ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1996
Seria: AUTOMATYKA z.117 Nr kol. 1337

Czesław SMUTNICKI
Politechnika Wrocławska

WORST-CASE PERFORMANCE EVALUATIONS FOR SOME SCHEDULING ALGORITHMS

Summary. A considerable attention has been passed recently to the flow shop problem with the mean flow time criterion, chiefly due to its industrial applications. Most of the known approximation algorithms recommended for this problem have the worst-case performance ratio equal the number of jobs n or the number of machines m. In this paper we propose an algorithm with this ratio equal $\lceil m/k \rceil \rho_k$, where ρ_k is the worst-case performance ratio of an algorithm which solves an auxiliary k-machine problem.

OSZACOWANIA NAJGORSZEGO PRZYPADKU DLA PEWNYCH ALGORYTMÓW SZEREGOWANIA

Streszczenie. Znaczną uwagę zwrócono ostatnio, głównie ze względu na liczne zastosowania przemysłowe, na przepływowy problem szeregowania z kryterium minimalizacji średniego czasu przepływu. Większość znanych algorytmów aproksymacyjnych dla tego problemu posiada współczynnnik najgorszego przypadku równy liczbie zadań n lub liczbie maszyn m. W pracy proponujemy nowy algorytm o współczynniku równym $\lceil m/k \rceil \rho_k$, gdzie ρ_k jest współczynnikiem najgorszego przypadku pewnego algorytmu rozwiązującego pomocniczy problem k-maszynowy.

1. Introduction

A considerable attention has been passed recently to the flow shop problem with the mean flow time criterion, chiefly due to its industrial applications. Since the problem is NP-hard for two and more than two machines, a lot of approximation algorithms have been developed to provide a good solution in a quick time, see the bibliography.

Traditionally, approximation algorithms are ranked according to the running time (or computational complexity) and the distance from a generated solution to the optimal solution (an algorithm performance). Several measures of the algorithm performance have been introduced. These measures can be investigated either experimentally (computer tests on random instances) or analytically (worst-case analysis, probabilistic analysis). Experimental analysis is the most popular, easy to perform, however subjective method of evaluation of an algorithm performance since results depend on a chosen sample of instances. Alternatively, the worst-case and/or probabilistic analyses yield an objective, instance independent evaluation of an algorithm performance. One can say that these analyses provide another, more suitable, characteristics of the algorithm behavior.

The paper deals with the permutation flow-shop problem formulated as follows. The set of n different jobs should be processed on m different machines. Each job j, $j \in J = \{1, \ldots, n\}$ passes through the machines $1, 2, \ldots, m$ in that order and requires uninterrupted time p_{ij} for processing on machine i, $i \in M = \{1, .., m\}$. Machine i, $i \in M$, can execute at most one job at a time and each machine processes the jobs is the same order. We wish to find the optimal job processing order, represented by a permutation π on the job set J, which minimizes the mean flow time

$$F(\pi) = \frac{1}{n} \sum_{j=1}^{n} C_{m\pi(j)}, \tag{1}$$

where

$$C_{i\pi(j)} = \max_{1 \leq j_0 \leq j_1 \leq \ldots \leq j_{i-1} \leq j_i \leq j} \sum_{s=1}^{i} \sum_{t=j_{s-1}}^{j_s} p_{s\pi(t)} \tag{2}$$

is the completion time of job $\pi(j)$ on machine i, $j = 1, \ldots, n$, $i \in M$.

The data n, m, $(p_{ij}, i \in M, j \in J)$, specify an instance Z of the problem. Denote by Π the set of all permutations on J, by $F(\pi; Z)$ the mean flow time for the job processing order π and instance Z. The processing order $\pi^* \in \Pi$ such that $F(\pi^*; Z) = \min_{\pi \in \Pi} F(\pi; Z)$ is called the optimal processing order. Let $\pi^A \in \Pi$ denote a permutation generated by an algorithm A. The worst-case performance ratio of the algorithm A is defined as $\eta^A = \min\{y : F(\pi^A; Z)/F(\pi^*; Z) \leq y,$ for each instance $Z\}$. In the sequel, the argument Z will be omitted in $F(\pi; Z)$ if it is not necessary.

We will use a simplified notation analysing Z. If the set J consists of $o \geq 1$ subsets $J_1, J_2, \ldots, J_o$ of identical jobs we assume that each job from a subset J_j is indexed by the same index j, $j = 1, 2, \ldots, o$. In such case the job processing order will be considered as a permutation with repetitions. Let π_I denote a permutation on a set $I \subset \{1, 2, \ldots, o\}$. The symbol (π_I), will denote the $s-$times concatenation of permutation π_I.

2. Existing algorithms

In this section we provide some new worst-case evaluations for few algorithms known from the literature. The complete list of currently known evaluations is given in Table 1.

Among *constructive approximation algorithms* provided for the stated problem one can find a group of methods that refer to *job waiting times*, *between jobs delays*, *gaps*, *jobs matching*, or *jobs fitness*, [9, 6, 20]. Such algorithms were designed for problems with the makespan criterion as well as to those with the total (weighted) sum of completion times. Five algorithms proposed in [9] also belong to this class. There are based on a common scheme and create the permutation by adding step-by-step a new job to the end of existed chain of jobs. Thus, the algorithm operates on sets of scheduled (S) and unscheduled yet jobs ($U = J\backslash S$). Jobs from the set S form a partial permutation σ, and let $C_{i\sigma(d)}$ denote the completion time of the last job from S on the machine i, $i \in M$, $d = |S|$. Let $j \in U$ be a job added to the end of σ, i.e. the new partial permutation is σj. Then the earliest completion times of job j are equal $C_{1j} = C_{1\sigma(d)} + p_{1j}$, $C_{ij} = \max\{C_{i\sigma(d)}, C_{i-1,j}\} + p_{ij}$, $i = 2, \ldots, m$. The latest completion times of job j are defined as follows $D_{ij} = C_{mj} - \sum_{s=i+1}^{m} p_{sj}$, $i \in M$. If $C_{i\sigma(d)} < D_{ij} - p_{ij}$ then we say that there is a *delay* between jobs $\sigma(d)$ and j on machine i. Since the starting time of job j on machine i can be chosen from the interval $[C_{ij} - p_{ij}, D_{ij} - p_{ij}]$, then if $C_{ij} < D_{ij}$ some of these delays can be reduced. To this order define for job j two indices: a_j is the smallest index i, $1 \le i \le m$ such that $C_{i\sigma(d)} = D_{ij} - p_{ij}$, whereas b_j is the highest index i satysfying this equality. Clearly $a_j \le b_j$. The value $\sum_{s=1}^{a_j-1}(D_{sj} - p_{sj} - C_{s\sigma(d)})$ is called the *total back delay*, which is reducible to the value $\sum_{s=1}^{a_j-1}(C_{sj} - p_{sj} - C_{s\sigma(d)})$ by appropriate selection of starting times of j on machines $1, \ldots, a_j - 1$. The value $\sum_{s=b_j+1}^{m}(C_{sj} - p_{sj} - C_{s\sigma(d)})$ is the *total front delay*, which however cannot be reduced.

A job k to follow by σ has been selected among those from U in the following manner. First, for each job $j \in U$ a priority value is calculated by using completion times of jobs j in the sequence σj. Next job $k \in U$ with the least priority value is selected and σk becomes the new sequence for the next iteration. The authors of [9] have proposed five priority rules, called hereafter as algorithms $KS1, \ldots, KS5$:

$KS1$: (total front delay) $\omega_j^{KS1} = \sum_{s=b_j+1}^{m}(C_{sj} - p_{sj} - C_{s\sigma(d)})$,

$KS2$: (total between jobs delay) $\omega_j^{KS2} = \sum_{s=1}^{m}(C_{sj} - p_{sj} - C_{s\sigma(d)})$,

$KS3$: (total back delay) $\omega_j^{KS3} = \sum_{s=1}^{a_j-1}(D_{sj} - p_{sj} - C_{s\sigma(d)})$,

$KS4$: (total weighted betweeen jobs delay) $\omega_k^{KS4} = \sum_{i=2}^{m} i\,(C_{sj} - p_{sj} - C_{s\sigma(d)})$,

$KS5$: (maximum left shift savings) $\omega_k^{KS5} = -\sum_{s=1}^{m}(D_{sj} - C_{sj})$

To ensure better solution performance, each job is considered in turn as the first job in the primal partial sequence, i.e. the algorithm is repeated n times, each time starting from $\sigma = (j)$ for succesive $j = 1, \dots, n$. Algorithms $KS1, \dots, KS5$ have the computational complexity $O(n^3m)$. Note that algorithm RC', [20], is a special case of $KS2$ since ω^{KS2} can be also written as $\sum_{i=2}^{m} \max\{C_{i-1,k} - C_{i\sigma(d)}, 0\}$.

We begin analysis of algorithms $KS1, \dots, KS5$ with an example.

Example 1. Let $m \geq 2$. The job set consists of two subsets J_1 and J_2 with cardinalities 1 and $n-1$, respectively. The processing times are equal to $p_{m1} = 1, p_{m-1,1} = p_{m-1,2} = p_{m2} = \epsilon$, where ϵ is the sufficiently small number such that $\lim_{\epsilon \to 0} \epsilon n^2 = 0$. All the remaining (undefined above) processing times are equal to zero. ◇

Starting from $\sigma = (1)$ we have to generate job processing order $\pi'^A = (1)_1(2)_{n-1}$, $A \in \{KS1, \dots, KS5\}$ since there are no other alternatives. In turn, starting from $\sigma = (2)$ we can generate the job processing order $\pi^A = (2)_1(1)_1(2)_{n-2}$, $A \in \{KS1, \dots, KS5\}$. Indeed, if $\sigma = (2)$ we can schedule next the job from J_1 or a job from J_2. In both cases delays between σ and newly scheduled job are zero, so the partial processing order for the next iteration can be $\sigma = (2)_1(1)_1$, and this result does not depend on the selected priority values $KS1, \dots, KS5$. Continuing, there are no other choices. Conseqently, $F(\pi'^A) = \frac{1}{n}[n + \frac{1}{2}\epsilon n(n+1)]$ and $F(\pi^A) = \frac{1}{n}[2\epsilon + \frac{1}{2}\epsilon(n-2)(n-1) + (n-1)(1+2\epsilon)]$. It can be verified that the permutation $\pi^* = (2)_{n-1}(1)_1$ is optimal and $F(\pi^*) = \frac{1}{n}[1 + \frac{1}{2}\epsilon n(n-1) + 2n\epsilon - \epsilon)]$. Clearly $F(\pi^A) < F(\pi'^A)$. Finally $F(\pi^A)/F(\pi^*)$ tends to $n-1$ if $\epsilon \to 0$, $A \in KS1, \dots, KS5$. ∎

In [3] it has been shown that the algorithm producing random permutation has the worst-case performance ratio equal to n. In the above context, the precise proof yielding value of η^A, $KS1, \dots, KS5$ seems to be of little importance and will be omitted.

3. New algorithms

In this section we propose a family of algorithms, called Tk, which are based on an approximation of m-machine problem by some k-machine flow shop problem for $k \leq m$. Let $m_1, \ldots, m_k$ be a sequence of integers such that $m_s \geq 1$, $s = 1, \ldots, k$, $\sum_{s=1}^{k} m_s = m$, for some $k \leq m$. We define $l_i = \sum_{s=1}^{i} m_s$, $i = 1, \ldots, k$, and $l_0 = 0$. The auxiliary k-machine flow shop problem has processing times defined as follows

$$q_{ij} = \sum_{s=l_{i-1}+1}^{l_i} p_{sj}. \tag{3}$$

Athough the latter problem is univocally defined by the sequence $(m_1, \ldots, m_k)$, however for the sake of notation simplicity we will identify it by k. Since this problem for $k > 1$ is still NP-hard, we assume that it is solved by an approximation algorithm with the worst-case performance ratio ρ_k. Clearly, the proposed algorithm Tk generates a permutation π^{Tk} by solving, with the worst-case bound ρ_k, the k-machine flow shop problem with processing times defined by (3). We will show that there exists in this family an algorithm with the worst-case performance ratio less than $\lceil m/2 \rceil$. Let us denote this k-machine instance by Z^k, and appropriate job completion times by $C_{ij}(Z)$ and $C_{ij}(Z^k)$. We start the analysis from some auxiliary properties.

Property 1. For any π, any Z, any $j \in N$, and any $(m_1, \ldots, m_k)$, we have

$$C_{m\pi(j)}(Z) \leq C_{k\pi(j)}(Z^k). \ \square \tag{4}$$

Proof. Let $1 \leq u_0 \leq u_1 \leq \ldots \leq u_m \leq j$ be the sequence of integers minimizing the right-hand side of formula (2) for $i = m$. Then, we obtain

$$C_{m\pi(j)}(Z) = \sum_{s=1}^{m} \sum_{t=u_{s-1}}^{u_s} p_{s\pi(t)} = \sum_{r=1}^{k} \sum_{s=l_{r-1}+1}^{l_r} \sum_{t=u_{s-1}}^{u_s} p_{s\pi(t)} \leq \sum_{r=1}^{k} \sum_{s=l_{r-1}+1}^{l_r} \sum_{t=u_{l_{r-1}}}^{u_{l_r}} p_{s\pi(t)}$$

$$= \sum_{r=1}^{k} \sum_{t=u_{l_{r-1}}}^{u_{l_r}} \sum_{s=l_{r-1}+1}^{l_r} p_{s\pi(t)} \leq \max_{1 \leq v_0 \leq \ldots \leq v_r \leq j} \sum_{r=1}^{k} \sum_{t=v_{r-1}}^{v_r} \sum_{s=l_{r-1}+1}^{l_r} p_{s\pi(t)}$$

$$= \max_{1 \leq v_0 \leq \ldots \leq v_r \leq j} \sum_{r=1}^{k} \sum_{t=v_{r-1}}^{v_r} q_{r\pi(t)} = C_{r\pi(j)}(Z^k),$$

which completes the proof. ■

Property 2. For any π, any Z, any $j \in N$, and any $(m_1, \ldots, m_k)$, we have

$$C_{r\pi(j)}(Z^k) \leq m_{max} C_{m\pi(j)}(Z), \tag{5}$$

where

$$m_{max} = \max_{1 \leq i \leq k} m_i. \quad \square \tag{6}$$

Proof. Employing (2) for $i = m$ we obtain the following sequence of inequalities

$$m_{max}\, C_{m\pi(j)}(Z) = \sum_{i=1}^{m_{max}} \max_{1 \leq j_0 \leq j_1 \leq \ldots \leq j_m \leq j} \sum_{s=1}^{m} \sum_{t=j_{s-1}}^{j_s} p_{s\pi(t)}$$

$$= \sum_{i=1}^{m_{max}} \max_{1 \leq j_0 \leq j_1 \leq \ldots \leq j_m \leq j} \left(\sum_{r=1}^{k} \sum_{s=l_{r-1}+1}^{l_r} \sum_{t=j_{s-1}}^{j_s} p_{s\pi(t)}\right) \geq \sum_{i=1}^{m_{max}} \max_{1 \leq j_0 \leq j_1 \leq \ldots \leq j_m \leq j} \left(\sum_{r=1}^{k} \sum_{t=j_{x_{ri}-1}}^{j_{x_{ri}}} p_{x_{ri}\pi(t)}\right)$$

where $x_{ri} = \min\{l_{r-1} + i, l_r\}$. Note that for any fixed i we have $x_{r-1,i} \leq x_{ri}$. Next note that for fixed i maximization can be done over indices $v_r = x_{ri}$, then we can continue as follows

$$= \sum_{i=1}^{m_{max}} \max_{1 \leq v_0 \leq v_1 \leq \ldots \leq v_k \leq j} \left(\sum_{r=1}^{k} \sum_{t=v_{r-1}}^{v_r} p_{x_{ri}\pi(t)}\right) \geq \max_{1 \leq v_0 \leq v_1 \leq \ldots \leq v_k \leq j} \sum_{r=1}^{k} \sum_{t=v_{r-1}}^{v_r} \left(\sum_{i=1}^{m_{max}} p_{x_{ri}\pi(t)}\right)$$

$$\geq \max_{1 \leq v_0 \leq v_1 \leq \ldots \leq v_k \leq j} \sum_{r=1}^{k} \sum_{t=v_{r-1}}^{v_r} \left(\sum_{i=1}^{m_r} p_{x_{ri}\pi(t)}\right) = \max_{1 \leq v_0 \leq v_1 \leq \ldots \leq v_k \leq j} \sum_{r=1}^{k} \sum_{t=v_{r-1}}^{v_r} \left(\sum_{i=l_{r-1}+1}^{l_r} p_{i\pi(t)}\right)$$

$$= \max_{1 \leq v_0 \leq v_1 \leq \ldots \leq v_k \leq j} \sum_{r=1}^{k} \sum_{t=v_{r-1}}^{v_r} q_{r\pi(t)} = C_{k\pi(j)}(Z^k).$$

The final inequality uses the well-known claim $\sum_{i \in A} \max_{j \in B} a_{ij} \geq \max_{j \in B} \sum_{i \in A} a_{ij}$. ∎

By using introduced properties we can derive evaluations on the worst-case performance ratio for algorithms Tk.

Theorem. For any algorithm Tk defined by the sequence $(m_1, \ldots, m_k)$ we have

$$\eta^{Tk} = m_{max}\, \rho_k. \quad \square \tag{7}$$

Proof. Applying (4) for $\pi = \pi^k$ in the definition (1), we obtain

$$F(\pi^k; Z) \leq F(\pi^k; Z^k).$$

Next, by the definition of ρ_k we have $F(\pi^k; Z^k)/F(\pi^{*k}; Z^k) \leq \rho_k$, where π^{*k} is the optimal permutation of the k-machine flow-shop problem. Next applying the formulae (5) for $\pi = \pi^*$ in the definition (1), we obtain

$$m_{max}\, F(\pi^*; Z) = m_{max} \frac{1}{n} \sum_{j=1}^{m} C_{m\pi^*(j)}(Z)$$

$$\geq \frac{1}{n}\sum_{j=1}^{n} C_{k\pi^*(j)}(Z^k) = F(\pi^*; Z^k) \geq F(\pi^{*k}; Z^k) \geq \frac{F(\pi^k; Z^k)}{\rho_k}.$$

Combining these results we get

$$F(\pi^k; Z) \leq m_{max}\, \rho_k F(\pi^*; Z)$$

which yields the upper bound on η^{Tk}. To complete the proof we will provide an example showing that this bound is tight.

Example 2. Let $m \geq 2$. Without losing generality we can assume that $m_{max} = m_1$. To simplify notation we set $c = m_{max}$. The job set J consists of c subsets $J_1, \ldots, J_c$ with cardinalities w each, where w is an integer. The processing times are equal $p_{ii} = 1, i = 1, 2, \ldots, c$. All the remaining (undefined above) processing times are equal zero. Thus, we have $n = cw$ jobs. ◇

For this instance we have $q_{1j} = \sum_{i=1}^{b} p_{ij} = 1$ and $q_{ij} = 0$, $i = 2, \ldots, k$, for all $j = 1, 2, \ldots, c$. Due to special structure, the auxilary problem can be solved to optimal using well-known SPT rule. Therefore $\rho^k = 1$ and algorithm Tk can generate the permutation $\pi^{Tk} = (1)_w(2)_w \ldots (c)_w$. Consequently $nF(\pi^{Tk}) = (1/2)[w^2c(c-1) + wc(w+1)]$. One can verify that the the optimal job processing order is $\pi^* = (c, c-1, \ldots 1)_w$ and $nF(\pi^*) = (1/2)cw(w+1)$. In consequence $F(\pi^{Tk})/F(\pi^*) = (c-1)\frac{w}{w+1} + 1$ which tends to $c = m_{max}$ if $w \to \infty$. ∎

The Theorem provides the following surprising theoretical result.

Corollary. There exists algorithm Tk such that

$$\eta^{Tk} \leq \lceil m/k \rceil \rho_k. \; \square \tag{8}$$

For $k = 1$ we obtain well-known result $\eta^{T1} = m$, [3], since in this case $T1$ is equivalent to SPT and $\rho_1 = 1$. For $k = 2$ we get evaluation $\eta^{T2} = \lceil m/2 \rceil \rho_2$, that had been found previously for other algorithms, see Table 1. In practice, solving the two- or three-machine case is usually easier than general m-machine problem. Therefore $k = 2, 3$ can be recommended for applications. For example, assuming $m = 9$, $k = 3$, $m_1 = m_2 = m_3 = 3$, and $\rho_3 = 1$ (i.e. the auxiliary three-machine problem is solved to optimal), we obtain $\eta^{T3} = 3$, whereas the best known up to now result refers to $\eta^{T1} = 9$ or $\eta^{T2} = \lceil m/2 \rceil = 5$. It is clear that Algorithm Tk will provide better results if and only if there exists an approximation algorithm for the k-machine problem with the worst-case performance ratio sufficiently good. The existence of such the algorithm remains a problem in question.

BIBLIOGRAPHY

1. Campbell H.G., Dudek R.A., and Smith M.L., 1970, A heuristic algorithm for the n job m machine sequencing problem. Management Science 16, B630-637.
2. Dannenbring D.G., 1977, An evaluation of flow-shop sequencing heuristics. Management Science 23, 1174-1182.
3. Gonzales T., Sahni S., 1978, Flowshop and Jobshop Schedules: Complexity and Approximation, Operations Research, 26, 36-52.
4. Gupta J.N.D., 1971, A functional heuristic algorithm for the flow-shop scheduling problem. Operational Research Quarterly 22, 39-47.
5. Ho J.C., 1995, Flowshop sequencing with mean flowtime objective, European Journal of Operational Research, 81, 571-578.
6. Ho J.C., and Chang Y.L., 1990, A new heuristic for the n-job, m-machine flow-shop problem. European Journal of Operational *Research* 52, 194-206.
7. Hoogeveen J.A., Kawaguchi T., 1995, Minimizing total completion time in a two-machine flowshop: analysis of special cases, Working paper. Eindhoven University of Technology.
8. Hundal T.S., and Rajgopal J., 1988, An extension of Palmers's heuristic for the flow-shop scheduling problem. International Journal of Production Research 26, 1119-1124.
9. King J.R. and Spachis A.S., 1980, Heuristics for Flowshop Scheduling, International Journal of Production Research, 19, 345-357.
10. Lai T.C., 1995, A Note on Heuristics of Flow-Shop Scheduling, Technical Report, National Taiwan University.
11. Miyazaki S., Nishiyama N., and Hashimoto F., 1978, An adjacent pairwise approach to the mean flow-time scheduling problem, Journal of the Operations Research Society of Japan, 21, 287-299.
12. Nawaz M., Enscore Jr. E.E., and Ham I., 1983, A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA International Journal of Management Science 11, 91-95.
13. Nowicki E., and Smutnicki C., 1989, Worst-case analysis of an approximation algorithm for flow-shop scheduling. Operations Research Letters 8, 171-177.

14. Nowicki E., and Smutnicki C., 1991, Worst-case analysis of Dannenbring's algorithm for flow-shop scheduling. Operations Research Letters 10, 473-480.

15. Nowicki E., and Smutnicki C., 1993, New results in the worst-case analysis for flow-shop scheduling. Discrete Applied Mathematics 46, 21-41.

16. Nowicki E., and Smutnicki C., 1994, A note on worst-case analysis of an aproximation algorithm for flow-shop scheduling, European Journal of Operational Research 74, 128-134.

17. Page E.S., 1961, An Approach to Scheduling Jobs on Machines, Journal of Royal Statistics Society, Ser.B, 23, 484-492.

18. Palmer D.S, 1965, Sequencing Jobs Through a Multi-Stage Process in the Minimum Total Time - A Quick Method of Obtaining Near Optimum. Operations Research Quarterly 16, 101-107.

19. Rajendran C., and Chaudhuri D., 1991, An efficient heuristic approach to the scheduling of jobs in a flow-shop, European Journal of Operational Research 61, 318-325.

20. Rajendran C., and Chaudhuri D., 1991, A flowshop scheduling algorithm to minimize total flowtime, Journal of the Operations Research Society of Japan, 34, 28-46.

21. Rock H., and Schmidt G., 1982, Machine Aggregation Heuristics in Shop Scheduling. Methods of Operations Research 45, 303-314.

22. Smutnicki C., 1995, Minimizing mean flow time in the permutation flow shop. A worst-case study, Technical Report PRE /95, (submitted to Operations Research Letters).

23. Smutnicki C., 1994, Some results of the worst-case analysis for flow-shop scheduling, Technical Report PRE /95, (submitted to European Journal of Operational Research).

24. Smutnicki C., 1995, Results of the worst-case analysis for some scheduling problem, Proceedings of 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation, 105-115.

Recenzent: Dr hab. inż. Eugeniusz Toczyłowski, prof. Pol.Warsz.

Wpłynęło do Redakcji do 30.06.1996 r.

Abstract

The paper deals with NP-hard flow shop scheduling problem with the mean flow time criterion which has received recently a considerable attention due to industrial applications. Most of the known approximation algorithms recommended for this problem have

the worst-case performance ratio equal the number of jobs n or the number of machines m. In this paper we propose an algorithm with this ratio equal $\lceil m/k \rceil \rho_k$, where ρ_k is the worst-case performance ratio of an algorithm which solves the auxiliary k-machine problem. A current state of art in the worst-case analysis for permutation flow-shop problems with various scheduling criteria has been also presented.

Tablica 1

Lower $\underline{\eta}^A$ and upper $\overline{\eta}^A$ bounds (provided in E) on the worst-case performance as ratio η^A of an algorithm A (developed in B) for various scheduling criteria K

A	B	K	$\underline{\eta}^A$	$\overline{\eta}^A$	E
GS	[3]	C_{max}	m	m	[3]
R	[21]	C_{max}	$\lceil m/2 \rceil$	$\lceil m/2 \rceil$	[21]
CDS	[1]	C_{max}	$\lceil m/2 \rceil$	$\lceil m/2 \rceil$	[13]
RA	[2]	C_{max}	$m/\sqrt{2}+O(1/m)$	$m/\sqrt{2}+O(1/m)$	[14]
$RACS, RAES$	[2]	C_{max}	$m/\sqrt{2}+O(1/m)$	$m/\sqrt{2}+O(1/m)$	[15]
P	[18]	C_{max}	$m/\sqrt{2}+O(1/m)$	$m/\sqrt{2}+O(1/m)$	[15]
NEH	[12]	C_{max}	$\sqrt{m/2}+O(1/m)$	$(m+1)/2$	[15]
HR	[8]	C_{max}	$m/\sqrt{2}+O(1/m)$	$m/\sqrt{2}+O(1/m)$	[16]
G	[4]	C_{max}	$m-1$	$m-1$	[16]
TG	[10]	C_{max}	$(m+1)/2$	$(m+1)/2$	[10]
IE, M	[17]	C_{max}	m	m	[10]
$KS1, KS2$	[9]	C_{max}	m	m	[10]
$CDS+HC$	[6]	C_{max}	$m/2$	$\lceil m/2 \rceil$	[23]
GS	[3]	F	n	n	[3]
SPT	[3]	F	m	m	[3]
RCo	[19]	F	$2m/3+1/3$	m	[22]
$RCo, m=2$	[19]	F	1.908	2	[22]
RC', RC''	[20]	F	n	n	[22]
RC'''	[20]	F	$2m/3+1/(3m)$	n	[22]
$RC''', m=2$	[20]	F	1.577	n	[22]
$CDS, CDS+HC$	[6]	F	n	n	[22]
$HK, m=2$	[7]	F	$2b/(a+b)$	$2b/(a+b)$	[7]
S^k	[22]	F	$(\lvert\frac{m}{2}-k\rvert+\frac{m}{2})\rho_2$	$(\lvert\frac{m}{2}-k\rvert+\frac{m}{2})\rho_2$	[22]
$KS1,\ldots,KS5$	[9]	F	$n-1$	n	
Tk		F	$\lceil m/k \rceil \rho_k$	$\lceil m/k \rceil \rho_k$	
GS	[3]	C	$1+(n-1)(\overline{w}/\underline{w})$	$1+(n-1)(\overline{w}/\underline{w})$	[23]
$CDS+HC, G+HC,$ $P+HC, RA+HC$	[6]	C	$1+(n-1)(\overline{w}/\underline{w})$	$1+(n-1)(\overline{w}/\underline{w})$	[23]
F	[23]	C	m	m	[23]
Q/X	[23]	C	$\lceil m/2 \rceil \rho_2$	$\lceil m/2 \rceil \rho_2$	[23]