数理解析研究所講究録
1093巻 1999年 11-14

Single Machine Scheduling with Generalized Precedence Relations

Shao Chin Sung* (宋 少秋)
Keisuke Tanaka† (田中 圭介)
Milan Vlach‡ (ミラン・ブラッハ)

Abstract

Single machine scheduling with precedence relations are well studied. In this paper, we extend the standard concept of these precedence relations, and study scheduling problems with several classes of this extended concept.

1 Introduction

We are concern with scheduling n jobs $J_1, J_2, \ldots, J_n$ on a single machine. The feasibility of schedules is constrained by a given precedence relation, and the objective is to minimize the maximum of given non-decreasing functions of completion times. Lawler [2] designed an $O(n^2)$ time algorithm for this problem when precedence relation is derived from a directed acyclic graph G as follows. The vertices of G represent the indices of jobs. If G contains a directed edge from j to k, then the processing of J_k can start only after J_j is completed. We called such precedence relation *traditional.*

In this paper, we introduce a wider concept of precedence which we call *generalized precedence relations.* We design polynomial time algorithms for the problem described above with several classes of generalized precedence relations.

The reminder of this paper is divided into three sections. In Section 2, we introduce the generalized precedence relations, and compare them to the traditional precedence relations. Section 3 is intended to give a brief review on Lawler's algorithm, which our algorithm is based on. Section 4 provides our algorithm and its analysis.

2 Generalized precedence relations

We begin with a simple example. Suppose we have three jobs J_1, J_2, J_3, and the following traditional precedence relation among them: $G = (V, E) = (\{1, 2, 3\}, \{(1, 3), (2, 3)\})$. It implies that both J_1 and J_2 should be completed before J_3 starts. We can interpret this requirement as an AND function.

*School of Information Science, Japan Advanced Institute of Science and Technology, Tatsunokuchi, Ishikawa 923-1292, Japan. son@jaist.ac.jp
†NTT Information Sharing Platform Laboratories, 1-1 Hikarinooka Yokosuka, Kanagawa 239-0847, Japan. keisuke@isl.ntt.co.jp
‡School of Information Science, Japan Advanced Institute of Science and Technology, Tatsunokuchi, Ishikawa 923-1292, Japan. vlach@jaist.ac.jp

In some situation, it is quite natural that we have an OR function rather than an AND function. If we replace this AND function with an OR function, then the precedence relations above implies that at least one of J_1 and J_2 should be completed before J_3 starts. We can also replace the function with a threshold function, a weighted threshold function, or even an arbitrary monotone function. If we have arbitrary monotone functions for precedence relations, we call these precedence relations *generalized.*

Furthermore, we allow different (general) monotone functions for different precedence relations, and consider precedence relations in this manner. This also extends the traditional concept of precedence relations in a different direction.

In the sequel, we follow the standard three-field notation $\alpha|\beta|\gamma$ proposed by Lawler et al. [3]. The first field α indicates the number of machines available and their environment. The second field β indicates a number of job characteristics including precedence relations. The third field γ indicates an objective function. If $\beta =$ prec-AND, it implies that the traditional precedence relation is specified. For problems with generalized precedence relations, we put $\beta =$ prec-monotone in the second field.

In addition to precedence relations, if $\beta =$ pmtn, it implies that preemtions of jobs are allowed; if $\beta = r_j$, it implies that release times of jobs are specified.

3 Lawler's algorithm for $1|\text{prec-AND}|f_{\max}$

The problem $1|\text{prec-AND}|f_{\max}$ has a particularly simple and elegant solution [2]. The objective is to minimize $f_{\max}$, which is defined as follows:

$$f_{\max} = \max_{1\le j\le n} f_j(C_j),$$

where f_j and C_j are the cost function and the completion time of job J_j, respectively. The cost functions f_j's of the jobs J_j's can be quite arbitrary and different from one another, provided only that they are nondecreasing.

Let $N = \{1, 2, \ldots, n\}$ be the index set of jobs, and let $L \subseteq N$ be the index set of jobs without successors in G. For any subset $S \subseteq N$, let $p(S) = \sum_{j\in S} p_j$ be the sum of processing times p_j's of jobs J_j's in S, and let $f^*_{\max}(S)$ denote the cost $f_{\max}$ of an optimal schedule for the jobs indexed by S. Clearly, $f^*_{\max}(N)$ satisfies the following two inequalities:

$$\begin{aligned} f^*_{\max}(N) &\ge \min_{j\in L} f_j(p(N)), \\ f^*_{\max}(N) &\ge f^*_{\max}(N - \{j\}) \text{ for all } j \in N. \end{aligned}$$

Now let J_l with $l \in L$ be such that

$$f_l(p(N)) = \min_{j\in L} f_j(p(N)).$$

We have

$$f^*_{\max}(N) \ge \max\{f_l(p(N)), f^*_{\max}(N - \{l\})\}.$$

But the right-hand side of this inequality is precisely the cost of an optimal schedule subject to the condition that J_l is in the last position. By repeated applications of this rule, one obtains an optimal schedule in $O(n^2)$ time.

Here, we present a pseudo-code of Lawler's algorithm. This returns a resulting schedule A as a permutation, i.e., $A(j)$ is the index of the job in the j-th position for each $1 \leq j \leq n$.

```
Lawler's Algorithm
  u ← Σ_{1≤j≤n} p_j ;
  for k = n down to 1 do
    Find L ;
    Choose l ∈ L such that f_l(u) = min_{j∈L} f_j(u) ;
    Update G ;
    u ← u − p_l ;
    A(k) ← l ;
  od
  Output(A)
end
```

Here, G is updated as follows. If job J_j is assigned to be the last job, then set G to a subgraph of itself that generated by $N - \{j\}$. Notice that, for problem $1|\text{prec-AND}|f_{\max}$, it is easy to find L and to update G in Lawler's Algorithm. Lawler's algorithm has been generalized by Baker, Lawler, Lenstra, and Rinnooy Kan [1] to an $O(n^2)$ algorithm for $1|\text{pmtn}, \text{prec-AND}, r_j|f_{\max}$.

4 Algorithm for $1|\text{prec-monotone}|f_{\max}$

First, let us define feasibility of schedules for the problem $1|\text{prec-monotone}|f_{\max}$. Let h_j for $1 \leq j \leq n$ be the monotone function assigned on vertex k in the directed acyclic graph G. Each h_k is defined on a set X_k of variables such that

$$X_k = \{x_{jk} \mid (j,k) \in E(G)\}.$$

Let A be a schedule. Then, the value of each variable x_{jk} is assigned according to A as follows.

$$x_{jk}|_A = \begin{cases} 1 & \text{if } J_j \text{ is completed before } J_k \text{ starts on } A. \\ 0 & \text{otherwise.} \end{cases}$$

Schedule A is *feasible* if for all $j \in N$,

$$h_j|_A = 1,$$

where $h_j|_A$ is the value of h_j when the value of each variable in X_j is assigned according to A. Note that the feasibility of schedules for this problem is consistent with one for the problem with the traditional precedence relation prec-AND, i.e., h_j is an AND function for all $j \in N$.

For the problem $1|\text{prec-monotone}|f_{\max}$, we use the Lawler's algorithm basically. Note that, in this case, the definition of set L in Lawler's algorithm must be modified. Here we define L to

be the index set of jobs which can be put in the last position. That is, $j \in L$ if for every k with $(j,k) \in E(G)$,

$$h_k|_{x_{jk}=0} \not\equiv 0.$$

By the same argument for the problem with traditional precedence relations $1|\text{prec-AND}|f_{\max}$, it is clear that Lawler's algorithm with the modified L also finds an optimal schedule for our problem $1|\text{prec-monotone}|f_{\max}$.

In the following, we present the procedure for finding L and updating G for the case that h_k is a weighted threshold function for every $k \in N$. Let T_k and W_k be the threshold and the sum of weights of all variables of h_k, respectively, and let w_{jk} be the weight of variable x_{jk}. Then,

$$h_k|_{x_{jk}=0} \not\equiv 0 \quad \Leftrightarrow \quad W_k - w_{jk} \geq T_k.$$

Hence, for every $1 \leq j,k \leq n$, we can decide in $O(1)$ time whether $h_k|_{x_{jk}=0} \not\equiv 0$ or not. Therefore, L can be found in $O(n^2)$ time.

Suppose job J_j is assigned to be the last job. Then, G is first updated as in Lawler's algorithm, and the monotone function h_k assigned on each vertex k in G is defined by $h_k := h_k|_{x_{jk}=0}$. It is obvious that the updated h_k is also a weighted threshold function for every $k \in N - \{j\}$. Therefore, for the case that all h_k's are weighted threshold functions, the time complexity for updating G is $O(n^2)$.

Hence, Lawler's algorithm with our procedure for finding L and updating G runs in time $O(n^3)$. For the case that h_j is a arbitrary monotone function, the time complexity for initially finding and updating L depends on one for h_j.

References

[1] BAKER, K. R., LAWLER, E. L., LENSTRA, J. K., AND RINNOOY KAN, A. H. G. Preemptive scheduling of a single machine to minimize maximum cost subject to release dates and precedence constraints. *Operations Research 31* (1983), 381–386.

[2] LAWLER, E. L. Optimal sequencing of a single machine subject to precedence constraints. *Management Science 19* (1973), 544–546.

[3] LAWLER, E. L., LENSTRA, J. K., RINNOOY KAN, A. H. G., AND SHMOYS, D. B. Sequencing and scheduling: Algorithms and complexity. In *Handbooks in Operations Research and Management Science, Volume 4—Logistics of Production and Inventory*, S. C. Graves, A. H. G. Rinnooy Kan, and P. H. Zipkin, Eds. North-Holland, Amsterdam, 1993, ch. 9, pp. 445–522.