MINIMIZING THE MEAN TARDINESS IN A N/1 SEQUENCING PROBLEM

F. Ghassemi Tari

Department of Industrial Engineering
Sharif University of Technology
Tehran, Iran

H. Fallah

Faculty of Engineering
Imam Hossein University
Tehran, Iran

Abstract This paper considers the problem of minimizing the mean tardiness of N jobs when the jobs are scheduled on a single machine. A simple algorithmic procedure is developed to obtain an optimal or a near optimal sequence for the N jobs while considering an equal penalty cost incurred to each job delivered later than its due date. The developed algorithm is applied to the several test problems.The results obtained reveals that the computational time and the required computer memory of the developed algotrithm to provide a good solution are very low.

چکیده یک مدل ریاضی جهت برنامه‌ریزی n کار مستقل روی یک ماشین به منظور حداقل کردن هزینه کل دیرکرد کارها در این مقاله مورد بررسی قرار می‌گیرد. الگوریتمی ساده و در عین حال کارا جهت تعیین جواب بهینه و یا نزدیک به بهینه برای تعیین توالی n کار مستقل روی یک ماشین با هدف حداقل کردن هزینه دیرکرد توسعه داده شده است. در این مدل فرض شده است که هزینه دیرکرد کارها به مدت زمان دیرکرد بستگی دارد و این هزینه به کارهائیکه پس از موعد تحویل خاتمه می‌یابد تعلق می‌گیرد. الگوریتم توسعه داده شده برای حل مسائل نمونه بکار گرفته شده و نتایج محاسباتی آن ارائه و تحلیل شده است.

INTRODUCTION

The problem of sequencing of N independent jobs on a single machine scheduling problem to minimize the mean tardiness of N jobs is considered in this paper. It is assumed that a set of N independent single operation jobs is available for processing at the beginning of scheduling time horizon. It is also assumed that the set up time for each job is independent of job sequence and the processing time of each job is exactly known in advance. Under the conditions that one machine is continuously available and preemption is not permitted, the objective function of the problem can be written as:

$$\text{MINIMIZE} \quad Z = 1 / N \sum_{j=1}^{N} T_j$$

where in the above formula, Z is defined to be the mean tardiness of N jobs, and T_j is the tardiness of job j.

A number of studies dealing with the total tardiness criteria has been conducted and are available in the literature. The early theoretical attempts to the tardiness problems are studies of Elmaghraby [6], Emmons [7], and Montagne [9]. Sirnivasan [15] developed a hybrid algorithm based on the concept of dynamic programming technique to minimize the mean tardiness of N independent jobs on a single machine. The hybrid algorithm solves this problem in two consecutive phases. In the first phase of the algorithm some of N jobs are assigned to the machine based on some rules dominance properties. The remaining jobs are then assigned by the use of the dynamic programming technique.

Shwimer [14] has employed a branch and

bound solution technique to handel this problem. He has shown the superiority of "jump tracking" approach via the "back tracking" approach in the test problems solved in the experimental section of his paper.

Cheng [3] in his published paper considered the problem of assigning due dates of N independent jobs on a single machine with the CON-due date assignment method. In this method a constant flow allowance is assigned to all jobs, in order to minimize the weighted average of missed due dates.

Fry, Armstrong, and Blackston [8] have proposed a heuristic solution algorithm to minimize the total weighted absolute value of penalty when N independent jobs are being scheduled on a single machine subject to earliness and tardiness penalties.

Quaddus [11] employed a linear programming model to find the optimal CON-due date for N independent jobs on a single machine sequencing problem. In this paper the duality theory is used to obtain an optimal solution to the problem.

Potts, and Wassenhove [10] developed an algorithm to minimize the total weighted number of late jobs. The branch and bound algorithm with the priciples of the dynamic programming is used to solve a very large sequencing problem (1000 independent jobs).

Sen, Raiszadeh, and Dillpan [13] are considered a bi-criterion scheduling problem with a linear combination of total flow time and range of lateness as a measure of performance of sequencing N jobs on a single machine problem. A branch and bound solution procedure is designed and used to solve this problem.

Abdul-Razaq and Pott [1] have shown how state space relaxation of dynami programming technique can reduce th computational efforts needed to obtain th optimal sequence of N independent jobs on single machine problem. The objectiv function for the problem is defined as function of jobs holding costs which ar completed before their due dates and th tardiness costs for jobs which are complete after their due dates.

Based on thorough review of the availabl literature, it is found that a unique solutio approach to solve the tardiness problem ha not yet been developed. Most of th approaches developed to date sufferd from number of limitations [4], [12]. The mos important difficulty in solving sequencin problems with tardiness based performanc measures is the fact that tardiness is not a linear function of completion time. Thi replies that any solution approach to thi problem has to be capable of challenging with the combinatorial aspect of the problem. The combinatorial nature of the problem will cause the exponential growth of the solution space, and hence will require a very large computer memory, and may need an extensive computational time. Because of this difficulty, there is apt to be more attention paid to efficient but suboptimal solution techniques.

DEVELOPMENT OF AN ALGORITHM

A thorough conceptual investigation of the behavior of the tardiness problems reveals that a complex optimization procedure has to be employed to solve even the most simple scheduling problems, when tardiness is considered as a performance criterion.

Dynamic programming and branch and bounds techniques demand a large amount of computational time and memory as the problem size increases. Hence a practical solution approach to medium or large problems (more than 15 jobs) could be a heuristic procedure.

The developed algorithm which will be presented in this paper is indeed a heuristic procedure which embodies some simple and yet efficient decision rules to improve an initial solution of the problem. The solution starts with an EDD (Earliest Due Date) schedule and attempts to find the best schedule through rearranging jobs which decrease the value of tardiness. The logic supporting the selection of these jobs are based on the following discussion.

Consider an EDD schedule in which job i proceeds job j. Let us define the following notation:

d_j = the due date of job j

T = the gap between start of job j and end of job i

S = the waiting time of job i

D_{ij} = the total tardiness of job i and job j while job i proceeds job j

D_{ji} = the total tardiness of job i and job j when job j proceeds job i

t_j = the processing time of job j

We are now seeking conditions under which the relocation of a job in the sequence will decrease the total tardiness of the existing sequence. To do so, we examine whether interchange of two jobs reduce the total tardiness. Let us consider the contribution of two jobs i and j. From the definition of tardiness we have:

$D_{ij} = \max(S + t_i - d_i, 0) + \max(S + t_i + T + t_i - d_j, 0)$

$D_{ji} = \max(S + t_j - d_j, 0) + \max(S + t_j + T + t_i - d_i, 0)$

In comparison of two jobs i and j, we examine all the possible conditions and will prove under which condition the interchange of job i and j provides an optimal tardiness value.

I) Let $t_i < t_j$, and $S + t_j < d_j$

$D_{ij} = \max(S + t_i - d_i, 0) + \max(S + t_i + T + t_j - d_j, 0)$

$D_{ji} = \max(S + t_j + T + t_i - d_i, 0)$

since $d_i < d_j$ (we start with an EDD schedule) it can be shown that D_{ij} is always less than or equal to D_{ji}. Hence in this case job i proceeds job j in an optimal manner.

II) Let $t_i < t_j$, and $S + t_j > d_j$

$D_{ij} = \max(S + t_i - d_i, 0) + S + t_i + T + t_i - d_j$

$D_{ji} = S + t_j - d_j + S + t_j + T + t_i - d_i$

it can also be shown that D_{ij} is less than or equal to D_{ji}, and hence job i will proceed job j in an optimal manner.

III) Let $t_i > t_j$, and $S + t_i < d_i$

$D_{ij} = \max(S + T + t_i + t_j - d_j, 0)$

$D_{ji} = \max(S + T + t_i + t_j - d_j, 0)$

it is clear that $D_{ij} < D_{ji}$, and hence job i must proceed job j.

IV) Let $t_i > t_j$, $S + t_i > d_i$, and $T + S + t_i + t_j < d_j$

$D_{ij} = S + t_i - d_i$

$D_{ji} = S + t_j + T + t_i - d_i$

it is seen that $D_{ij} < D_{ji}$, and hence job i must proceed job j.

V) Let $t_i > t_j$, $S + t_i > d_i$, and $S + t_j > d_j$

$D_{ij} = S + t_i - d_i + S + t_i + T + t_j - d_j$

$D_{ji} = S + t_j - d_j + S + t_j + T + t_i - d_i$

in this case $D_{ij} > D_{ji}$, and hence job j, must proceed job i.

VI) Let $t_i < t_j$, $S + t_i < d_i$, and $S + t_j < d_j < S + t_i + t_j + T$

$D_{ij} = S + t_i - d_i + S + t_i + T + t_j - d_j$

$D_{ij} = S + t_j + T + t_i - d_i$

$D_{ij} - D_{ji} = S + t_i - d_j$

It can be seen that if $S + t_i > d_i$, then job j must proceed job i, otherwise job i must proceed job j. This is the only situation in which we cannot decide on the position of jobs i and j in the sequence unless we know the value of S.

The above discussion reveals that in the absence of condition number VI we can always optimally decide whether to interchange two jobs or leave them as they are. In the case of condition VI we let job j proceed job i and this may or may not be an optimal decision rule. This is the only situation that if occurs may lead to non-optimal solution. To reduce the chance of occurrence of this situation we impose a decision rule in our algorithm by which instead of interchanging job i and j, we relocate all the jobs between these two jobs. To support this decision rule we have needed to prove two the following theorems:

Theorem 1. Consider two adjacent jobs i and j. If in an optimal sequence, job i must proceed job j, then regardless of their position in the sequence, if we insert a time span T between these two jobs still job i must proceed job j.

Proof: From the assumption of the theorem if we let the current position of job i, and job j in a sequence to be denoted by [1] and [2] respectively, and define $D_{k[1]}$ as the amount of tardiness of job K in the position 1, then we can write:

$$D_{i[1]} + D_{j[2]} < D_{i[2]} + D_{j[1]}$$

If we insert T between job i and job j we will have a new position for job j in the sequence which we denote by [3] then we have to show that:

$$D_{i[1]} + D_{j[3]} < D_{i[3]} + D_{j[1]} \qquad (1)$$

The inserted T may increase the tardiness of any job which appears in position (3). Let us define:

$$D_{i[3]} = D_{i[2]} + G_i \qquad (2)$$

$$D_{j[3]} = D_{j[2]} + G_j \qquad (3)$$

where G_i and G_j each has a value between O and T. By substituting equation (2) and (3) in inequality (1) we will have:

$$D_{i[1]} + D_{j[2]} < D_{i[2]} + G_i + D_{j[1]}$$

Since G_j can never be greater than G_i, and the remaining terms of the left side of the inequality are less than the remaining terms of the right side therefore the inequality is always held.

Theorem 2. Consider three jobs i, j and k. If in an optimal solution job i must proceed job j and job k must proceed job i regardless of their position in the sequence, then job k must proceed job j.

Proof: Let us denote the current position of job i and job j by [1] and [2], respectively. From the assumption of the algorithm we can write:

$$D_{i[1]} + D_{j[2]} < D_{i[2]} + D_{j[1]} \qquad (4)$$

By the result of theorm 1 we can write:

$$D_{k[1]} + D_{i[2]} < D_{k[2]} + D_{i[1]} \qquad (5)$$

we now add two inequalities (4) and (5) after cancelling out the identical terms from both sides we obtain the following inequality and hence, we reach the proof

$$D_{k[1]} + D_{j[2]} < D_{j[1]} + D_{k[2]}$$

Based on the above theoretical concepts a heuristic algorithm is developed. To present the steps of the developed algotithm, first we need to give some definitions. Let

- C = an ordered set containing the jobs which have been decided to be in the final solution.
- C' = the compliment of C.
- S = the sum of processing time for all jobs in C.
- N = the total number of jobs.
- [i] = indication of position of a job in a sequence.
- $t_{[i]}$ = the processing time of i^{th} job in sequence.
- $d_{[i]}$ = the due date of i^{th} job in sequence.

The steps of the algorithm are presented as below:

STEP 1. Let C be empty, assign all jobs to C' in EDD order, and let i = 1, j = 2.

STEP 2. If $t_{[i]} < t_{[j]}$ go to step 7, otherwise go to step 3.

STEP 3. If $S + t_{[i]} < d_{[i]}$, go to step 7, otherwise go to step 4.

STEP 4. If $S + t_{[i]} + t_{[j]} < d_{[j]}$ go to step 7, otherwise go to step 5.

STEP 5. If $\max(S + t_{[i]}, S + t_{[j]}) > d_{[j]}$ go to step 6, otherwise go to step 7.

STEP 6. Remove the job in position j and assign it to position i, and assign jobs i, i + 1, ..., j - 1 one position further, go to step 7.

STEP 7. Let j = J + 1, if j < N go to step 2, otherwise go to step 8.

STEP 8. Remove job in position i from C' and assign it to the last position in C, let i = i + 1, if i = N stop otherwise go to step 2.

COMPUTATIONAL EXPERIENCES

The developed algorithm is applied to several test problems to check validity of the solution obtained by the algorithm as well as its performance. More than 100 test problems are solved via the developed algorithm and two well known solution procedures. Some of the test problems are selected from the available literature and some are generated using the concept of Mont-Carlo simulation. The size of test problems is varied from four jobs up to twenty jobs.

The required data to generate a test problem are the number of jobs (n), the processing time (t_j) for each job, and the associated due date (d_j) for each job. After deciding on the number of jobs, we used pseudo random number generation to randomly generate t_j' s, and d_j' s. To have a realistic date for the generated test problems we assigned a range for the processing time of each job. Let us define the upper value, and the lower value of t_j's as TU and TL, respectively. Based on the values of TU and TL, a range for the values of d_j' s is determined using the following relations:

$$DL = TL$$

$$DU = N * [TL + (TU - TL) / 2]$$

Where DU and DL, are the upper and lower values of d_j's. Using the generated random number the values of t_i and its associated d_j are generated in the defined ranges while ignoring those d_j's which are smaller than its

Table 1. Summery of Computational Experiences

Number of Problems	Number of Jobs	Time (sec)			Average deviation
		D.P.	H	A	
20	4	0	0	0	0%
20	5	1	0	0	0%
16	8*	23	7	23	39%
55	10	118	10	.1	1.55%
10	15	**	217	.6	2.23%
10	20	**	**	2	***

* : The problems are selected from literature

** : Time is too large

*** : The optimal value is not available associated tj's.

Two powerful and exact algorithms are selected to compare the efficiency of the algorithm for solving test problems. The selected algorithms are the dynamic programming approach and the hybrid algorithm [2]. Table 1 summarizes the results of the computational experiences.

In this table the first column represents the number of problems which has been solved, and the second column represents the size of the problem in each category. The following three columns represent the average time spent to solve problems in each category via D. P. (dynamic programing), H (hybrid algorithm), and A (the developed algorithm). The double stars in these columns represent a very large amount of time to obtain the optimal solution. However the required time necessary to obtain optimal solution for the cases of problems with 15 jobs using D. P., 20 jobs D. P., and 20 jobs using H has been determined to be 1, 5, and 17 hours, respectively. In the last column the average deviation from the optimal solution is shown. It is to be note that the optimal solution of the majority the problems in the case of 20 jobs could n be obtained by D. P. and H algorithm. Th optimal solution of two problems out of th 10 problems solved via H algorithm obtained in the case of 20 jobs. The solutior provided by the developed algorithm fo both of these problems were also optimal

The computational time necessary t reach an optimal solution by D. P. and algorithm, and a near optimal solution by algorithm verses problem size is depicted i Figure 1. Based on 131 test problems solve by the above mentioned algorithms it is see that when an exact algorithm is employe the computational time grow exponentially as the problem size increase while in the case of using the develope algorithm, this growth is most likely linea This fact and the small deviation of th solution obtained by A algorithm from th optimal solution, reveal the power an efficiency of the developed algorithm.

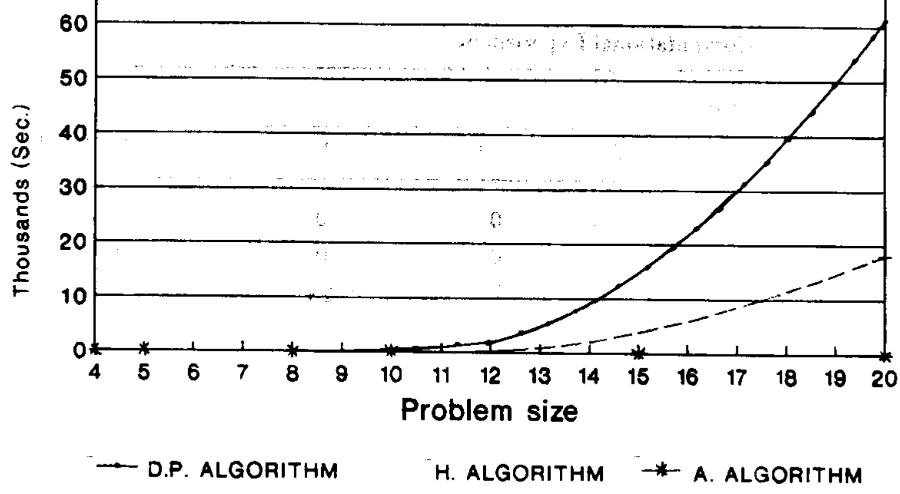

Figure 1. Computational time versus problem size

CONCLUSION

In this paper a powerful algorithm is developed to solve N/1 sequencing problems. The average tardiness is the measure of performance for assigning N single operation independent jobs on a single machine.

The performance of the developed algorithm is tested using 131 test problems with respect to the closeness of the solution to the optimal solution and its computional time. It is deduced that this algoritim uses very little computer meory and requires very short computer time.

Out of 123 test problems that could have optimal solutions, the average deviation of the solutions obtained by the developed algorithm was less than 1% while the computational time for large problems (20 jobs) was in the range of a few seconds.

REFERENCES

1. T. S. Abdul - Razaq and C. N. Potts. *J. opl. Res. Soc.* 39, 141 (1988).
2. K. R. Baker, "Introduction to sequencing and Scheduling", John Wiley and Sons, Inc, NewYork (1974).
3. T. C. E. Cheng, Computer *Opns Res.* 14,537(1987).
4. N. Christofides A. Mingozzi and P. Toth. *Network* 11, 145 (1981).
5. R. W. Conway, W. L. Maxwell, and L. W. Miller, "Theory of Scheduling", Addisòn - Wesley, Reading, Mass, (1967).
6. S. E. Elmaghraby, *J. of Industrial Engineering*, 19, (1968).
7. H. Emmons, *Opns. Res.* 17, (1969).
8. T. D. Fry, R. D. Armstrong and J. H. Blackstone, *IIE* Transaction 19, (1987).
9. E. R. Montagne, Jr. "Sequencing with Time Delay Costs", *Industrial Engineering Research Bulletin*, No. 5, Arizona State University, (1969).
10. C. N. Potts, and L. N. Van Wassenhove, Management Science 34, 843.
11. M. A. Quaddus, *J. Opl. Res.* Soc. 38, 353, (1987).
12. A. H. G. Rinnooy Kan B. J. Legeweg and J. K. Lenstra, *Opns. Res.* 23, 908, (1975).
13. T. Sen, F. M. E. Raiszadeh and P. Dileepan, *Technical Notes, Management Science*, 34, 254.
14. J. Shwimer. *Management Science*, 18, (1972).
15. V. Srinivasan, *Naval Research Logistics Quarterly*, 18, (1971).
16. L. J. Wilkerson, and J. D. Irwin, *AIIE Transaction*, 3, (1971).