Using a Modified Simulated Annealing Algorithm to Minimize Makespan in a Permutation Flow-shop Scheduling Problem with Job Deterioration

S.A. Seyed-Alagheband, H. Davoudpour, S.H. Hashemi Doulabi, M. Khatibi

Abstract - **Recently, the problem of scheduling flowshops with linear deteriorating jobs has attracted the attention of researchers. However, due to the computational complexity, there have been few studies in this class of problems having more than two machines. In this paper, we strive to address this problem considering *m*-machine case. We deployed a modified simulated annealing algorithm to solve this NP-hard problem in a reasonable computation time. Results obtained from the algorithm are an illustration of its high efficiency.**

Keywords - **Scheduling, Deteriorating jobs, Simulated Annealing, Meta-heuristics, Makespan**

I. INTRODUCTION

The flow shop scheduling problem (FSSP) is a well-known and complex combinatorial optimization problem with many variations. In the permutation FSSP (PFSP), all jobs must enter the machines in the same order and the goal is to find a job permutation that minimizes a specific performance measure (usually makespan or total flowtime).

In most of the machine scheduling problems, the job processing times are assumed to be constant; however, in many scheduling environments, job processing times are an increasing function of their starting times. This phenomenon, known as deterioration, has been extensively studied in the last decade in various machine settings and performance measures.

Pinedo [8] reviewed comprehensively the stochastic scheduling problems. It is observed that in the last decade there is a growing interest in considering problems involving scheduling with time-dependent processing times, i.e., problems where processing time of the job depends on the starting time on each machine. There are two main categories for these sorts of the problems.

S.A. Seyed-Alagheband, H. Davoudpour, S.H. Hashemi Doulabi and M. Khatibi are with the Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran
(e-mail: Alagheband@aut.ac.ir)

If the processing time decreased by postponing the operation on machine it is called jobs with learning effect, for example, in drying process, if the drying operation of the batch is postponed to a later time, natural evaporation leads to decrease in processing time. In the second category processing time increased by postponing the starting time of the process; which is known as deteriorating jobs. Kunnathur and Gupta [5], Mosheiov [6] and Sundararaghavan and Kunnathur [7] have depicted the applications of these problems, some of them are mentioned below:

- In crisis management, like fire fighting or earthquake, where the situation deteriorated by time, the processing time of each reaction increased by postponing the process
- Maintaining and repairing production machines in factories usually depends on the duration of their active work time; if the interval between each repairing audit increases, the time for repairing operation increases as well
- The recovery duration of many diseases depends on the curing start time
- The similar case is for military attacks or defense actions where the condition worsens with time

II. LITERATURE REVIEW

Since the first publication of Gupta and Gupta [9], there has been a growing interest in the literature to study scheduling problems of deteriorating jobs. They proposed that each job processing time on a machine can be expressed by the following function: $P_i = f(t_i)$, in which t_i is the starting time of i^{th} job on the machine. After the definition of deteriorating jobs, many researchers tried to extend the basic problem; however, they mostly concentrated on simple linear function. Recently, other functions such as step-wise functions have also been considered.

Alidaee and Womer [10] and Cheng et al. [11] provided extensive reviews of this subject for a single machine. However, topics relating to deteriorating jobs are mostly discussed in a single machine setting; they remain relatively unexplored in the multi-machine environment. In recent years, some authors investigated two-machine flow shop with deteriorating jobs via different objective functions. Wang et al. [12] considered minimizing total completion time. They have proposed a heuristic algorithm to overcome the inefficiency of the branch-and-bound algorithm. Wu and Lee [13] studied minimizing mean flow time and developed a branch-and-bound procedure and several heuristic algorithms to search for the optimal solution and the near-optimal solutions. Later, Lee et al. [14] provided an extensive review of that problem with makespan function. Because of computational complexity, there have been fewer studies in problems having more than two machines. Mosheiov [15] considered the case of m-parallel identical machines, with one step operation and multi-step deterioration function. He introduced a heuristic algorithm for minimizing make-span and compared the answers with that of integer model solution. Hsieh and Bricker [16] studied two deteriorating job multi-machine scheduling problems with the goal of minimizing the makespan. For the simple linear deterioration model, they proposed a heuristic algorithm, proved that the ratio of the makespan obtained by the heuristic algorithm to the optimal makespan was bounded, and showed that their heuristic algorithm possessed an asymptotic optimality property. For the general linear deterioration model, they proposed three heuristic algorithms and indicated that these heuristic algorithms provided good solutions by a complete enumeration. Later on, Mosheiov [17] studied the m-parallel identical machines problem having deterioration function $p_i = \alpha_i t_i$. He showed that the problem was NP-hard even for two identical parallel machines under the simple linear deterioration assumption. Thus, a heuristic algorithm and a lower bound were provided and tested.

Khalil and Samson [18] solved the (Hsieh and Bricker [16]) problem by a simulated annealing algorithm and compared their results with former ones. Gawiejnowicz et al. [19] considered the m-parallel identical machines problem with $p_i = x_i + \alpha_i t_i$ and $p_i = \alpha_i t_i$ deterioration functions and completion time objective function. They proposed another heuristic algorithm and reported their computational results. Kang and Ng [20] studied the NP-hard problem of scheduling n deteriorating jobs on m identical parallel machines to minimize the makespan. They presented a fully polynomial-time approximation scheme for the problem. Ji and Cheng [21] considered the parallel-machine scheduling problem in which the objective is to minimize the total completion time. They deployed a Fully Polynomial Time Approximation Scheme (FPTAS) for the case with m identical machines, where m is fixed. Lee and Wu [22] investigated a multi-machine scheduling problem in which machines are not always available; each machine is assumed to have a maintenance period which is known in advance. Both the resumable and non-resumable cases are discussed with the objective of minimizing the makespan. A lower bound and a heuristic algorithm are proposed for each case.

In all of the multi-machine studies, only parallel machines have been investigated. However, Mosheiov [23] has assumed a simple linear deterioration function $p_i = \alpha_i t_i$ and makespan objective function and provided a complete analysis of the complexity of flow-shops, open-shops and job-shop problems. He has proved that the Johnson algorithm provides optimal result for two-machine case in flow-shop environment. He also showed that the computational complexity of problems having more than two machines falls in the category of NP-completeness.

The $n\backslash m\backslash f\backslash C_{max}$ problem is one of the classic machine scheduling problems. In this case, m different operations are done by m different machines on n different jobs and jobs follow the flow shop model, i.e., the operational sequence of all the jobs remains the same. If the job sequence on all machines be also the same, it is called permutation scheduling problem which is noted by $n\backslash m\backslash p\backslash C_{max}$.

Johnson [24] has found optimal algorithm for $m = 2$. Sice then, many researchers have proposed different heuristics and meta-heuristic algorithms for solving problems for $m>2$. There have been numerous comparisons between the performances of proposed algorithms. Ruiz and Concepcion [25] have presented a comprehensive review of these comparisons. They compared 25 different heuristic and meta-heuristic methods for solving $n\backslash m\backslash p\backslash C_{max}$ problem by using Taillard's [26] test method- a set of 120 instances of various sizes, having 20, 50, 100, 200 and 500 jobs and 5, 10 or 20 machines, with 10 problems inside each set. It is observed that the CDS and Palmer heuristic algorithms are well accepted algorithms, which have been the base for most of the algorithms and have good performances with comparatively short run times.

III. PROBLEM SPECIFICATIONS

In this study, we have considered $n\backslash m\backslash p\backslash C_{max}$, problem for $m>2$ with simple linear deterioration function $p_{ij}=\alpha_{ij}t_{ij}$ where i refers to order of sequence of the job on machine and j refers to order of machine itself. The problem under consideration has been denoted in the form of $n\backslash m\backslash p$, $p_j = \alpha_{ij}t_{ij}\backslash C_{max}$ where it is assumed that all jobs are available on $T_0 = 1$. As mentioned earlier this problem is NP-complete for $m>2$.

It is assumed that a job's initially arriving order is introduced by i, but jobs are allocated to all the machines with permutation order which is identified by an array like 1. in the following we use of these denotes: The notations are as follows:

$OI_i=[k]$: The k^{th} job in array I is i^{th} initial arrived job

$S_{[k]j}$: The start time of operation on k^{th} job in array I on j^{th} machine

$P_{[k]j}$: The process time of k^{th} job in array I on j^{th} machine

$C_{[k]j}$: The process completion time of k^{th} job in array I on j^{th} machine

$C_{[k]}$: The completion time of k^{th} job in array I on all machines

$\alpha_{[k]j}$: The deterioration coefficient for k^{th} job m array I on j^{th} machine

Now, assume that the initial order of jobs is rearranged and allocated to machines by the order of array I, so for the first job in array I on the first machine we should have:

$$S_{[1]1} = 1$$

$$P_{[1]1} = 1 \times \alpha_{[1]1} = \alpha_{[1]1}$$

$$C_{[1]1} = S_{[1]1} + P_{[1]1} = 1 + \alpha_{[1]1}$$

For the second job in array I on the first machine we should have:

$$S_{[2]1} = C_{[1]1} = 1 + \alpha_{[1]1}$$

$$P_{[2]1} = S_{[2]1} \times \alpha_{[2]1} = (1 + \alpha_{[1]1})\alpha_{[2]1}$$

$$C_{[2]1} = S_{[2]1} + P_{[2]1} = (1 + \alpha_{[1]1})(1 + \alpha_{[2]1})$$

Likewise, for the k^{th} job in array I on the first machine, we should have:

$$S_{[k]1} = C_{[k-1]1} = \prod_{l=1}^{k-1}(1 + \alpha_{[l]1})$$

$$P_{[k]1} = S_{[k]1} \times \alpha_{[k]1} = \alpha_{[k]1} \times \prod_{l=1}^{k-1}\left(1 + \alpha_{[l]1}\right)$$

$$C_{[k]1} = \prod_{l=1}^{k}(1 + \alpha_{[l]1})$$

Similarly for the k^{th} job in array I on the j^{th} machine, we should have:

$$S_{[k]j} = Max\left\{C_{[k-1]j}, C_{[k]j-1}\right\}$$

$$P_{[k]j} = S_{[k]j} \times \alpha_{[k]j}$$

$$C_{[k]j} = S_{[k]j}\left(1 + \alpha_{[k]j}\right) \quad k = 1, \dots, n \quad j = 1, \dots, m$$

IV. Proposed simulated annealing algorithm

Simulated annealing algorithm is a well-known neighborhood search approach, deriving its acceptance mechanism from annealing process in order to let the current solution escape from the local optima. It starts with an initial solution and navigates around it by any kind of neighborhood search structure. In order to escape from local optima, this algorithm typically has a control parameter called temperature (T). In high temperatures, the algorithm may replace the current solution x by its worse neighbor solution x' by calculating an acceptance probability $e^{(-(f(x') - f(x) / T)}$. As the temperature falls through the algorithm, the probability will decrease and the result will converge to the best solution.

Simulated annealing algorithm has been widely used in classical scheduling problems; however, to the best of our knowledge, this algorithm is not applied to the problem of scheduling deteriorating jobs. Classical SA algorithm is usually inefficient and returns unacceptable results. One of the shortcomings of SA is its large number of parameters. An efficient SA needs to be tuned properly. Precise definition of the neighborhood search structure is also another weakness of the algorithm. One structure may perform very well in the early stages of the algorithm; however, it may be powerless to help the current solution escape from the local optima. As the classical version of SA is inefficient to our case as well, we provided several modifications to the classical SA. The modified SA algorithm provided by [4] is the closest SA algorithm to our problem. The following subsections provide details about the parameters defining the behavior of the algorithm.

A. Initialization

Providing a suitable solution for the initialization of the algorithm is important and has a significant influence on the output. There are several ways introduced in the literature which provide a good initial solution for the start of the algorithm [2, 3]. We tested different approaches like job-based and random key, but there have not been meaningful differences between approaches in solution quality. Therefore, we just provided a random initial solution for this problem.

B. Temperature settings

Settings related to the temperature issue include initial temperature, final temperature and number of temperatures to be seen. Both initial and final temperatures need to be selected wisely. The initial solution should be chosen so that around 25 percent of worse cases can be accepted. The final temperature should be low enough to not let worse solutions replace with better ones. In this problem, we consequently choose 10 and 0.005 as the initial and final temperatures.

Clearly, increasing the number of observed temperatures would improve the solution quality, but however, would increase the running time of algorithm. Thus, a tradeoff should be made regarding this issue. In this case, we have set the observed temperatures to 55*n ('n' is the number of jobs).

C. Cooling schedule

To control the acceptance mechanism, it is important to determine with what trend the temperature should be dropped so that the current solution is not allowed to bias toward its bad neighbors through most of the running time of the algorithm.

In general, there are two well-known types of cooling schedule in the literature of SA: linear, exponential (see Table 1 for more details. T_0, T_f and N are the initial temperature, final temperature and the expected number of observed temperatures, respectively). Regarding our problem, experiments show that using exponential trend would result in better outputs and other approaches guides the current solution mostly to bad neighbors.

Table 1. Types of cooling schedule

Cooling schedule	Formula
Linear	$T_i = T_0 - i * \frac{T_0 - T_f}{N} \quad i = 1, \dots, N$
Exponential	$T_i = \frac{A}{i+1} + B \quad A = \frac{(T_0 - T_f)(N+1)}{N}$ $B = T_0 - A \quad i = 1 \dots N$

D. Number of neighborhood search in each temperature (NNS)

In each temperature, some sorts of neighborhood search should be carried out. Most of the previous works rely on selecting a constant number as the NNS, for example, set it to 20, 100, 500, etc. In this paper, a different approach is employed. Our approach suggests that the number of neighborhood search in higher temperatures be lower and as the temperature falls, NNS will increase. We hereby used an exponential raise in NNS, setting 20 and 600 respectively as the initial and final NNS. Results declared the remarkable higher efficiency of exponential NNS compared to constant NNS.

E. Neighborhood Search Structure (NSS)

The neighborhood search structure is the way of producing new different solutions from the current solution. There are various kinds of NSSs available in the literature, differentiated both in terms of nature and size of navigation. Three kinds of NSSs are widely used in the literature [4]:

- Single Point Operator (SPO): The position of one randomly picked job is regenerated
- Swap Operator: The positions of two randomly selected jobs are swapped
- Inversion Operator: The positions between two randomly selected cut points are reversed

The use of an appropriate NSS would have a tremendous influence on the quality of solutions. One of the shortcomings of SA algorithm in our case is that despite its natural mechanism of escaping from local optima, the algorithm causes the solution encounter a premature convergence to local optima. As a result, it brings about considering major modifications to our algorithm in order to cause the trapped solution escape from local optima and navigate unvisited spaces. We have defined a new mechanism for this end, using an effective type of NSS so that it will not cause the solution being trapped in local optima. We defined two new operators, which are extensions of the existed NSSs:

- Multi Single Point Operator (MSPO)
- Multi Swap Operator (MSO)

The aforementioned operators perform like SPO and Swap, but instead, in an iterative manner. In this problem, we have their iterations to 2 times, which means, for instance, MSPO regenerates the place of 2 randomly picked jobs.

To our experiments and knowledge, none of the operators mentioned above can solely prevent the solution from getting trapped in the local optima. To establish a well-organized NSS for our problem, we employed the concept of variable neighborhood search (VNS) within our SA algorithm. More precisely, we used a combination of operators together and if one NSS does not provide better solutions after some algorithm iterations, another NSS is used. In this problem, if the NSS *i* does not improve the best solution after 7 consecutive temperatures, the NSS *i*+1 is used. We consequently used 'SPO - Swap - Invertion - MSPO - MSO' as the NSS; which through this, the algorithm starts from SPO and searches the neighborhood of the current solution until it is unable to continue the improvements after 5 consecutive temperatures. Then, the next NSS in the queue, 'Swap', is used. Through this process, if the algorithm reaches the end of the queue, the NSS will restart.

V. COMPUTATIONAL RESULTS

For the purpose of the assessment of the effectiveness and competitiveness of different algorithms, our proposed SA algorithm is compared with other methods in the literature. We compared our algorithm with 3 heuristic algorithms, a fortified local search and an Electromagnetic Algorithm presented in [1]. The problem is run in MATLAB 7.0 and on a PC with 1.5 GHz Intel Core 2 Duo and 2 GB of RAM memory. The test problem used in our work is the one used in the work conducted by [1]. They considered number of jobs in five levels (5, 15, 25, 35, and 45) and number of machines in three levels (3, 4 and 5) and generated 15 test beds for this problem. They compared the results of electromagnetic algorithm with results from:

- One randomly produced arrangement of jobs
- Two modified CDS and Palmer heuristics
- results from the fortified local search approach

Table 2 shows the results obtained from SA and other approaches developed in [1] along with the electromagnetic approach. Clearly, as is shown in Table 1, the modified SA algorithm dominates the results from heuristics and electromagnetic approach and is so close to the local search results, in some cases, dominating its results as well.

Table 2. Comparison between SA and other approaches

Jobs No	Machine No	Random	CDS	Palmer	Average results		
					EM	Local search	SA
5	3	23	16	16	15	14	14
	4	49	33	31	30	29	28
	5	94	54	58	55	52	52
15	3	2,509	794	898	744	710	710
	4	8,530	2,626	2,457	2,035	1,703	1,656
	5	15,081	4,953	8,013	4,404	3,469	3,465
25	3	230,675	65,302	74,472	66,634	65,302	65,302
	4	582,487	117,866	103,036	84,846	68,766	67,369
	5	1,183,794	241278	249,314	176,780	135,719	130,341
35	3	31,167,300	5,325,850	5,271,280	4,323,770	4,189,163	4,189,219
	4	52,847,270	6,483,760	6,645,520	4,522,562	4,194,543	4,383,704
	5	79,720,100	15,622,920	19,553,640	11,034,873	6,834,972	7,012,215
45	3	1,031,607,000	176,950,400	166,900,000	152,469,100	148,273,533	148,337,727
	4	1,730,438,000	222,345,400	223,166,400	136,708,744	93,730,800	90,819,519
	5	6,232,300,000	520,280,000	597,990,000	419,038,539	280,830,767	300,423,100

VI. CONCLUSION AND FUTURE RESEARCH

In this paper, we investigated the problem of scheduling deteriorating jobs in a permutation flow shop environment. The problem is proved to be NP-hard in nature; therefore, we developed a simulated annealing algorithm with some modifications in the classical algorithm. The algorithm is shown to be efficient in terms of solution quality compared to other approaches available in the literature.

Job deterioration is a very promising issue and due to the very simple structure of the problem, many extensions of flow-shops, including hybrid and flexible flow-shops, job-shops and open-shops can be explored as well. Moreover, using more than one objective and developing other metaheuristic algorithms, especially population-based meta-heuristics, can be regarded as some other future research directions.

REFERENCES

[1] Davoudpour H., Hadji Molana M. (2008) Solving flow shop sequencing problem for deteriorating jobs by using electromagnetic algorithm. Journal of Applied Sciences 8 (22) 4121-4128

[2] Ruiz R., Maroto C. (2006) A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. European Journal of Operational Research 169:781–800, doi:10.1016/j.ejor.2004.06.038

[3] Kurz ME, Askin RG (2004) Scheduling flexible flow lines with sequence-dependent setup times. European Journal of Operational Research 159(1):66–82, doi:10.1016/S0377-2217(03)00401-6

[4] Naderi B, Zandieh M, Roshanaei V (2008) Scheduling hybrid flowshops with sequence dependent setup times to minimize makespan and maximum tardiness. International Journal of Advanced Manufacturing Technology, doi: 10.1007/s00170-008-1569-3

[5] Kunnathu, A.S, Gupta S.K (1990) Minimizing the makespan with late start penalties added to processing times in a single facility scheduling problems. Eur. I. Operat. Res., 47: 56-64.

[6] Mosheiov G (1994) Scheduling jobs under simple linear deterioration. Comput. Operat. Res., 21: 653-659.

[7] Sundararaghavan P.S, Kunnathur A.S (1994) Single machine scheduling with start time-dependent processing times: Some solvable cases. Eur. I. Operat. Res., 78: 394-403.

[8] Pinedo M (1995) Scheduling, Theory, Algorithms and Systems. 1 st EOO. Prentice-hall, Englewood Cliffs, NI.

[9] Gupta J.N.D, Gupta S.K (1988) Single facility scheduling with nonlinear processing times. Computers and Industrial Engineering 14: 387-393

[10] Alidaee N.K.W (1999) Scheduling with time dependent processing times: Review and extensions. J. Operat. Res. Soc., 50: 711-720.

[11] Cheng T.C.E, Ding Q, Lin B.M.T (2004) A concise survey of scheduling with time-dependent processing times. Eur. J. Operat. Res., 152: 1-13.

[12] Wang, I.B., e.T. Daniel Ng, T.e.E. Cheng and L.L. Liu, 2006. Minimizing total completion time in a twomachine flow shop with deteriorating jobs Applied Math. Com putat., 180: 185 -193.

[13] Wu e.e, Lee w.e (2006) Two-machine flowshop scheduling to minimize mean flow time under linear deterioration. Int. J. Prod. Econ., 103: 572-584.

[14] Lee, W.e., Wu e.e., Wen e.e. Chung Y.R (2007) A two-machine flowshop makespan scheduling problem with deteriorating jobs Comput. Ind. Eng., 54: 737-749.

[15] Mosheiov G (1995) Scheduling jobs with step-deterioration: minimizing Makespan on a single and multi-machine. Comput. Ind. Eng., 28: 869-879.

[16] Hsieh Y. C, Bricker D.L (1997) Scheduling linearly deteriorating jobs on multiple machine. Comput. Ind. Eng., 32: 727-734.

[17] Mosheiov G (1998) Multi-machine scheduling with linear deterioration. INFOR., 36: 205-214.

[18] Khalil S.H, Samson M (2001) Scheduling linearly deteriorating jobs on parallel machines: A simulated annealing approach Prod. Plann. Control, 12: 76-80.

[19] Gawiejnowicz S, Kurc W, Pankowska L (2006) Parallel machine scheduling of deteriorating jobs by modified steepest descent search.

In: Parallel Processing and Applied Mathematics, 6th International Conference, September 11-14, Poman, Poland, pp: 116-123.

[20] Kang, L. and C.T. Ng, 2007. 'A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs. Int. J. Prod. Econ., 109: 180-184.

[21] Ji M, Cheng T.C.B (2008) Parallel-machine scheduling with simple linear deterioration to minimize total completion time Eur. J. Operat. Res., 188: 342-347.

[22] Lee w.e, and Wu c.e (2008) Multi-machine scheduling with deteriorating jobs and scheduled maintenance. Applied Math. Modell., 32: 362-373.

[23] Mosheiov G (2002) Complexity analysis of job-shop scheduling with deteriorating jobs. Discrete Applied Math., 117: 195-209.

[24] Johnson S.M (1954) Optimal two and three stage production schedules with setup times included. Naval Res. Logist. Q., 1: 61-68.

[25] Ruiz R, Concepcion M (2005) A comprehensive review and evaluation of permutation flowshop heuristics. Eur. J. Operat. Res., 165: 479-494.

[26] Taillard (1993) Benchmark for basic scheduling problems. Eur. J. Operat. Res., 47: 278-285.