\sum_{j=1}^{n} x_{ij} \leq m(\tau_i - \tau_{i-1}) \qquad i = 2, \ldots, l \tag{2}$$

$$\sum_{i=2}^{l} x_{ij} \geq p_j \qquad j = 1, \ldots, n \tag{3}$$

$$x_{ij} = 0 \text{ if } \tau_{i-1} < r_j \qquad i = 2, \ldots, l \tag{4}$$

$$x_{ij} = 0 \text{ if } \tau_i > d_j \qquad i = 1, \ldots, l-1 \tag{5}$$

$$\tau_i \leq \tau_{i+1} \qquad i = 1, \ldots, l-1 \tag{6}$$

In the above linear program minimizing $\sum_{i=1}^{l} \tau_i$ is equivalent to minimizing $\sum_{i=1}^{n} c_i$, because τ_i's corresponding to ready times and deadlines are constants. Inequalities (1) guarantee that no task j uses more than δ_j processors in the interval $[\tau_{i-1}, \tau_i]$. By inequalities (2) tasks processed in the interval $[\tau_{i-1}, \tau_i]$ use no more processing than the capacity of the m processors. Inequalities (3) ensure that all tasks receive necessary processing. By constraints (4),(5),(6) tasks are processed within the bounds of their ready times, and deadlines, and the order of the events is preserved. A feasible schedule can be built in each interval $[\tau_{i-1}, \tau_i]$ using an extension of McNaughton's wrap-around rule. If a task is wrapped it may use more than one processor at the same time. By constraints (1) it is guaranteed that no task uses more than δ_j processors simultaneously in interval $[\tau_{i-1}, \tau_i]$.

The above formulation can be extended to the criterion $\sum_{i=j}^{n} w_j c_j$ if the weights w_j of the tasks were given.

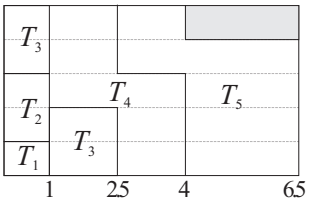

Figure 1: Example from Section 3.

3 Agreeable $\frac{p_j}{\delta_j}$ and δ_j

In this section we propose a low-order polynomial time algorithm for *agreeable* orders of the minimum processing times $\frac{p_j}{\delta_j}$ and parallelism bounds δ_j, i.e. $\frac{p_1}{\delta_1} \leq \frac{p_2}{\delta_2} \leq \ldots \leq \frac{p_n}{\delta_n}$ and $\delta_1 \leq \delta_2 \leq \ldots \leq \delta_n$. The agreeable feature of an instance can be checked in $O(n \log n)$ time by sorting the tasks. We also assume $r_j = 0, d_j = \infty$, for all tasks j. The algorithm can be formulated as follows:

Algorithm Agreeable
1: **for** j:=1 **to** n **do**
2: assign task j to the earliest possible time intervals using maximum possible number of processors, i.e. either δ_j or all the processors remaining available in a given time interval.

Theorem 1 *[1] Algorithm Agreeable constructs the optimum schedule in* $O(n^2)$ *time if* $\frac{p_1}{\delta_1} \leq \frac{p_2}{\delta_2} \leq \ldots \leq \frac{p_n}{\delta_n}$ *and* $\delta_1 \leq \delta_2 \leq \ldots \leq \delta_n$.

Consider an example: $m = 5$, n=5, processing requirements of the tasks are $[1, 2, 5, 9, 13]$, maximum usable numbers of processors are $[1, 2, 2, 3, 4]$. The optimum schedule built by algorithm Agreeable is shown in the Fig.1.

Algorithm Agreeable can be applied in the general, not agreeable case. If the tasks are scheduled according to the increasing processing requirement, i.e. $p_1 \leq p_2 \leq \ldots \leq p_n$, then algorithm Agreeable builds schedules with criterion $\sum_{j=1}^{n} c_j$ not worse than twice the optimum.

Theorem 2 *[1] Algorithm Agreeable has the worst case performance ratio at most 2.*

4 Conclusion

The above problem can be the subject of further work. To our best knowledge, the complexity status of the case without ready times or deadlines remains open. Approximation algorithms for this problem can be also the subject of the further research.

References

[1] M.Caramia, M.Drozdowski, Scheduling malleable tasks for mean flow time criterion, Technical Report RA-008/05, Institute of Computing Science, Poznan University of Technology, 2005 `http://www.cs.put.poznan.pl/mdrozdowski/rapIIn/RA008-05.pdf`

[2] M.Drozdowski, Scheduling parallel tasks - Algorithms and complexity, in: J.Y.-T.Leung, ed., *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapman & Hall/CRC, Boca Raton, 2004, chapter 25.

[3] J. Du, J.Y.-T. Leung, G.H. Young, Minimizing mean flow time with release time constraint, *Theoretical Computer Science* **75**, No.3, (1990) 347-355.

[4] D.G.Feitelson, L.Rudolph, U.Schweigelshohn, K.Sevcik, and P.Wong, Theory and practice of job scheduling, *Lecture Notes in Computer Science* **1291**, Springer, Berlin, (1997) 1-34.