A GRASP algorithm for the Two-Machine Flow-Shop Problem with Weighted Late Work Criterion and Common Due Date

Conference Paper · December 2009

DOI: 10.1109/IEEM.2009.5373211

CITATIONS
4

READS
104

5 authors, including:

Amir Zadeh
Wright State University
49 PUBLICATIONS **827** CITATIONS
SEE PROFILE

Hamid Afshari
Dalhousie University
56 PUBLICATIONS **866** CITATIONS
SEE PROFILE

Kamran Kianfar
University of Isfahan
15 PUBLICATIONS **330** CITATIONS
SEE PROFILE

Michel Fathi
University of North Texas
113 PUBLICATIONS **762** CITATIONS
SEE PROFILE

A GRASP algorithm for the Two-Machine Flow-Shop Problem with Weighted Late Work Criterion and Common Due Date

Amir Hasanzadeh [1], Hamid Afshari[1], Kamran Kianfar[2], Mehdi Fathi[3], Afshin Oroojlooy Jadid[2]
[1] Department of Industrial Engineering, Amirkabir University of Technology, Tehran, Iran
[2] Department of Industrial Engineering, Isfahan University of Technology, Tehran, Iran
[3] Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran
(amir.hasanzade@aut.ac.ir)

Abstract - **In this paper, a metaheuristic approach for the two-machine flow-shop problem with a common due date and the weighted late work performance measure ($F2|d_j=d|Y_w$) are presented. The late work criterion estimates the quality of a solution with regard to the duration of the late parts of jobs, not taking into account the quantity of the delay for the fully late activities. Since the problem mentioned is known to be NP-hard, a trajectory methods, namely GRASP is proposed based on the special features of the case under consideration. Then, the results of computational experiments are reported, in which the metaheuristic solution is compared with exact approach and three other heuristic methods' results.**

Keywords – **Scheduling, Flow-Shop, Late work criteria, Metaheuristic, GRASP.**

I. INTRODUCTION

The rapid development of real-time systems makes the due date involving criteria especially useful and important, increasing the interest devoted by researchers to this branch of the scheduling theory. The quality of solutions in real systems is usually estimated from different points of view, which can be modeled by different performance measures (cf. e.g. [1–4]).

The late work criteria are relatively new objective functions, which have not been so intensively explored as the maximum lateness or tardiness ones, for example. They estimate the quality of a solution with regard to the number of tardy units of particular activities executed in a system. The late work concept was introduced in the context of a scheduling problem on identical parallel machines [5] and, then, applied to uniform [6] and single [7–10] machine cases. Recently, practical motivations have directed the research to the shop environment, i.e. to systems with dedicated machines [11–17].

For example, modern control systems [5] utilize a great amount of information on the managed environment, gathered from different sensing devices. Hard real-time restrictions cause that some pieces of information are lost, if they are exposed beyond feasible periods, due to e.g. system overloading or failures. Any control algorithm has to work out its steering strategy based only on the information available. The information loss, modeled as late work, should be minimized in order to increase the accuracy and quality of a control process.

The late work minimization can be also considered at different levels of managing flexible manufacturing systems [17]. These production environments usually work in a shift manner, which complicates planning a production. The shift length, or more generally speaking, a planning horizon, may be considered as a common due date for all tasks realized in a system. Activities which cannot be scheduled within a certain planning horizon, modeled as late work, have to be assigned to the following ones. Obviously, planners tend to minimize late work, i.e. the amount of work which has to be performed in the incoming planning slot in addition to orders newly appearing in a system. The late work parameter allows validating a production plan also from a slightly different point of view. Interpreting customer orders as tasks to be executed, minimizing the late work is equivalent minimizing parts of orders, which are delayed. Obviously, every customer is interested in minimizing late parts of his/her orders. Moreover, taking into account a fine, which has to be usually paid for delays, an owner of a system is also concerned in minimizing financial loss caused by the work not finished on time.

Finally, it is worth to be mentioned that the late work idea is a special case of the imprecise computation model, in which tasks are divided into two parts: a mandatory and an optional one. This model has its own extensive application field, especially in real-time systems (e.g. flight control systems). In this paper, we continue the research [14,15,16] on the two-machine flow-shop problem with the weighted late work criterion and a common due date, which is known to be NP-hard [18], presenting a metaheuristic approaches for its solution. In Section 2, the formal definition of the case under consideration is given. In Section 3, the *GRASP* method is described, while Section 4 contains the results of computational experiments performed for this search strategy. Some conclusions are given in Section 5.

II. Problem formulation

The two-machine flow-shop problem with the weighted late work criterion and a common due date, $F2|d_j = d|Y_w$, concerns the scheduling of a set of jobs $J = \{J_1, \ldots, J_j, \ldots, J_n\}$ on two dedicated machines M_1, M_2. Each job J_j has to be performed first on machine M_1 and then on M_2 for p_{1j} and p_{2j} time units, respectively. Each machine can process only one job at any time and, analogously, each job can be executed by only one machine at any time. We look for a non-preemptive schedule minimizing the late work in the system, i.e.

978-1-4244-4870-8/09/$26.00 ©2009 IEEE

minimizing the amount of work executed after a given common due date d. Denoting by C_{ij} the completion time of job Jj on machine M_i , the late work Y_j for this job is determined as (cf. Fig. 1):

$$Y_j = \Sigma_{i=1,2} \min\{\max\{0, C_{ij} - d\}, p_{ij}\}$$

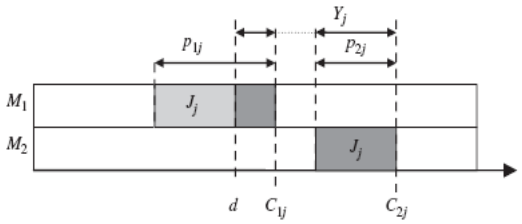

Fig. 1. The late work parameter Y_j for job J_j in the two-machine flow shop enviroment

The criterion value to be minimized, estimating the quality of a complete schedule for the whole set of n jobs, taking into account their given weights w_j , j = 1, . . . , n, is determined as:

$$Y_w = \Sigma_{j=1}^{n} w_j Y_j$$

Problem $F2|d_j = d|Y_w$ stated above is NP-hard [14], since a polynomial transformation was constructed from the set partition problem to its decision counterpart. Moreover, it cannot be NP-hard in the strong sense, because a dynamic programming (DP) method with pseudo-polynomial time complexity, $O(n^2d^4)$, was proposed. This exact approach is important mainly from a theoretical point of view, because it determines the complexity status of the case under consideration, as NP-hard in the ordinary sense.

The theoretical studies on problem $F2|d_j = d|Y_w$ were followed by the computational analysis of exact methods solving it [15]. The complex DP approach was compared to an enumerative approach in order to validate its correctness in practice, as well as to validate their efficiency. Unfortunately, DP found optimal solutions in reasonable time only for small problem instances. Thus, to solve the problem efficiently, heuristic approaches had to be proposed. First, a list scheduling method was implemented for the problem under consideration and compared to DP and enumerative algorithms. The list approach is a scheduling technique commonly used, especially for practical applications, because of its ease of implementation and the low time complexity. The computational experiments showed a very high efficiency of the list scheduling algorithm from the run time and the solution quality points of view in comparison to exact strategies. This heuristic constructed solutions with a criterion value of only 2.5% worse, on average, than the optimum. The high efficiency of the list approach resulted from the particularities of the problem under consideration, whose optimal solution has a very specific structure.

Taking into account the encouraging results of the list scheduling algorithm, it was interesting, whether a further improvement of the solution quality could be obtained by extending this approach with an additional search engine, i.e. by proposing Metaheuristic methods.

III. Metaheuristic approach

In this paper, we present a GRASP algorithm for problem $F2|d_j = d|Y_w$. Similarly to the DP approach [14] and the list scheduling algorithm [15], these approach is based on the specific structure of an optimal solution of the problem under consideration. It was proven [14,17] that in an optimal schedule, the set of early jobs, executed before a common due date, has to be sequenced in Johnson's order, which is optimal from the schedule length point of view. Thus, any method solving problem $F2|d_j = d|Y_w$ (cf. Fig. 2) has to select the first late job (L_1) in the system and to divide the remaining activities into two sets of early (E) and late jobs (L). Early jobs are scheduled by Johnson's algorithm, while the late activities are executed in non-increasing order of their weights w_j . In consequence, the crucial element of any method solving the problem is taking the decision whether a particular job is processed early or late in a schedule.

We consider in this paper a combinatorial optimization problem, defined by a definite ground set E = {1,…,n}, a set of feasible solutions $F \leqslant 2^E$, and an objective function $f : 2^E \to R$. In the minimization version, we search an optimal solution $S^* \in F$ such that $f(S^*) \leq f(S); \forall S \in F$. The ground set E, the cost function f, and the set of feasible solutions F are defined for each specific problem. For instance, in the case of the traveling salesman problem, the ground set E is that of all edges connecting the cities to be visited, f(S) is the sum of the costs of all edges $e \in S$, and F is formed by all edges subsets that determine a Hamiltonian cycle.

The GRASP (Greedy Randomized Adaptive Search Procedure) metaheuristic [19, 20] is a multi-start or iterative process, in which each iteration consists of two phases: construction and local search. The construction phase builds a feasible solution, whose neighborhood is investigated until a local minimum is found during the local search phase.

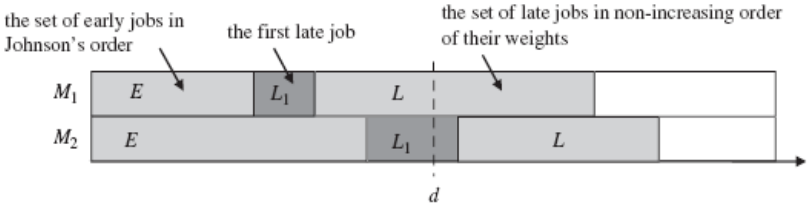

Fig. 2. The general structure of an optimal solution for problem $F2 \mid d_j = d \mid Y_w$ (with the first late job partially late on machine M_2)

The best overall solution is kept as the result. An extensive survey of the literature is given in [21]. The pseudo-code in Figure 3 illustrates the main blocks of a GRASP procedure for minimization, in which Max Iterations are performed and Seed is used as the initial seed for the pseudorandom number generator.

Fig. 4. illustrates the construction phase with its pseudo-code. At each iteration of this phase, let the set of candidate elements be formed by all elements that can be incorporated to the partial solution under construction without destroying feasibility. The selection of the next element for incorporation is determined by the evaluation of all candidate elements according to a greedy evaluation function.

```
procudure GRASP(Max_Iterations, Seed)
1  Read_Input();
2  for k = 1,...., Max_Iterations do
3      Solution ← Greedy_Randomized_Construction(Seed);
4      Solution ← Local_Search(Solution);
5      Update_Solution(Solution, Best_Solution);
6  end;
7  return Best_Solution;
end GRASP.
```

Fig. 3. Pseudo-code of the GRASP metaheuristic

This greedy function usually represents the incremental increase in the cost function due to the incorporation of this element into the solution under construction. The evaluation of the elements by this function leads to the creation of a restricted candidate list (RCL) formed by the best elements, i.e. those whose incorporation to the current partial solution results in the smallest incremental costs (this is the greedy aspect of the algorithm). The element to be incorporated into the partial solution is randomly selected from those in the RCL (this is the probabilistic aspect of the heuristic). Once the selected element is incorporated to the partial solution, the candidate list is updated and the incremental costs are reevaluated (this is the adaptive aspect of the heuristic). This strategy is similar to the semi-greedy heuristic proposed by Hart and Shogan [22], which is also a multi-start approach based on greedy randomized constructions, but without local search.

```
procudure Greedy_Randomized_Construction(Seed)
1   Solution ← φ;
2   Evaluate the incremental costs of the candidate elements;
3   while Solution is not a complete solution do
4       Build the restricted candidate list(RCL);
5       Select an element s from the RCL at random;
6       Solution ← Solution ∪{s};
7       Re evaluate the incremental costs;
8   end;
9   return solution;
end greedy_Randomized_Constracti on.
```

Fig. 4. Pseudo-code of the construction phase

The solutions generated by a greedy randomized construction are not necessarily optimal, even with respect to simple neighborhoods. The local search phase usually improves the constructed solution. A local search algorithm works in an iterative fashion by successively replacing the current solution by a better solution in the neighborhood of the current solution. It terminates when no better solution is found in the neighborhood. The pseudo-code of a basic local search algorithm starting from the solution constructed in the first phase and using a neighborhood N is given in Fig. 5.

```
procudure Local_Search(Solution)
1  while Solution is not locally optimal do
2      Find s′∈N(solution) with f(s′) < s(Solution);
3      Solution ← s′;
4  end;
5  return Local_Search.
```

Fig. 5. Pseudo-code of the local search phase

The effectiveness of a local search procedure depends on several aspects, such as the neighborhood structure, the neighborhood search technique, the fast evaluation of the cost function of the neighbors, and the starting solution itself. The construction phase plays a very important role with respect to this last aspect, building high-quality starting solutions for the local search. Simple neighborhoods are usually used. The neighborhood search may be implemented using either a best improving or a first-improving strategy. In the case of the best-improving strategy, all neighbors are investigated and the current solution is replaced by the best neighbor. In the case of a first-improving strategy, the current solution moves to the first neighbor whose cost function value is smaller than that of the current solution. In practice, we observed on many applications that quite often both strategies lead to the same final solution, but in smaller computation times when the first-improving strategy is used. We also observed that premature convergence to a non-global local minimum is more likely to occur with a best-improving strategy.

IV. Computational experiments

Computational experiments performed within the reported research were devoted to comparing the efficiency of particular metaheuristic method, as well as to evaluating the quality of their solutions with regard to optimal schedules. Moreover, the analysis of the test results made it possible to determine some specific features of the approach proposed.

A. Time of computations

One of the most important factors for checking whether a method is appropriate or not is elapsed time for computations. As coding of trajectory method was

done in MATLAB software, based on some examples time of computations was recorded. As it's shown in table 1, number of iteration is a key factor in total time. Because of characteristics of GRASP, it is possible to select and decide about max no. of iterations. For more investigation, computation time of a 20 jobs problem is calculated and shown in table 1.

TABLE 1
Computation time as iteration increases in seconds.

No. of iteration(s)	time
1	1.68"
5	9.10"
10	15.95"
15	24.73"
20	32.06"

B. *Objective function value*

GRASP is known as one of optimization methods which presents optimum and near optimum solutions. In optimization methods, there is a trade of between solution quality and time of computations. Because of stochastic nature of this trajectory method, quality of solution is dependent on random seed number and usually is improved as iteration number is increased. This would let approximate low time of computations and near optimum solutions. An example is shown in table 2 about objective function of 20 jobs problem. Even though objective function value is improved as iterations increased but it's not a absolute because of probabilistic features.

TABLE 2
Objective function value as iteration increases

No. of iteration(s)	Objective function value
1	6924
5	6579
10	5789
15	5331
20	5033

C. *Comparison with exact method*

Checking metaheuristic solutions with exact method is a criterion for evaluating trajectory methods. Exact methods are time consuming but reliable. For more insurance, exact methods results are checked by applied metaheuristic solutions where number of jobs is increased. Ten test problems with different sizes are presented to evaluate the performance of the proposed algorithm. The proposed GRASP is coded in Matlab 7.0, and LINGO software is used to compare the results numerically. All the test problems are solved on a Pentium 4 computer with 512 MB RAM and 3.08 GHz CPU. Results are shown in table 3. A quality criterion, the gap of solution, is defined to show the percentage of deviation of the objective functions obtained by proposed GRASP from the values obtained by LINGO, according to the following equation:

$$Gap = \frac{(GRASP_{answer} - LINGO_{answer})}{LINGO_{answer}} \times 100$$

As the results show, the solution gaps vary from zero to 10.44% for different number of jobs. And since the run times for the GRASP algorithm are significantly lower than the LINGO computation times, using the GRASP algorithm with seems to be quite acceptable.

D. *Comparison with other heuristic methods*

In this section, three metaheuristic approaches are selected from Blazewicz et al. [16] paper and compared with the new GRASP method proposed in Section 3. They considered the two-machine flow shop problem with a common due date and the weighted late work performance measure and presented three trajectory methods, namely simulated annealing (SA), tabu search (TS) and variable neighborhood search (VNS) to solve the problem under consideration.

All of the four algorithms (*GRASP, SA, TS, VNS*) are coded in MATLAB 7.0 software and are compared using the earlier ten test problems. Table 4 shows the relative results.

As it is seen from Table 4, despite two first test problems for witch the methods have the same results, GRASP creates the best solutions in seven of the eight remaining test problems. These results can be considered as another proof for efficiency of the new proposed method.

Table 3
Evaluating GRASP solutions quality by comparison with exact method

Test Problem	*No. of jobs*	*Exact method*		*Grasp metaheuristic(100 iterations)*		*Gap (%)*
		time	*Obj. Fun. value*	*time*	*Obj. Fun. value*	
1	3	0.5"	4	3.40"	4	0
2	5	1.5"	14	4"	14	0
3	10	37"	97	12.8"	99	2.06
4	20	50"	121	18"	125	3.30
5	35	625"	153	23"	159	3.92
6	50	4120"	204	50"	215	5.39
7	80	13288"	312	134"	331	6.09
8	150	32497"	628	267"	679	8.12
9	300	[a)] 42953"	1383	349"	1522	10.05
10	500	[a)] 42721"	2137	535"	2360	10.44

a) Lower bound achieved by Lingo after about 12 hours

TABLE 4
Results of the four heuristic methods

Test Problem	*NO. of jobs*	*Obj. Fun. Value*			
		Grasp	*SA*	*TS*	*VNS*
1	3	4	4	4	4
2	5	14	14	14	14
3	10	99	102	104	102
4	20	125	123	127	127
5	35	159	161	176	181
6	50	215	227	243	237
7	80	331	346	352	336
8	150	679	694	723	735
9	300	1522	1537	1642	1629
10	500	2360	2675	2988	2940

V. CONCLUSION

The research presented completes the studies on the two-machine flow-shop problem with the common due date and the weighted late work criterion, $F2|d_j = d|Y_w$. The theoretical investigation resulted in the NP-completeness proof for its decision counterpart and the proposal of a pseudo-polynomial time dynamic programming approach that allowed us to classify the case as NP-hard in the ordinary sense [14]. A trajectory metaheuristic was compared in the computational experiments one to each other as well as to the list scheduling approach as a heuristic for $F2|dj = d|Y_w$ and to an exact enumerative method.

Appling developed solution with different problems solve that it presents near optimum quality while very low computation time is consumed. As exact methods and other heuristics are very time consuming in more than 10 jobs, GRASP gives capability of fast and near optimum till at least 500 jobs.

The experience gained during the research on problem $F2|d_j = d|Y_w$ provided many useful hints for future work on other scheduling problems with the late work performance measure.

REFERENCES

[1] J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt, J. Weglarz , ‘‘Scheduling computer and manufacturing processes’’, 2nd ed., Berlin, Heidelberg, NewYork: Springer, 2001.

[2] P. Brucker, Scheduling algorithms. 2nd ed., Berlin, Heidelberg, NewYork: Springer; 1998.

[3] B. Chen, CN. Potts, GJ. Woeginger, A review of machine scheduling. In: Du D-Z, Pardalos PM, editors. Handbook of combinatorial optimization. Boston: Kluwer Academic Publishers; 1998.

[4] M Pinedo, X. Chao, Operation scheduling with applications in manufacturing and services. Boston: Irwin/McGraw-Hill; 1999.

[5] J. Blazewicz, ‘‘Scheduling preemptible tasks on parallel processors with information loss’’, Technique et Science Informatiques 1984; 3(6), pp. 415–20.

[6] J. Blazewicz, G. Finke, Minimizing mean weighted execution time loss on identical and uniform processors. Information Processing Letters, (24) 1987, pp. 259–63.

[7] A.M.A. Hariri, C.N. Potts, L.N. Van Wassenhove, ‘‘Single machine scheduling to minimize total late work’’. *INFORMS Journal on Computing*, (7), 1995, pp. 232–42.

[8] R.B. Kethley, B. Alidaee, ‘‘Single machine scheduling to minimize total late work: a comparison of scheduling rules and search algorithms’’, *Computers and Industrial Engineering (*43), 2002, pp. 509–28.

[9] M.Y. Kovalyov, C.N. Potts, L.N. Van Wassenhove, ‘‘A fully polynomial approximation scheme for scheduling a single machine to minimize total weighted late work’’, *Mathematics of Operations Research* 19(1), 1994, pp. 86–93.

[10] J.Y.T Leung, ‘‘Minimizing the weighted number of tardy task units’’, *Discrete Applied Mathematics* (51), 1994.

[11] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, “Total late work criteria for shop scheduling problems’’, In: Inderfurth K, Schwoediauer G, Domschke W, Juhnke F, Kleinschmidt P,Waescher G, editors. *Operations Research Proceedings* 1999. Berlin: Springer; 2000. pp. 354–9.

[12] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, “Revenue management in a job-shop: a dynamic programming approach’’, Preprint Nr. 40/03, FMA, Otto-von-Guericke-University Magdeburg; 2003.

[13] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, ‘‘Open shop scheduling problems with late work criteria’’, *Discrete Applied Mathematics,* 2004 (134), pp. 1–24.

[14] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, “The two-machine flow-shop problem with weighted late work criterion and common due date’’, *European Journal of Operational Research* 2005;165(2): pp. 408–15.

[15] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, ‘‘A comparison of solution procedures for two-machine flow shop scheduling with late work criterion’’, *Computers and Industrial Engineering* 2005;49(4).

[16] J. Blazewicz, E. Pesch, M. Sterna, F. Werner, “Metaheuristic approaches for the two-machine flow-shop problem with weighted late work criterion and common due date”, *Computers & Operations Research* Volume 35, Issue 2, 2008, pp. 574-599.

[17] Sterna M. Problems and algorithms in non-classical shop scheduling. Poznan: Scientific Publishers of the Polish Academy of Sciences, Poland;2000.

[18] M.R. Garey, D.S. Johnson, “Computers and ntractability”, W.H. Freeman and Co., San Francisco, 1979.

[19] T.A. Feo and M.G.C. Resende, “ A probabilistic heuristic for a computationally difficult set covering problem”, *Operation Research Letters,* 8, pp. 67-71.

[20] T.A. Feo and M.G.C. Resende, “ Greedy randomized adaptive search procedures”, *Journal of Global Optimization,* 6, 1995, pp. 109-133.

[21] P. Festa and M.G.C. Resende, GRASP: a annotated bibliography. inC.C. Riberiro and p. Hansen, *Essays and Surveys in Metaheuristics,* Kluwer Academic Publishers, 2002, pp. 325-367.

[22] J.P. Hart and A.W. Shogan, “Semi-greedy heuristics: an empirical study”, *Operation Research Letters,* 6, 1987, pp. 107-114.