Journal of Scientific & Industrial Research
Vol. 79, May 2020, pp. 377-382

A Single Machine Scheduling Problem with Individual Job Tardiness based Objectives

Atif Shahzad[1]*, Waqar Ahmed Gulzar[1] and Aeysha Shahzad[2]

[1]Faculty of Engineering, King Abdulaziz University, Jeddah, Saudi Arabia
[2]Independent Researcher

Received 30 April 2019; revised 21 November 2019; accepted 21 March 2020

A multi-objective scheduling problem with specified release times and due dates for individual tasks is analysed in this study. Distinct tardiness value of each task *j* comprises the part of the objective, while it is desired to identify all non-dominated solutions. Tardiness values for a total number of *n* tasks complete a single solution making it an *n*-objective scheduling problem. Tardiness is treated here as a task specific objective, being different in the usual scheduling context. A branch and bound procedure is proposed for individual tardiness of tasks in multi-objective contexts. The procedure is illustrated with an example. Active schedule enumeration scheme with depth-first strategy for branching is used in branching while two different bounding schemes are tested. However, an improved bounding scheme to find better-quality need to be developed. Procedure is found to perform well on small scale problems. For an *n*-objective problem like this, a more robust data structure may further improve the performance of the procedure.

Keywords: Group scheduling, Multi-objective optimization, Job shop, Branch and bound, Dominance rule

Introduction

Scheduling is concerned with a set of scarce resources to dynamically allocate to competing tasks. In manufacturing context, these resourcing are usually machines on the shop floor. At the shop floor, these tasks are the jobs waiting for the machines within a dynamically changing environment under various constraints and with certain desired objectives. While allocating a task to a resource at a specific time slot, one is actually identifying the optimal or near-optimal start time for the processing of each task. The issue of assigning an individual task to a machine does not arise for the case of a single machine problem. Each task is to be processed on the same machine and it is desired to sequence the tasks and to find an optimal starting time for each task with desired objectives.

In the case of single machine problem, a set of competing tasks (J) is to be scheduled on one machine. Each of these tasks has certain attributes that need to be considered for their potential place in the desired schedule. Among these attributes, release date, processing time and due date are primitive attributes. Release date of a task $j \in$ J is noted as r_j and reflects the time for the availability of task for the machine, to consider it as a potential candidate for scheduling. Processing time p_j of a task j is the maximum time (value-added) that can be spent by that task on machine in an uninterrupted manner. Due date d_j of a task j is a limiting value on the finish time or else considered to be tardy and has associated cost (non-value-added). Completion time C_j of a task j is the actual time that tasks finishes its processing and releases the machine. These definitions are implied for single machine scheduling in a non-preemptive processing environment.

With these definitions and the desired objectives, we have a $1|r_j|\#\{T_j\}$ problem, hereinafter referred as P. P is a single machine *n*-objective combinatorial optimization problem where *n* is the total number of tasks (n =|J|). P can be deterministic in nature or stochastic. In the deterministic case, processing time and the release date of each task is fixed. If all of this information is known apriori, it is static in nature. In our study, we speak of a problem that is dynamic and stochastic in nature. This means that schedule is recalculated and processed dynamically while the new tasks are appearing in the system at random intervals of time with initially unknown processing times. The randomness in these processes follows certain playability laws.

Solving shop scheduling problems have remained active area among researchers for many reasons. This

*[1]Author for Correspondence Email: mmoshtaq@kau.edu.sa
[2]Author contributed for part of the work while she was affiliated with Univesrité de Nantes, France. (http://oro.univ-nantes.fr/)

interest has led to a plethora of methods with achievements of various degrees. An exact processing order of the tasks is achieved by gradually building up the schedule in a constructive algorithm. This uses simple rules and procedures. Priority dispatching rules are constructive in nature. These have found an extensive use in industry, primarily because of inherently reduced computational complexity, implementation ease and transparency in its processing. These are able to find an optimal solution for a few of single machine problems. Lawler's algorithm and Moore's algorithm are some classical scheduling algorithms for single machine problem. Elimination of non-optimal schedules from a list of all possible schedules prepared by enumerating is the other way of finding optimal or near optimal solutions. These are termed as enumeration methods such as dynamic programming and branch and bound method.

One of the approaches used to limit the number of possible solutions is to use dominance rule. This acts as an additional constraint to the initial problem. However, it does not have any effect on the value of the optimum. Emmons[1] has listed some robust dominance rules. These rules form the basis of many of the exact methods that are used for solving $1||\Sigma_j T_j$ problem. Chu & Portmann[2] defined a dominant subset of schedules for $1|r_j|\Sigma_j T_j$. Using these, many approximate scheduling algorithms were proposed. Using a new decomposition rule, Szwarc & Mukhopadhyay[3] devised a branch and bound algorithm for $1||\Sigma_j T_j$. Later, Szwarc *et al.*[4,5] worked out on an algorithm of enhanced performance by analyzing the impacts of deletion of lower bounds. Decision theory can also be employed to find the impact of preferring one task to schedule ahead of others[6]. They used stronger decomposition rules resulting in an improved performance of the early algorithm. Baptiste *et al.*[7] worked on branch and bound procedure by generalizing the set of dominance rules and found much improved lower bounds for $1|r_j|\Sigma_j T_j$. Loukil *et al.*[8] provided a literature review on single machine scheduling problem. They proposed a multi-objective method based on simulated annealing.[9]

In literature on single machine scheduling, the method that is most widely used to solve the case of multi-objective is ε-constraint method.[10] This method employs the minimization of one of the objectives while keeping an upper bound constraint on each of the other objectives. Nelson *et al.*[11] and Pinedo & Chao[12] made use of dominance rules to develop a branch and bound algorithm. They identified some non-dominated schedules for the same problem using these dominance rules. By using a similar branch and bound algorithm, Nelson *et al.*[11] enumerated a Pareto optimal for $1||E(C/U)$. Later, Kiran & Unal[13] extended this work by providing some general conditions for these optima. Lin[14] proposed a posteriori algorithm for $1||\#(C, T)$, which relied on principles of dynamic programming. They included some new dominance rules in their proposed algorithm.

Branch and bound procedure

Among the exact approaches used to solve scheduling problems, one of the most notable methods is the branch and bound method[15]. Various enumeration strategies are adopted in Branch and Bound (B & B) algorithms in order to dynamically construct a tree structure of schedules. This tree structure represents a solution space of all viable sequences. Search of the desired solution(s) is guided by repetitive branching and various bounding schemes. Topmost node of the tree structure represents the root problem, which is the original problem with the complete feasible region. This is taken as the starting point of the solving process for the root problem.

The branching procedure where problem is split into two or more sub-problems is illustrated in Fig 1. The union of these sub-problems is always the parent node problem. At different levels of the search tree, a node represents a partial solution of the node problem at that level. Recursively, the algorithm is applied to the sub-problems at different levels. Progress of the search is determined from an active node, which is unselected yet. In this way, subsequent set of nodes are determined. To compute the lower and upper bounds, bound procedures are used. The quality of the best schedule found during the search is represented by the upper bound. On the other hand, best possible quality at a given node is reflected by the lower bound at that node. In solving combinatorial optimization problems, where the search space is exponentially huge, these bounds play a vital role to limit the search space and guide the direction of the search in fruitful area of the search space. Initiating from the root node, for each of the branched sub-problem, procedures of lower-bounding and upper-bounding are applied iteratively. Limiting of the search space is obtained by discarding a node (and hence all sub-problem emanating from that node), for which the lower bound

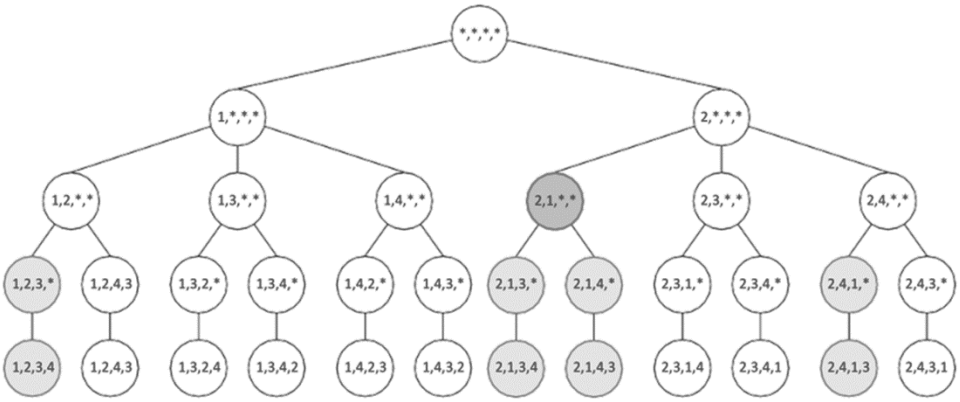

Fig. 1 — Multi-objective branch and bound tree of active schedules

exceeds the best-the known value of the best feasible solution. A local optimal may be found in the sub-space feasible region of that node but no globally optimal solution can exist in that subspace. This iterative solving, bounding and pruning process continue till all the nodes are exhausted. Finding an optimal solution for a sub-problem, merely represents one of the many feasible solutions for the root problem and not necessarily a globally optimal solution for the root problem.

The procedure of branch and bound can easily be applied in the case of a single objective optimization problem. In this process, bounds can be computed by applying certain relaxations to the original problem. For instance, construction of a branch and bound procedure for $1|r_j|L_{max}$ may be as follows. First check the eligibility of a task for a particular position is checked in the branching procedure. For this, let task c is considered as a candidate for position k. This is possible if and only if $r_c<\min(\max(t, r_l) + p_l)$., where $J_l \in J$. J represent the set of tasks that are not yet scheduled while t denotes the completion time of the previous task on the machine. Pruning of the node from the tree is applied, if any job task does not satisfy this inequality.

Applying preemptive EDD (Earliest Due Date) rule is one of the many possible relaxations that can be applied to compute the bounds. It is known that the preemptive EDD rule is able to find an optimal schedule for $1|r_j, prmp|L_{\max}$. This problem represents a relaxation of $1|r_j|L_{max}$. Lower bound is calculated at each node and if the lower bound for a node exceeds the upper bound (found previously), then no further branching of this node is considered.[11]

In contrast to single objective scheduling problems, where one optimal schedule is to be found, multi-objective branch-and-bound scheduling procedures look for the Pareto front of schedules (in fact, for each Pareto point in the objective space, one schedule is to be found). Therefore, instead of a single schedule, a set of non-dominated schedules found are kept at each step. Furthermore, from a particular node in the search tree, it is possible that many Pareto optimal schedules, reachable from that node, are existent unlike the single-objective case. Hence, a set of lower bounds is associated with a single node instead of a single bound. The generalization of the concepts of bounds is the bound sets, that can be adapted in multi-objective optimization problems.

For the success of any branch and bound procedure, quality of bounds is vital. For the case of multi-objective optimization problems, ideal point y^I and nadir point y^N, with $y^I<y<y^N$ are well-known. Ideal point represents a lower bound. Nadir point y^N represents the upper bound on the value of any efficient point. However, it is unfortunate that these bounds are not very effective, being quite distant from the non-dominated schedules. Applying the concept of bound sets, local ideal points represent a set of lower bounds while the local nadir points represent the set of upper bounds. One can derive these local ideal and local nadir points from supported solutions adjacent to each other in the objective space. Now the application of this branch and bound procedure on P is presented in the next section.

Individual Task Tardiness Enumeration

With the execution of n tasks on a single machine and having the objective to minimize the tardiness of each task independently, one comes across an enumerative n-objective optimization problem. However, there is a strong coupling among the individual task tardiness values. Enumerating these tardiness values T_j for all j tasks as performance

measure makes the problem as dynamic *n*-objective optimization problem. Note that, by setting $\forall j$, $d_j = 0$, $1|r_j|\#\{C_j\}$ becomes a special case of individual task-tardiness-enumeration problem. As a generic notation for single machine problem, $1||\#(f_1, ..., f_K)$, any task related objective is enumerated to find a non-dominated solution. In order to construct pareto optima, the algorithm needs to enumerate all the solutions. It is evident from the notation representing a posteriori resolution contexts.[16] As the number of objectives are dynamic according to the number of tasks appearing in the system, enumerative notation of $\#\{T_j\}$ is employed. This notation is representing the individual task tardiness as part of the objective. This means that the concept of dominance of the schedules is to be used here. We find the set of all non-dominated schedules of this scheduling problem. To find these non-dominated schedules for P, we used a branch and bound procedure. For the enumeration and branching, we have used active schedule generation procedure with depth first strategy for generation of the active nodes.

We have employed two bounding schemes. As a node represents a partial schedule, we solve instances of the problem $1|r_j, prmp|\#T_j$ at each node in the first bounding scheme. We obtain the set of lower bounds for the original problem as the tardiness for all *n* tasks. If the tardiness value is not dominated the prior lower bound set, we can prune the node. Here we intrinsically assume that the corresponding sub-problem of a node having its dominated schedules are themselves dominated. Local ideal points are used in the other bounding scheme. We illustrate the procedure in Fig.1 for a $1|r_j|\#\{T_j\}$ problem listed in Table 1. At each stage during the execution of the procedure, all the partial and complete schedules are listed in Table 2. Two possible branches (active) can be seen at the root node (*,*,*,*). At this node, both the branches are explored because we have a non-dominated schedule with a bound of (0,5,4,0). Despite the fact that the nodes 20–23 are not explored by the procedure (as their root node 19 is dominated), in Table 2, we have listed these nodes for the sake of clarity. This is evident from Fig. 1 as well.

Table 1 — A 1|rj|# {Tj} problem P

j	r_j	p_j	d_j
1	0	4	8
2	1	2	12
3	3	6	11
4	5	5	15

Subsequently, for the problem in Table 1, all the non-delay schedules are given in Fig. 2. All the active, but not non-delay, schedules are listed in Fig. 3. A total of 12 schedules are identified as active whereas a total of 8 schedules are non-dominated.

Discussion

Though seems surprising, but not unusual that in practice, the single machine problem arises quite frequently. For example, an obvious one is a single processor non-time-sharing computer for processing of tasks awaiting service. Then, we have other instances of its application, where we can split large problems in complex plants to act as single machine

Table 2 — Node listing for branch and bound procedure

Node number	Node	Number of branches	Tardiness	Dominated?
1	(*,*,*,*)	2	(0,5,4,0)	N
2	(1,*,*,*)	3	(0,5,4,0)	N
3	(1,2,*,*)	2	(0,0,6,1)	N
4	(1,2,3,*)	1	(0,0,1,7)	D
5	(1,2,3,4)	0	(0,0,1,7)	D
6	(1,2,4,*)	1	(0,0,6,1)	N
7	(1,2,4,3)	0	(0,0,6,1)	N
8	(1,3,*,*)	2	(0,5,0,5)	N
9	(1,3,2,*)	1	(0,0,0,7)	N
10	(1,3,2,4)	0	(0,0,0,7)	N
11	(1,3,4,*)	1	(0,5,0,5)	N
12	(1,3,4,2)	0	(0,5,0,5)	N
13	(1,4,*,*)	2	(0,6,5,0)	N
14	(1,4,2,*)	1	(0,0,7,0)	N
15	(1,4,2,3)	0	(0,0,7,0)	N
16	(1,4,3,*)	1	(0,6,5,0)	N
17	(1,4,3,2)	0	(0,6,5,0)	N
18	(2,*,*,*)	3	(0,0,7,2)	N
19	(2,1,*,*)	2	(0,0,7,2)	D
20	(2,1,3,*)	1	(0,0,2,8)	D
21	(2,1,3,4)	0	(0,0,2,8)	D
22	(2,1,4,*)	1	(0,0,7,2)	D
23	(2,1,4,3)	0	(0,0,7,2)	D
24	(2,3,*,*)	2	(5,0,0,8)	N
25	(2,3,1,*)	1	(5,0,0,8)	N
26	(2,3,1,4)	0	(5,0,0,8)	N
27	(2,3,4,*)	1	(10,0,0,4)	N
28	(2,3,4,1)	0	(10,0,0,4)	N
29	(2,4,*,*)	2	(6,0,9,0)	N
30	(2,4,1,*)	1	(6,0,9,0	D
31	(2,4,1,3)	0	(6,0,9,0)	D
32	(2,4,3,*)	1	(12,0,5,0)	N
33	(2,4,3,1)	0	(12,0,5,0)	N

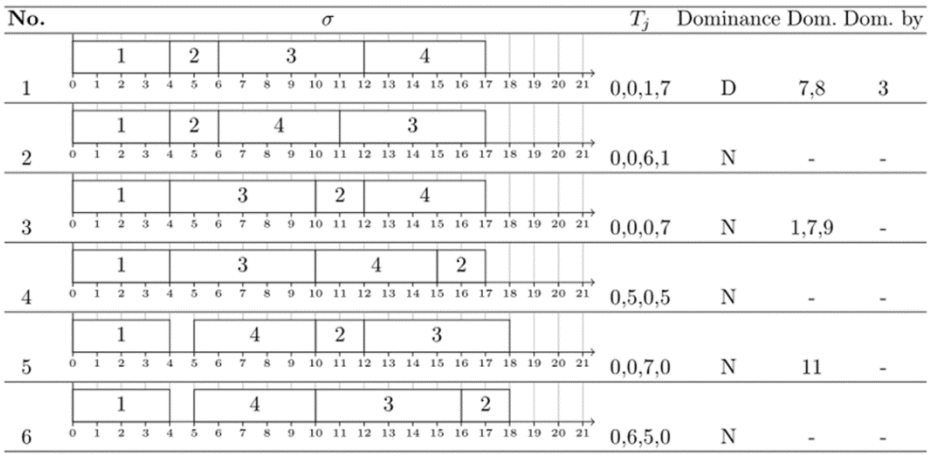

Fig. 2 — Non Delay Schedules for Problem P

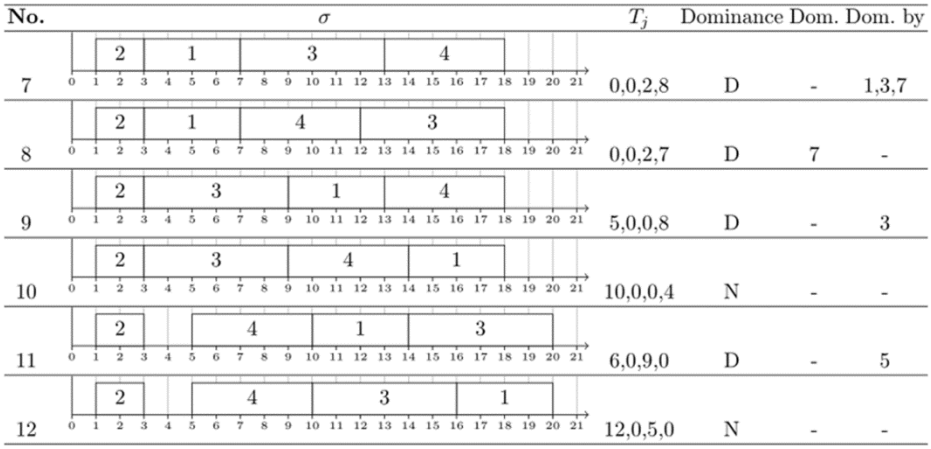

Fig. 3 — Active Schedules for Problem P

problem. For instance, to make a single colour in paint manufacture, the entire plant may be employed at a time. Finally, we have a bottleneck machine in a multi-machine complex, where the concept of single machine is used to decompose a large problem. Resolution of relatively complex scheduling problems is often achieved by the study of such a kind of single machine problems. Hence this can be treated as a relaxed version of a complex problem, where relaxation is a method in which a strict requirement imposed on the problem is temporarily removed, by either substituting for it another more easily handled requirement or else dropping it completely.

Conclusions

In this work, we present an n-objective enumerative scheduling problem represented as $1|r_j|\#\{T_j\}$. Individual task tardiness makes part of the objective value. We have proposed a branch and bound procedure where active schedule generation is used for the enumeration during branching. We used depth first strategy for the tree exploration and two distinct bounding schemes to find lower bound set. This study has the main perspective to reduce the search space by finding a set of dominance rules in the case of a single machine problem. However, in order to enhance the performance of the procedure even for the case of a single machine with n-objectives.

References

1 Emmons H, One-Machine Sequencing to Minimize Certain Functions of Job Tardiness, *Oper Res*, **17**(**4**) (1969) 701–715.

2 Chu C & Portmann M C, Some new efficient methods to solve the n/1/ri/[epsilon]Ti scheduling problem, *Eur J Oper Res*, **58**(**3**) (1992) 404–413.

3 Szwarc W & Mukhopadhyay S K, Decomposition of the single machine total tardiness problem, *Oper Res Lett*, **19**(**5**) (1996) 243–250.

4 Szwarc W, Della Croce F & Grosso A, Solution of the single machine total tardiness problem, *J Sched*, **2**(**2**) (1999) 55–71.

5 Szwarc W, Grosso A & Della Croce F, Algorithmic paradoxes of the single-machine total tardiness problem, *J Sched*, **4**(**2**) (2001) 93–104.

6 Gahm C, Kanet J J & Tuma A, On the flexibility of a decision theory-based heuristic for single machine scheduling, *Comput Oper Res*, **101**(**1**) (2019) 103–115.

7 Baptiste P, Carlier J & Jouglet A, A Branch-and-Bound procedure to minimize total tardiness on one machine with arbitrary release dates, *Eur J Oper Res*, **158**(**3**) (2004) 595–608.

8 Loukil T, Teghem J & Tuyttens D, Solving multi-objective production scheduling problems using metaheuristics, *Eur J Oper Res*, **161**(**1**) (2005) 42–61.

9 Kolahan F & Sharifinya A, Simultaneous job scheduling and tool replacement based on tool reliability by proposed Tabu-SA algorithm, *J Sci Ind Res*, **68(6)** (2009) 496–504.

10 Parthiban P & Abdul Zubar H, An integrated multi-objective decision making process for the performance evaluation of the vendors, *Int J Prod Res*, **51**(**13**) (2013) 3836–3848.

11 Nelson R T, Sarin R K & Daniels R L, Scheduling with multiple performance measures: the one-machine case, *Manage Sci*, **32**(**4**) (1986) 464–479.

12 Pinedo M & Chao X, *Operations Scheduling with Applications in Manufacturing and Services*. Irwin/McGraw-Hill; 1999.

13 Kiran A S & Unal A T, A single-machine problem with multiple criteria, *Nav Res Logist*, **38**(**5**) (2006) 721–727.

14 Lin K S, Hybrid algorithm for sequencing with bicriteria, *J Optim Theory Appl*, **39**(**1**) (1983) 105–124.

15 Xu J, Lin W-C, Yin Y, Cheng Y & Wu C-C, A Two-Machine Flowshop Scheduling Problem with a Job Precedence Constraint to Minimize the Total Completion Time, *J Sci Ind Res*, **76**(**12**) (2017) 761–766.

16 T'kindt V & Billaut J C, *Multicriteria Scheduling: Theory, Models and Algorithms*. Springer Verlag; 2006.