Journal of the Korean Institute of Industrial Engineers
Vol. 29, No. 3, pp. 190-196, September 2003.

주어진 구간 내의 기계에서만 생산 가능한 병렬기계문제에 대한 2-근사 알고리듬

황학진 · 김규태†

조선대학교 산업공학과

2-Approximation Algorithm for Parallel Machine Scheduling with Consecutive Eligibility

Hark-Chin Hwang · Gyutai kim

Department of Industrial Engineering Chosun University, Gawngju, 501-759

We consider the problem of scheduling *n* jobs on *m* machines with the objective of minimizing makespan. Each job cannot be eligible to all the machines but to its consecutively eligible set of machines. For this problem, a 2-approximation algorithm, MFFH, is developed. In addition, an example is presented to show the tightness of the algorithm.

Keywords: parallel machine scheduling, analysis of algorithm, consecutive eligibility

1. Introduction

When scheduling jobs on machines, we are often confronted with a situation where each job is not equally important and machines have different capabilities to each other. An important job might be originated from a customer of high priority, or impending requirements that could not be resolved within previous production period, etc. Likewise, no two machines are of the same but a machine can produce products of high quality and lower defects to another one.

Formally, we are given a set of machines $M=\{1,\cdots,m\}$ where machine i is thought to have better capability than machine $i+1$. On the m machines, we shall schedule n jobs, $J=\{1,\cdots,n\}$. Each job j has processing time p_j and has a range of machines, called *consecutively eligible* set, $\overline{E_j}\subseteq M$, only one of which the job can be assigned to. $\overline{E_j}$ is consecutively eligible set in the sense that if $\overline{E_j}=\{i_1,\cdots,i_r\}$, then the machines can be ordered so that $i_{k+1}=i_k+1, k=1,\cdots,r-1$. An important job is likely to have the first machine i_1 closer to the best capable machine 1 and wider range of eligible machines. The objective of our problem is to find a schedule with minimum *makespan,* in which each job is assigned to a machine in its consecutively eligible set. Machine scheduling problems are often classified using the three-field representation $\alpha|\beta|\gamma$ of Graham *et al*. (1979) where α describes machine environment, β sets restrictions on job processing and γ defines objective function. Then, we can denote our problem by $P|\overline{E_j}|C_{\max}$ using the representation.

In order to better understand our problem, we consider more general problem than ours, called *parallel machine scheduling problem with eligibility* denoted by $P|E_j|C_{\max}$. In this problem, the eligible set $E_j\subseteq M$ need not be consecutive. We for-

† Corresponding author : Professor Gyutai Kim, Department of Industrial Engineering, Chosun University, 375 Seosuk-Dong, Dong-Gu, Gwangju, Korea, 501-759, Fax : +82-62-230-7128, e-mail : gtkim@chosun.ac.kr

Received September 2002; revision received May 2003, June 2003; accepted June 2003.

mulate the problem $P|E_j|C_{\max}$ as the following integer linear program:

Minimize C
Subject to

$$\sum_{i \in E_j} x_{ij} = 1 \quad \text{for} \quad j = 1, \cdots, n$$

$$\sum_{j: i \in E_j} p_j x_{ij} \le C \quad \text{for} \quad i = 1, \cdots, m$$

$$x_{ij} \in \{0, 1\}.$$

The problem $P|E_j|C_{\max}$ can be applied to the health care industry. The processing flexibility of each operating room depends on the equipments it has. Hospitals want to fit in the rooms as many cases (patients) as possible, while considering the type of facilities needed for each case (Vairaktarakis and Cai, 2001). Thus, each case cannot be handled in all the operating rooms but some subset of them. Here, operating rooms represent the machines and the cases represent the jobs. Vairaktarakis and Cai developed a branch-and-bound procedure for the problem. In semiconductor manufacturing, machine capabilities in workcenters are different: machines in a wafer test workcenter cannot process all the jobs but pre-specified type of jobs (Centeno and Armacost, 1997). Sometimes, machine eligible set for each type of jobs is consecutive and thus the problem $P|\overline{E_j}|C_{\max}$ applies to in this case.

When every $E_j = M$ for all jobs j, it is the classical parallel machine scheduling problem denoted by $P||C_{\max}$. We note that $P||C_{\max}$ is a special case of our problem $P|\overline{E_j}|C_{\max}$. The classical parallel machine scheduling problem is NP-Complete (Garey and Johnson, 1979), so that it is unlikely that there exists efficient algorithms to find a schedule with optimum makespan. As a result, the focus of the theoretical research concerning parallel machine scheduling problem has been the analysis of the worst-case performances of various heuristic methods. Polynomial time algorithms that always produce schedules with makespan at most $(1+\varepsilon)$ times optimum are called $(1+\varepsilon)$-*approximation algorithms*. A family of algorithms $\{A_\varepsilon : \varepsilon > 0\}$ is called *polynomial time approximation scheme (PTAS)*, if each A_ε is $(1+\varepsilon)$-approximation algorithm. If each $(1+\varepsilon)$-algorithm in a family runs in polynomial of $1/\varepsilon$ as well as problem size, the family is called fully *polynomial time approximation scheme (FPTAS)*.

Consider first the problem $P||C_{\max}$ for $m=2$ or $P_2||C_{\max}$. The decision version of this problem is *partition* problem, which is NP-Complete in the ordinary sense. Thus, there exists exact (pseudo-polynomial time) algorithm using dynamic programming(Garey and Johnson, 1979). Furthermore, we can generate fast a schedule with makespan at most $(1+\varepsilon)$ times optimum by deploying FPTAS's for *knapsack* problems (Ibarra and Kim, 1975; Kellerer and Pferschy, 1999). Recently, $12/11$-approximation algorithm with running time of $O(n)$ was developed (He *et al*., 2000). Though there are many efficient algorithms for $P_2||C_{\max}$, it does not seem to be so in the case of $P||C_{\max}$ as the problem is proven to be NP-Complete in the strong sense. Hence, it does not exist any FPTAS for this problem unless $P=NP$. It was shown that LPT (*Longest Processing Time first)* and MULTIFIT algorithm yield schedules with makespan no more than $4/3-1/3m$ (Graham, 1969) and $13/11$ times optimum (Friesen, 1984; Yue, 1990), respectively. Hochbaum and Shmoys (1987, 1997) devised a PTAS for this problem though it runs not so fast in practice.

To the problem of $P|E_j|C_{\max}$, a greedy LS (*List Scheduling*) algorithm is applied and the performance ratio is proved to be $\log_2 2m$ (Azar *et al*., 1995). However, in general, it has been known that there cannot even exist PTAS for $P|E_j|C_{\max}$ unless $P=NP$ from the research of Lenstra *et al*. (1990). That is, they showed that no $(1+\varepsilon)$-approximation algorithm can exist for all $\varepsilon < 1/2$. Moreover, they presented an LP-based algorithm with worst-case performance ratio two on the most general case of unrelated parallel machine scheduling problem. Thus, using the algorithm we can generate schedule whose makespan is at most twice optimum for the problems $P|E_j|C_{\max}$ and $P|\overline{E_j}|C_{\max}$.

For the problem $P|\overline{E_j}|C_{\max}$, it is still an open question whether a PTAS exists or not. Though the LP-based algorithm is 2-approximate for $P|\overline{E_j}|C_{\max}$, its running time is not polynomial of the parameters of n and m but of the problem size. In other words, it is not a strongly polynomial time algorithm. Recently, Hwang(2002) presented strongly polynomial algorithm MFFH (*Multi First Fit preferred in Horizon*) similar to the binary search algorithm MULTIFIT (Coffman *et al*., 1978). However, the statement was not proved that MFFH always generates a schedule of twice optimum in Hwang (2002). In addition, it has been unknown until now whether the worst-case performance ratio of MFFH is strictly less than two.

The primary contribution of this paper is (1) to present the proof that the performance ratio of MFFH is at most two and (2) to show that MFFH cannot yield a schedule with makespan strictly less than twice optimum in worst-case by providing an example. As a result, we show that the performance ratio of MFFH is two.

We present some definitions in Section 2. In Section 3, the subfunction FFH *(First Fit preferred in Horizon*) of MFFH is described. Then, in Section 4, we describe MFFH and prove its worst-case performance. The time complexity MFFH is presented in Section 5. Finally, conclusions are followed.

2. Definitions

An instance of our scheduling problem is denoted by (J, m). Let μ_j and ν_j denote the first and last eligible machine in the consecutively eligible machine set. That is, when $\overline{E_j}=\{i_1, \cdots, i_r\}$, we let $\mu_j=i_1$ and $\nu_j=i_r$. We simply call the set $\overline{E_j}$ *eligible horizon* and just denote it as an interval $[\mu_j, \nu_j]$. For two jobs j and k, job j is said to be *preferable* to job k if $\nu_j \le \nu_k$. Then a schedule for an instance (J, m) is a partition of J into m disjoint sets, $S=\langle S_1, \cdots, S_m\rangle$ such that $j \in S_i$ if machine i belongs to the eligible horizon of job j, that is, $\mu_j \le i \le \nu_j$. The makespan of a schedule S, denoted by $z(S)$, is defined as $\max_{1\le i\le m} t(S_i)$, where for any set of $J' \subseteq J$, $t(J')=\sum_{j\in J'} p_j$ if $J' \neq \emptyset$, 0, otherwise. The optimum makespan for (J, m) is denoted by $z^*(J, m)=\min z(S)$, where the minimization is over all the schedules S. Then the *performance ratio* for an algorithm A is defined by

$$R(A)=\sup\left\{\frac{z_A(S)}{z^*(J,m)}: S \text{ is a schedule by A for all instances } (J,m)\right\}.$$

Any algorithm with performance ratio no greater than r is called r-approximation algorithm.

3. The algorithm FFH

The algorithm MFFH (Multi FFH) is a binary search method which uses internal subfunction FFH (First Fit preferred in Horizon). Firstly, we describe the function FFH. Given pre-specified time deadline C, FFH checks if it is possible to assign all the jobs within the time line C. FFH takes the first job in a list of jobs which is in the order of preference and then assigns the job on a machine with the smallest possible index, which is in the eligible horizon of the job, and can process the job within time C. Function FFH returns value *true* with a schedule if all the jobs were assigned within the specified time, *false,* otherwise. Now we present the algorithm FFH in <Figure 1>.

```
Boolean function FFH( J, m, C)
begin
    Sort jobs of J in the order of preference
    so that
    ν1 ≤ , ⋯ , ≤ νn;
    for i := 1 to m do Si := ∅;
    FFH := true; i := μ1; j := 1;
    repeat
        if t(Si) + pj ≤ C then begin
            Si := Si ∪ {j};
            j := j + 1;
            i := μj;
        end
        else i := i + 1;
    until ( j > n or i > νj)
    if i > νj then FFH : = false;
end
```

Figure 1. Algorithm FFH.

Without loss of generality, from now on we assume $\nu_j \le \nu_{j+1}$ for all $1 \le j < n$. We denote $S^j=\langle S_1^j, \cdots, S_m^j\rangle$ as the *partial* FFH schedule which has been constructed at the time right before job j is tried to be assigned. The following lemma implies the algorithm FFH has monotonicity property for binary search and assures a schedule with some bounded makespan.

Lemma 1 Suppose that for each job j in J, its processing time is not greater than $\rho z^*(J, m)$, $0 < \rho \le 1$. Then for all $C \ge (1+\rho) z^*(J, m)$, $FFH(J, m, C)$=*true*.

Proof. Assume the lemma is not true and (J, m) is a counterexample in which $FFH(J, m, C)$=*false*, for some $C \ge (1+\rho) z^*(J, m)$. Let j be the first job which could not be assigned within time C. That is, FFH terminated with *false* value with the partial FFH schedule S^j. Then, we obtain

$t(S_i^j)+p_j > C \geq (1+\rho)z^*(J,m)$ and thus

$$t(S_i^j) > z^*(J,m) \quad \text{for} \quad \mu_j \leq i \leq \nu_j, \tag{1}$$

since $p_j \leq \rho z^*(J,m)$. Note that there exists at least one machine i such that $t(S_i^j) \leq z^*(J,m), 1 \leq i \leq \nu_j$, since the sum of all the processing times of the jobs, whose eligible machines are within $[1, \nu_j]$, is not greater than $\nu_j z^*(J,m)$. Let machine r be the last machine such that $t(S_i^j) \leq z^*(J,m), 1 \leq i \leq r$ and $t(S_i^j) > z^*(J,m)$ for all $r < i \leq \nu_j$. Such r exists by (1). From the fact that $t(S_r^j) \leq z^*(J,m)$, we know all the jobs assigned to machines $i > r$ will not be eligible to machines from 1 to r. It is because if there is a job k with $\mu_k \leq r$ and it is assigned to machines $i > r$, then it means

$$t(S_r^j)+p_k \geq t(S_r^k)+p_k > C \geq (1+\rho)z^*(J,m)$$

and thus $p_k > \rho z^*(J,m)$, which is a contradiction to the assumption. Hence, all the jobs in $\bigcup_{i=r+1}^{\nu_j} S_i^j$ and j must only be eligible to machines from $r+1$ to ν_j. However, from the fact that

$$\sum_{i=r+1}^{\nu_j} t(S_i^j)+p_j > (\nu_j - r)z^*(J,m)+p_j,$$

we see it is impossible for these jobs to be assigned on the $\nu_j - r$ machines within time $z^*(J,m)$. This is a contradiction.

4. Algorithm MFFH and its Performance

The algorithm FFH will be used as the underlying subfunction of the binary search method MFFH for which we want to prove that it always generates a schedule with makespan no greater than $(1+\rho)$ times optimal when each job has processing time at most $\rho z^*(J,m)$, for $0 < \rho \leq 1$. MFFH starts with predetermined lower and upper bound for the estimated makespan of a problem instance. MFFH utilizes the results on LS (List Scheduling) schedule to obtain the needed lower and upper bound, $CL(J,m)$ and $CU(J,m)$, respectively. We know from the the study of Azar *et al.* (1995) that the LS algorithm can do no worse than $\log_2 2m$ times the optimum. We thus first apply the LS algorithm and then constructs:

$$CL(J,m) = \max\left\{\sum_j p_j/m, \frac{z_{LS}}{\log_2 2m}\right\},$$

$$CU(J,m) = z_{LS},$$

where z_{LS} is the makespan of a schedule by LS. Note that $CL(J,m) \leq z^*(J,m)$.

The MFFH with k binary search iteration is denoted by MFFH[k] and can be described as shown in <Figure 2>.

```
Procedure MFFH [k](J, m)
begin
    CL := CL(J, m);
    CU := CU(J, m);
    for i := 1 to k do begin
        C := (CL + CU)/2;
        if FFH (J, m, C)=true
            then CU := C;
        else CL := C;
    end
end
```

Figure 2. Algorithm MFFH

The MFFH schedule is defined to be the latest feasible schedule generated by the FFH with returned value *true* during the k iterations. If all the k operations of FFH resulted in failure, we set the one generated by LS as the MFFH schedule.

In the execution of the algorithm MFFH[k], the size of the interval $[CL, CU]$ is reduced by half at each iteration. It is easy to see that the gap between $CL(J,m)$ and $CU(J,m)$ is at most $\log_2 2m \cdot z^*(J,m)$. Then this together with Lemma 1 implies the following theorem, which can be proved in a similar way to that by Coffman *et al.* (1978). First, let CL_k and CU_k be the final lower and upper bound value, respectively, made right after the kth calling of the algorithm FFH. Then from the nature of binary search of MFFH, we obtain that

$$\begin{aligned} CU_k - CL_k &\leq 2^{-k}(CU(J,m) - CL(J,m)) \\ &\leq 2^{-k} \cdot \log_2 2m \cdot z^*(J,m). \end{aligned} \tag{2}$$

Theorem 2 For parallel machine scheduling problem with consecutive eligibility, if all the processing times are not greater than ρ times optimum ($0 < \rho \leq 1$), then we have $R(MFFP[k]) \leq 1 + \rho + 2^{-k} \cdot \log_2 2m$.

Proof. Suppose the theorem does not hold and there exists a problem instance (J, m) for which

$$\frac{z_{MFFH[k]}}{z^*(J,m)} > 1+\rho+2^{-k}\cdot\log_2 2m \qquad (3)$$

where $z_{MFFH[k]}$ is the makespan of the final schedule of MFFH with k iterations. We further suppose $FFH(J, m, CU_k)=false$. Then, from the operation of MFFH, all the k operations of FFH were in failure. Therefore, in this case we have $z_{MFFH[k]}=z_{LS}$, where z_{LS} is the makespan of the schedule by LS. Since $FFH(J, m, CU(J, m))=false$, by Lemma 1, we obtain that $CU(J, m)<(1+\rho)z^*(J, m)$. However, this is a contradiction to (3), noting $CU(J, m)=z_{LS}=z_{MFFH[k]}$. Hence, we have $FFH(J, m, CU_k)=true$ and thus $CU_k\geq z_{MFFH[k]}$, which means

$$CU_k > (1+\rho+2^{-k}\cdot\log_2 2m)z^*(J, m).$$

Then, from this with (2), it is true that

$$CL_k > (1+\rho)z^*(J, m) \geq CL(J, m), \qquad (4)$$

since $CL(J, m)\leq z^*(J, m)$. From this we know that FFH must have been executed with time deadline CL_k at some point during the binary search with the result $FFH(J, m, CL_k)=false$. However, this is impossible by Lemma 1 and (4).

As an immediate result of Theorem 2, we have the following corollary.

Corollary 3 For parallel machine scheduling problem with consecutive eligibility,

$$R(MFFP[k])\leq 2+2^{-k}\cdot\log_2 2m.$$

Proof. From the fact that each job has processing time no larger than the optimal makespan, we prove the corollary.

From Corollary 3, we see that the MFFH is 2-approximation algorithm. By choosing k large enough, a schedule with makespan at most twice optimum can be generated. In the case that the chosen k is so large that the term $2^{-k}\cdot\log_2 2m$ is almost zero, we just write that $R(MFFP)\leq 2$.

Then, a natural question is what the exact value of $R(MFFP)$ is, that is, whether $R(MFFP)$ is two or strictly less than two. In general, when one wants to prove that the performance ratio of an algorithm is γ, it is enough to show that $R(A)\leq\gamma$ and then, for every $r<\gamma$, to find instance (J, m) such that A cannot generate a schedule with makespan at most $rz^*(J, m)$. We note that MFFP cannot generate a schedule with makespan r times optimum for an instance (J, m), if $FFH(J, m, rz^*(J, m))=false$ for any r, $0<r<2$. Thus, in order to show that $R(MFFP)=2$, we need to get instances (J, m) such that $FFH(J, m, rz^*(J, m))=false$ for every $r<2$.

In the following example, we will see that for every r, $0<r<2$, there exists worst-case instances for which FFH cannot assign all the jobs within r times optimum.

Example 1. Consider a problem with (J, m). In J, there are $2m-2$ jobs: $m-2$ small jobs and m big jobs. Each big job j, $2\leq j\leq m-1$, has processing time $1-\varepsilon$ and eligible horizon of $[j, j]$, where $\varepsilon=\frac{1}{m-2}$. The big jobs 1 and m have processing time 1 with eligible horizons of $[1, m]$ and $[m, m]$, respectively. Finally, the small jobs $m+1,\cdots,2m-2$ have processing time ε and eligible horizon of $[1, m-1]$.

Then as shown in <Figure 3> (b), the optimal schedule S^* with makespan $z^*(J, m)=1$ would be given as follows:

$$S_1^*=\{1\},$$
$$S_i^*=\{i, m+i-1\}, \quad 2\leq i\leq m-1,$$
$$S_m^*=\{m\}.$$

Now, consider the schedule generated by FFH with time deadline $2-\varepsilon$. Since the $m-2$ big jobs are preferable to the remaining jobs, these will first be assigned to their corresponding machines. Next, all the $m-2$ small jobs will be assigned to machine 1. Then, the job m is assigned to machine m.

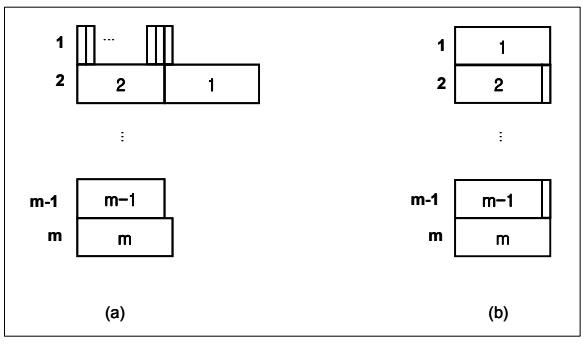

Figure 3. (a) FFH and (b) optimal schedule.

At this time, the machines 1 and m have load of 1 and the remaining $m-2$ machines have load of $1-\varepsilon$. Thus, finally, the first big job 1 is

assigned to machine 2, resulting to its completion time $2-\varepsilon$. Hence, the makespan by FFH is $2-\varepsilon$ (<Figure 3> (a)), which is $2-\varepsilon$ times optimum makespan. Note that for any $r<2-\varepsilon$, FFH cannot assign all the jobs within time r, i.e., $FFH(J, m, r)=false$. Thus, we can see that for any r, $0<r<2$, there exists (J, m) such that $FFH(J, m, rz^*(J, m))=false$ (by choosing m such that $2-\varepsilon=2-\frac{1}{m-2}>r$). Therefore, the exact performance ratio of MFFP is two.

5. Complexity of MFFH

In general, for parallel machine scheduling problem with consecutive eligibility, the performance ratio of MFFH is the same as that of the LP-based algorithm (Lenstra *et al*., 1990). However, in execution time, it might be said that MFFH is much faster than the LP-based algorithm.

Let us examine how long MFFH takes to run. The sorting step of FFH can actually be implemented once at the time MFFH starts, and hence we first note that each application of FFH only takes $O(n\log m)$ steps (Coffman *et al*., 1978). In addition, the algorithm LS runs in time $O(n\log m)$. Thus the total time for $MFFH[k]$, including the initial sorting of J by preference, is $O(n\log n+kn\log m)$. Therefore, MFFH is a strongly polynomial time algorithm (polynomial in the parameters of n and m) while the LP-based algorithm is polynomial in the problem size, which means in general MFFH runs faster than the LP-based algorithm.

6. Conclusions

We considered the problem of scheduling n jobs to m parallel machines where each job cannot be eligible to all the machines but some subset of consecutive machines ($P|\overline{E_j}|C_{\max}$). For this problem, we developed a binary search algorithm MFFH, which invokes its subfunction FFH. FFH is a kind of FF (First Fit) algorithm for bin-packing problems. It assigns jobs to the first possible machine that can process it within given time deadline. We showed that MFFH is a 2-approximation algorithm running in polynomial time of n and m. Thus, it is likely that MFFH executes much faster than the known LP-based algorithm. Though MFFH is a strongly polynomial time algorithm, its performance ratio is still not good. More effective algorithm, if it exists, might consider the information of processing times. It is shown that there are no algorithms with performance ratio less than *1.5* for the parallel machine scheduling problem with general eligibility ($P|E_j|C_{\max}$). The hardness of our problem is still open question.

Acknowledgements

The authors would like to thank the referees for their careful reading and advice.

References

Azar Y., Naor J., Rom R. (1995), The competitiveness of On-Line Assignments, *J. Algorithms*, **18**, 221-237.

Centeno G., Armacost R.L. (1997), Parallel machine scheduling with release time and machine eligibility restrictions, *Computers and Industrial Engineering,* **33**, 273-276.

Coffman Jr E.G., Garey M.R., Johnson D.S. (1978), An application of bin-packing to multiprocessor scheduling, *SIAM J. Comput.*, **7**, 1-17.

Friesen D.K. (1984), Tighter bounds for the multi processor scheduling algorithm. *SIAM J. Comput*., **13**, 35-59.

Garey M.R., Johnson D.S. (1979), *Computers and Intractability: A Guide to the theory of NP-Completeness*, Freeman, San Francisco.

Graham R.L. (1969), Bounds on multiprocessor timing anomalies, *SIAM J. Appl. Math.*, **17**, 263-269.

Graham, R.L., Lawler E.L., Lenstra J.K. and Rinnooy Kan A.H.G. (1979), Optimization and Approximation in Deterministic Machine Scheduling: A Survey. *Annals of Discrete Mathematics,* **5**, 287-326.

He Y., Kellerer H., Kotov V. (2000), Linear Compound Algorithms for the Partitioning Problem, *Naval Research Logistics,* **47**, 593-601.

Hochbaum D.S., Shmoys D. (1987), Using dual approximation algorithms for scheduling problems: Theoretical and practical results, *J. ACM,* **34**, 144-162.

Hochbaum D.S. (1997), *Approximation Algorithms for NP-Hard Problems,* PWS PUBLISHING COMPANY, Boston, 370-371.

Hwang H.-C. (2002), A strongly polynomial algorithm for parallel machine scheduling problem with consecutive eligibility, *The Research Institute of Industrial Technology Chosun University,* **24**, 27-31.

Ibarra O.H., Kim C.E. (1975), Fast approximation algorithms for the knapsack and sum of subset problems, *J. ACM,* **22**, 463-468.

Kellerer H., Pferschy U. (1999), A New Fully Polynomial Approximation Scheme for the Knapsack Problem, *Journal of Combinatorial Optimization,* **3**, 59-71.

Lenstra J.K., Shmoys D.B., Tardos E. (1990), Approximation Algorithms for Scheduling Unrelated Parallel Machines, *Mathematical Programming,* **46**, 259-271.

Yue M. (1990), On the exact upper bound for the MULTIFIT processor scheduling algorithm, in *Operations Research in China,* M. Yue (ed.), Vol. 24 of Annals of Operations Research, Baltzer, Basel, Switzerland, 233-259.

Vairaktarakis G.L., X. Cai (2001), The Value of Processing Flexibility in Multipurpose Machines, Technical Memorandum Number 744, Weatherhead School of Management, Case Western Reserve University.