Journal of the Operations Research
Society of Japan
Vol. 37, No. 3, September 1994

ON FINDING THE MOST VITAL JOB IN RELOCATION PROBLEM

Bertrand M.-T. Lin
Ming-Chuan College of Management

(Received June 15, 1992; Final March 19, 1993)

Abstract In this paper, we consider a variant of relocation problem, which originated from Public Housing Projects in Boston area. In the relocation problem, there is a set of h jobs to be processed on a single machine without preemptions. Each job demands, from the resource pool, a fixed amount of resources for its processing and returns some amount of resources to the pool at its completion. The number of resources returned by a job is not necessarily equal to that demanded. The minimum resource requirement is the number of resources that should be initially stored in the pool such that all jobs can be successfully completed. Relocation problem seeks to find a schedule which guarantees the minimum resource requirements. The question we investigate here relates to the flexibility, for decision makers, that a job can be left unprocessed. The goal is to find a job such that the resource requirement for scheduling the remaining jobs is minimized. Naive and efficient methods are proposed and discussed.

1. Introduction

Resource-constrained scheduling problems have been widely studied in the past decades [1]. In this paper, we consider a specialized resource-constrained problem, called **relocation problem**. There is a set of jobs to be processed on a single machine without preemptions. All jobs are available from scratch. And, a common pool of resources is given. Each individual job demands a fixed amount of resources from the pool for its ongoing processing. At the completion, this job returns some number of resources to the pool. The number of resources returned is not necessarily equal to that required. That is, the successful completion of a job may make a positive or negative contribution. The relocation problem is to determine the minimum number of resources that must be initially stored in the pool so that there exists some schedule within which jobs can be successfully processed. This problem was proposed by Kaplan [2] and originated from the public housing projects in Boston area. In the redevelopment project, there were a set of buildings to be torn down and erected. The capacities of a building before and after its redevelopment respectively correspond to numbers of resources demanded and returned by a job. The authority had to house residents of the building being reconstructed. The problem seeks to arrange a reconstruction sequence of buildings under the budget constraint. As a matter of fact, the relocation problems also have relevance to memory management in database systems and operating systems [8].

Flexibility is an important issue in decision making. In this paper, we consider the case in which the authority get the chance to leave some job unscheduled due to some constraints, say budget limits. Once a job is ignored the resource requirement for the remaining jobs may be increased, decreased or unaltered. The problem concerned is how to select a job such that the resource requirement for processing the remaining jobs is minimized. Such a job is said to be *the most vital*. The topic of the most vital elements has been mentioned in the literature. For example, [3][4][5] dealt with problems of finding the most vital edges

with respect to important graph parameters, such as the weight of a minimal spanning tree and the number of spanning trees.

We now formally describe the problem of concern.

Notations:
$\mathcal{J} = \{J_1, J_2, \ldots, J_h\}$: The set of jobs to be processed.
n_i: Number of resources job J_i demands.
a_i: Number of resources job J_i returns.
δ_i: Contribution made by job J_i, i.e. $a_i - n_i$.
S: A schedule for jobs in $\mathcal{J}$.
$J_{[i]}$: The i-th job in schedule S.
$V_{[i]}$: Number of resources in the pool at the time when the i-th job is finished.
$R(S)$: Number of initial resources required for finishing jobs under schedule S.

A schedule or sequence is a permutation of jobs in $\mathcal{J}$. With a fixed amount of initial resources, $V_{[0]}$, schedule $S = J_{[1]}J_{[2]}\cdots J_{[h]}$ is said to be **feasible** if $n_{[i]} \leq V_{[i-1]}$ for every job $J_{[i]}$, $1 \leq i \leq h$, where $V_{[i-1]}$ denotes the number of available resources in the pool when job $J_{[i-1]}$ is finished. This inequality means that sufficient resources are needed for the ongoing processing of each job. Note that $V_{[i-1]} = V_{[0]} + \sum_{k=1}^{i-1} \delta_{[k]}$, where $\delta_{[k]} = a_{[k]} - n_{[k]}$ is referred as the contribution made by job $J_{[k]}$. The minimum value of $V_{[0]}$ that guarantees the existence of feasible schedules for job set $\mathcal{J}$ is called the **minimum resource requirement** $R^*_{\mathcal{J}}$ for $\mathcal{J}$. For an arbitrary job $J_i \in \mathcal{J}$ we use $\mathcal{J} - J_i$ to denote $\mathcal{J} - \{J_i\}$. Similarly, $S - J_{[i]}$ denotes the schedule resulting from the removal of the i-th job from schedule S. A job $J_i \in \mathcal{J}$ is called the **most vital job** if $R^*_{\mathcal{J}-J_i} \leq R^*_{\mathcal{J}-J_j}$ for any $J_j \neq J_i \in \mathcal{J}$. Given a set of jobs, we seek to find the most vital job. Note that the solution is not necessarily unique. In many cases, more than one job can be most vital.

In the next section, we will present a naive approach for dealing with this problem. An efficient algorithm will be given and analyzed in Section 3.

2. A Naive Approach

With a fixed amount of initial resources, the problem of deciding if there is a feasible schedule can be answered in $O(h \log h)$ time. This algorithm was proposed by Kaplan and Amir [6]. It consists of three major steps and is outlined as below:

Algorithm KA
INPUT: Job set $\mathcal{J}$ and a fixed amount of initial resources, $V_{[0]}$.
OUTPUT: Yes/No answer to the existence of feasible schedules.

Step 1 Partition $\mathcal{J}$ into two disjoint subsets $\mathcal{J}^+$ and $\mathcal{J}^-$ such that $\mathcal{J}^+$ contains jobs having non-negative contributions and $\mathcal{J}^-$ contains jobs having negative contributions.

Step 2 Derive a sequence S^+ (S^-) by arranging jobs in $\mathcal{J}^+$ ($\mathcal{J}^-$) in non-decreasing (non-increasing) order of n_i (a_i).

Step 3 If the concatenated sequence S^+S^- is feasible then return "YES" else return "NO".

Steps 1 and *3* together take $O(h)$ time, while *Step 2* takes $O(h \log h)$ time for arranging jobs in the desired order. Hereafter, $S^*_{\mathcal{J}}$ denotes the resultant sequence S^+S^- in *Step 3*. The correctness of this solution method stems from the fact [2][6] that $R(S^*_{\mathcal{J}}) = R^*_{\mathcal{J}}$.

From the result of Algorithm KA, we immediately come up with a simple algorithm

that finds the most vital job by first deriving the schedule $S^*_{\mathcal{J}-J_i}$ for each job J_i and then selecting the one whose resource requirement is minimum. A problem however arises here is how to calculate the resource requirement for schedule $S^*_{\mathcal{J}-J_i}$. Now, we describe a method for computing $R(S)$ for schedule S. Suppose that $V_{[0]} = 0$. Denote the number of supplementary resources that should be additionally afforded so as to ensure the successful completion of job $J_{[k]}$ by $q_{[k]}$, $1 \leq k \leq h$. Then, $q_{[k]}$ can be formulated as

$$q_{[k]} = \max\{0, n_{[k]} - V_{[k-1]}\},\ 1 \leq k \leq h.$$

Note that

$$V_{[k]} = \sum_{i=1}^{k} q_{[i]} + \sum_{i=1}^{k} \delta_{[i]},\ 1 \leq k \leq h.$$

We cite a useful lemma stated by Lin [8].

LEMMA 1 *Resource requirement $R(S)$ for schedule S is equal to $\sum_{i=1}^{h} q_{[i]}$.*

EXAMPLE 1 Consider the following set of seven jobs.

$\mathcal{J}$	J_1	J_2	J_3	J_4	J_5	J_6	J_7
n_i	4	8	5	21	7	19	18
a_i	7	5	15	17	16	12	11

Let schedule $S = J_1J_3J_5J_4J_6J_7J_2$. The number of supplementary resources for each job can be determined as the followings:

i	0		1		2		3		4		5		6		7
$V_{[i]}$	0		7		17		26		22		15		11		8
job		J_1		J_3		J_5		J_4		J_6		J_7		J_2	
$q_{[i]}$		4		0		0		0		0		3		0	

From the derivation tabulated above, we know that $R(S) = 4 + 3 = 7$. To verify the feasibility, we let $V_0 = 7$. It can be seen that all jobs can be successfully completed if they are processed in accordance with the order specified by S.

i	0		1		2		3		4		5		6		7
$V_{[i]}$	7		10		20		29		25		18		11		8
job		J_1		J_3		J_5		J_4		J_6		J_7		J_2	

□

Computing the resource requirement for schedule $S^*_{\mathcal{J}-J_{[i]}}$ can be done in $O(h)$ time. As a consequence, our naive procedure takes $O(h^2)$ time in total. Nevertheless, there leaves a room for further improvement. In the next section, we present an efficient algorithm with a time complexity $O(h \log h)$.

3. Algorithm for the Most Vital Job

In this section, we present an efficient method, by accelerating the computation of resource requirements for schedules, for finding the most vital job. We first introduce the concept of **generalized jobs** which was mentioned in [7]. Within a specific job schedule S, a

generalized job $J_{[i:j]}$ corresponding to subsequence $J_{[i]}J_{[i+1]}\cdots J_{[j]}$, $1 \le i \le j \le h$, is defined by letting

$$n_{[i:j]} = R(J_{[i]}J_{[i+1]}\cdots J_{[j]});$$
$$a_{[i:j]} = n_{[i:j]} + \sum_{k=i}^{j} \delta_{[k]}.$$

LEMMA 2 *For an arbitrary schedule S,*

$$R(S) = R(S'),$$

where $S' = J_{[1]}J_{[2]}\cdots J_{[i-1]}J_{[i:j]}J_{[j+1]}\cdots J_{[h]}$, $1 \le i \le j \le h$.

PROOF: We prove that the summations of supplementary resources for jobs in schedules S and S' are the same. It is easy to see that $q_{[k]} = q'_{[k]}$, $k = 1, 2, \ldots, i-1$, where $q'_{[k]}$ for jobs under S' is similarly defined as $q_{[k]}$. When considering solely the sequence $T = J_{[i]}J_{[i+1]}\cdots J_{[j]}$, its resource requirement $R(T)$, which is equal to $n_{[i:j]}$, is derived from the summation of supplementary resources for jobs $J_{[i]}, J_{[i+1]}, \ldots,$ and $J_{[j]}$. Within S,

$$\sum_{k=i}^{j} q_{[k]} = \max\{R(T) - V_{[i-1]}, 0\},$$

which is equal to $q'_{[i:j]} = \max\{n_{[i:j]} - V_{[i-1]}, 0\}$. Because, within schedules S and S', the numbers of available resources at the time when jobs $J_{[j]}$ and $J_{[i:j]}$ are respectively completed are the same, $q_{[k]} = q'_{[k]}$ $(k = j+1, \ldots, h.)$ We conclude that $R(S) = R(S')$. □

This Lemma dictates the fact that the effects of processing $J_{[i:j]}$ is the same as that of processing subsequence $J_{[i]}J_{[i+1]}\cdots J_{[j]}$.

When considering the removal of job $J_{[i]}$ from schedule $S^*_{\mathcal{J}}$, we know that $S^*_{\mathcal{J}-J_{[i]}}$ is not necessarily the same as $S^*_{\mathcal{J}}-J_{[i]}$. $R(S^*_{\mathcal{J}-J_{[i]}})$ is however equal to $R(S^*_{\mathcal{J}}-J_{[i]})$. We may without loss of generality assume that $S^*_{\mathcal{J}-J_{[i]}} = S^*_{\mathcal{J}} - J_{[i]}$. Therefore, $S^*_{\mathcal{J}-J_{[i]}}$ is composed of two subsequences $J_{[1]}J_{[2]}\cdots J_{[i-1]}$ and $J_{[i+1]}J_{[i+2]}\cdots J_{[h]}$. If the values of parameters pertaining to generalized jobs $J_{[1:i-1]}$ and $J_{[i+1:h]}$ are known, the resource requirement for $S^*_{\mathcal{J}-J_{[i]}}$ can be calculated in constant time. We describe a preprocessing algorithm Generalized_Jobs for obtaining desired information.

Algorithm Generalized_Jobs
INPUT: Job set $\mathcal{J}$ and schedule $S^*_{\mathcal{J}}$.
OUTPUT: $n_{[1:i]}, a_{[1:i]}, n_{[i:h]}, a_{[i:h]}$ for each i.
Step 1 For $i = 2$ to h do *Steps 1.1* and *1.2*
  Step 1.1 $n_{[1:i]} = n_{[1:i-1]} + \max\{0, n_{[i]} - a_{[1:i-1]}\}$.
  Step 1.2 $a_{[1:i]} = n_{[1:i]} + \sum_{k=1}^{i} \delta_{[k]}$.
Step 2 For $i = h-1$ downto 1 do *Steps 2.1* and *2.2*
  Step 2.1 $n_{[i:h]} = n_{[i]} + \max\{0, n_{[i+1:h]} - a_{[i]}\}$.
  Step 2.2 $a_{[i:h]} = n_{[i:h]} + \sum_{k=i}^{h} \delta_{[k]}$.
Step 3 Stop.

This algorithm determines the role $J_{[1:2]}$ plays by composing jobs $J_{[1]}$ and $J_{[2]}$. Subsequently, the fusion of $J_{[1:2]}$ and $J_{[3]}$ induces $J_{[1:3]}$. Similar operations are carried out in a

simple iterative loop to deliver $J_{[1:4]}, \ldots, J_{[1:h-1]}$. Another loop iterates in a backward fashion and successively outputs $J_{[h-1:h]}, J_{[h-2:h]}, \ldots, J_{[2:h]}$. Since *Steps 1.1, 1.2, 2.1* and *2.2* all take O(1) time, the time complexity of Algorithm Generalized_Jobs is dominated by $O(h)$.

In the next page, we describe our algorithm Most_Vital_Job for finding the most vital job for a given job set.

Algorithm Most_Vital_Jobs
INPUT: Job set $\mathcal{J}$ and schedule $S^*_{\mathcal{J}}$.
OUTPUT: The most vital job MVJ.
Step 1 Call **Algorithm KA** to find schedule $S^*_{\mathcal{J}}$.
Step 2 Based on $S^*_{\mathcal{J}}$, call **Algorithm Generalized_Job.**
Step 3 $MVJ = J_{[1]}$, $Diff = \infty$.
Step 4 For $i = 1$ to h do *Step 5*
Step 5 If $R(J_{[1:i-1]}J_{[i+1:h]}) < Diff$ then
$Diff = R(J_{[1:i-1]}J_{[i+1:h]})$ and $MVJ = J_{[i]}$.
Step 6 Stop.

Before come to our conclusion theorem, we further emphasize that for a specific job $J_{[i]}$ $R(S^*_{\mathcal{J}-J_{[i]}}) = R^*_{\mathcal{J}-J_{[i]}}$, although it is clear.

THEOREM 1 *Given a set of h jobs, the most vital job with respect to minimum resource requirement can be found in $O(h \log h)$ time.*

PROOF: Finding the unique schedule $S^*_{\mathcal{J}}$ in *Step 1* takes $O(h \log h)$ time because of the sorting stage. The time complexity of *Step 2* is dominated by $O(h)$. After applying Algorithm Generalized_Job, for each job $J_{[i]}$ we compute $R(S^*_{\mathcal{J}-J_{[i]}})$ in constant time by composing two generalized jobs $J_{[1:i-1]}$ and $J_{[i+1:h]}$ as done in *Step 5*. Therefore, finding the most vital job can be done in $O(h \log h)$ time. The proof is completed. □

4. Concluding Remarks

In this paper, we investigated the problem of determining the most vital job in relocation problem, which is a variation of conventional resource-constrained job scheduling problems. We have proposed a simple but efficient algorithm to achieve the goal. Our algorithm takes $O(h \log h)$ time, while a naive approach will take $O(h^2)$ time.

The concept of the most vital object leads to a new field of research, as a variety of problems solvable in polynomial time can be re-studied. On the other hand, relocation problems of new objectives under different constraints are also interesting and challenging. For example, let us consider the variant where each job is associated with processing length. Under a fixed amount of given resources, we are asked to determine a feasible schedule, if exists, whose mean flow time is minimum. The problem of determining if it is NP-hard is still open.

5. Acknowledgements

The author thanks to the anonymous referees for their constructive comments, which greatly improved the original presentation.

References

[1] J. Blazewicz, J.K Lenstra, and A.H.G. Rinnoy Kan, Scheduling subject to resource constraints: classification and complexity, *Discrete Applied Mathematics* **5**(1989) 11-24.

[2] E.H. Kaplan, Relocation models for public housing redevelopment programs, *Planning and Design* **13**(1986) 5-19.
[3] H.W. Corley and D.Y. Sha, Most vital links and nodes in weighted graphs, *Operations Research Letters* **1**(1982) 157-160.
[4] L.H. Hsu, R.H. Jan, Y.C. Lee, C.N. Hung and M.S. Chern, Finding the most vital edge with respect to minimum spanning trees in weighted graphs, *Information Processing Letters*, **39**(1991) 277-281.
[5] L.H. Hsu, T.Y. Sung, F.S.P. Tsen and M.Y. Lin, Finding the most vital edge with respect to minimum spanning trees in weighted graphs, *private communications.*
[6] E.H. Kaplan and A. Amir, A fast feasibility test for relocation problems, *European Journal of Operational Research* **35**(1988) 201-205.
[7] B.M.T. Lin and S.S. Tseng, Generating the best K sequences in relocation problems, *European Journal of Operational Research* **37**(1993) 357-362.
[8] M.T. Lin, A Study of Resource Relocation Problems, A Ph.D. Dissertation, Department of Computer Science and Information Engineering (November 1991) National Chiao-Tung University, Taiwan, Republic of China.

Bertrand Miao-Tsong LIN:
Department of Information Management
Ming-Chuan College of Management
Shih-Lin, Taipei 11120
Republic of China