A Single Machine Scheduling Problem with Individual Job Tardiness based Objectives, $1|r_j|\sharp\{T_j\}$

Aeysha SHAHZAD

LINA
Université de Nantes
2, rue de la Houssinière
BP 92208, 44322 Nantes Cedex 3, France
aeysha.atif@etu.univ-nantes.fr

Abstract. We consider a single machine multi-objective scheduling problem with each task j having a release date, r_j and a due date, d_j. The goal is to find a set of non-dominated solutions with individual tardiness of each task, T_j as the objective, which is a task specific objective. The problem is denoted as $1|r_j|\sharp\{T_j\}$. A branch and bound procedure is proposed.

1 Introduction

Scheduling concerns the allocation of limited resources (also known as machines) to the given set of tasks over time. Different components of a scheduling problem are the tasks, constraints, resources and the objective functions. Scheduling of tasks to resources is to find a processing start time for each task. For a single machine, there is only one resource available for the processing of tasks, there does not arise the question of assigning a task to resource and hence the goal is to determine a schedule of tasks.

A single machine scheduling problem consists of a set $\mathbb{J}$ of tasks, with each task $j \in \mathbb{J}$ having release dates r_j and due date d_j, that arrives on a single machine to be executed for a processing time p_j and completion time C_j. It is desired to sequence the jobs on a machine in a particular order to optimize certain objectives. This study focuses on the $1|r_j|\sharp\{T_j\}$ problem. This is an n-objective combinitorial optimization problem with n tasks to be executed on a single machine.

In the case where all the characteristics of the problem (processing time of each operation, release dates, *etc.*) are known, we speak of a deterministic problem. Conversely, some of these characteristics may be random variables of known probability law. In this case we speak of a stochastic problem. If all the data of the problem are known at the same time we speak of a static problem. For some problems, a schedule may have been calculated and being processed when new operations arrive in the system. Then the foregoing schedule has to be re-established in real time. These problems are said to be dynamic.

Scheduling Definitions

Definition 1. *A task j is a fundametal entity described in time domain by a start time and finish time, that is executed for a processing time on a certain resource (Squirrel & Lopez, 1999).*

A task is associated with it a release date, r_j, specifying the time of arrival of task in the scheduling system, a processing time, p_j, the time for which task executes on a machine, a due date, d_j, the date at which job is promised to be completed. Number of tasks in the problem is denoted as n reflecting the problem size, while number of tasks available for processing at any instant t, is denoted as n_t. A resource is required to perform tasks.

Definition 2. *A resource, $M_k \in \mathbb{M}$ is any physical or virtual entity of limited capacity and/or availibility, allocated to the tasks competing for it.*

The term machine is generally used instead of resource in shop scheduling literature. A resource can be classified as renewable and non-renewable. Renewable resources become available again after use, while non-renewable resources disappear after use (T'kindt & Billaut, 2006). Usually a limit on the capacity of the resources is the major constraint besides others, *i.e.* a machine cannot perform several tasks at the same time.

A constraint represents a condition which definitely must be respected. Constraints are not restricted to the limitations on resource, but to any decision variable involved in the problem. A constraint may be viewed as a resticted set of values that these decision varibles can take. For example, the occurrence of different release dates constitutes a constraint which must be stated precisely. A solution of a scheduling problem must always satisfy a certain number of constraints.

Definition 3. *A processing sequence (π) represents a permutation of the tasks competing for a resource, or an order in which tasks are to be processed on a given resource.*

A processing sequence therefore contains no explicit information about the times at which the various operations start and finish. Timetabling is refered as the process of deriving a processing schedule from a processing sequence (S.French, 1982).

Definition 4. *A schedule (σ) usually refers to an allocation of jobs within a more complicated setting of machines, allowing possibly for preemptions of tasks by other tasks that are released at later points in time (Pinedo & Chao, 1999).*

A feasible schedule is a schedule that satifies all the constraints of the subject scheduling problem. These constraints, generally refered as hard constraints must be staified or the schedule is infeasible. Similarly we speak of soft constraints, as the constraints that are desired to be satified, but do not cause the schedule to be out of feasible space, if unsatified. These are refered as objectives of the problem.

Definition 5. *An objective reflects the desired characteristics in the solution to find for a scheduling problem by executing a given set of tasks on the machines in a certain sequence.*

Different performance measures may be used for the evaluation of schedules in regards with the objectives under consideration. The objectives reflect the characterics desired in the final schedule. These may be based upon completion times, due dates or inventory, utilization costs, *etc*. Two performance measures are equivalent if a schedule which is optimal with respect to one is also optimal with respect to the other and vice versa.

The most studied objective related to completion times is minimizing the completion time of the entire schedule, known as makespan and denoted as $C_{\max}$, defined as:

$$C_{\max} = \max_{1 \leq j \leq n} C_j$$

The objective can also a function of the due dates. The lateness of job j is defined as $L_j = C_j - d_j$, which is positive when job j is completed late and negative when it is completed early. The maximum lateness, $L_{\max}$, is defined as:

$$L_{\max} = \max_{1 \leq j \leq n} L_j$$

One of the most important objectives to deal with in a manufacturing system is to minimize tardiness which can be measured through several performance measures. The tardiness of a job is computed as, $T_j = \max(0, L_j)$.

The most used performance measure to evaluate the tardiness is the mean tardiness (*i.e.* $\bar{T}=\Sigma_j\ T_j\ /n$). The maximum tardiness, *i.e.*

$$T_{\max} = \max_{1 \leq j \leq n} T_j,$$

can be of great interest for the decision-maker in the shop which is an indication of a worst case behavior during a particular experiment. The number of tardy jobs is defined as,

$$U_j = \begin{cases} 1 & \text{if } L_j > 0 \\ 0 & \text{otherwise} \end{cases}$$

The conditional mean tardiness described as

$$CMT = \frac{\sum_j T_j}{\sum_j U_j}$$

measures the average amount of tardiness for the completed jobs which are found to be tardy. The two large families of objectives are:

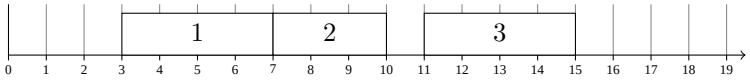
- minimax, which represent the maximum value of a set of functions to be minimised, and
- minisum, which represent a sum of functions to be minimised.

Table 1: Some Performance Measures

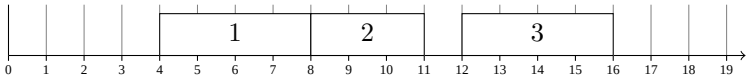
Name	**Notation**	**Regular**	**Type**
Makespan	$C_{\max}$	yes	minimax
Maximum lateness	$L_{\max}$	yes	minimax
Maximum tardiness	$T_{\max}$	yes	minimax
Total tardiness	$\Sigma_j\ T_j$	yes	minisum
Total weighted tardiness	$\sum_j w_j T_j$	yes	minisum
Mean tardiness	$\bar{T}$	yes	minisum
Conditional mean tardiness	CMT	no	minisum

Table 2: An example problem to illustrate Regular measure

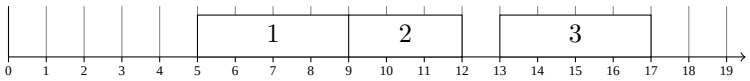
j	r_j	p_j	d_j
1	0	4	8
2	2	3	8
3	4	4	12



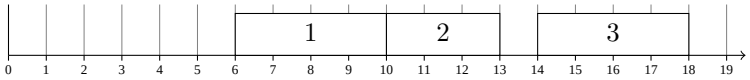
(a) σ_1: $C_{\max}$= 15, CMT= 2.5



(b) σ_2: $C_{\max}$= 16, CMT= 3.5



(c) σ_3: $C_{\max}$= 17, CMT= 3.33



(d) σ_4: $C_{\max}$= 18, CMT= 4.33

Fig. 1: Schedules for problem in table 2 illustrating performance measures

Performance measures can be classified into regular ones and those that are not.

Definition 6. *A regular measure of performance is one that is non-decreasing in the completion times $C_1, ..., C_n$ (S.French, 1982). Thus R is the function $C_1, C_2, ..., C_n$ such that*

$$C_1 \leq C_1', C_2 \leq C_2', ..., C_n \leq C_n' \text{ , together}$$
$$\Rightarrow R\{C_1, C_2, ..., C_n\} \leq R\{C_1', C_2', ..., C_n'\}.$$

Maximum completion time ($C_{\max}$) and mean tardiness ($\bar{T}$) are examples of regular measure whereas CMT is not a regular one. Table 1 lists a few performance measures.

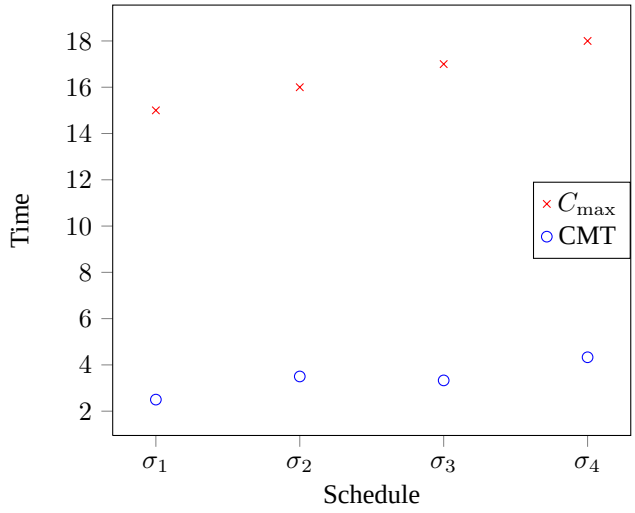


Fig. 2: Regular and Non Regular Measures

In Table 3, is given an example for illustrating the difference between regular and non-regular performance measures.

Four different solutions are illustrated in fig. 2 and the values of performance measures are illustrated in fig. 2 for problem of table 3. If we compare, for example, the σ_2 and σ_3 schedules, although $C_j(\sigma_2) \leq C_j(\sigma_3), \forall j$, the CMT obtained is less for this schedule, *i.e.*

$$C_j(\sigma_2) \leq C_j(\sigma_3), \forall j, \not\Rightarrow CMT(\sigma_2) \leq CMT(\sigma_3)$$

Hence CMT is a non-regular measure.

Multi-objective Optimization

Generally, single objective scheduling problems make the major part of the literature in machine scheduling domain. However, it is often desired to have multiple preferences for a solution to find. These preferences may lead to conflicting objectives. Thus maximizing one objective may degrade the other objectives to an unacceptable level. It is required to find a compromised solution satisfying these objectives simultaneously, although may not be optimal with respect to a single objective.
Multi-objective (MO) optimization is the process of simultaneously optimizing two or more objectives subject to certain constraints. A multi-objective optimization problem is defined by (Ehrgott & Gandibleux, 2000) as

$$\min f(x), x \in X, \text{ where } f(x) = [f_1(x), f_2(x), ..., f_p(x)]^T$$

with each $f(x) \in Y$, p being the number of objectives, X is the set of all possible solutions and Y is the objective space.

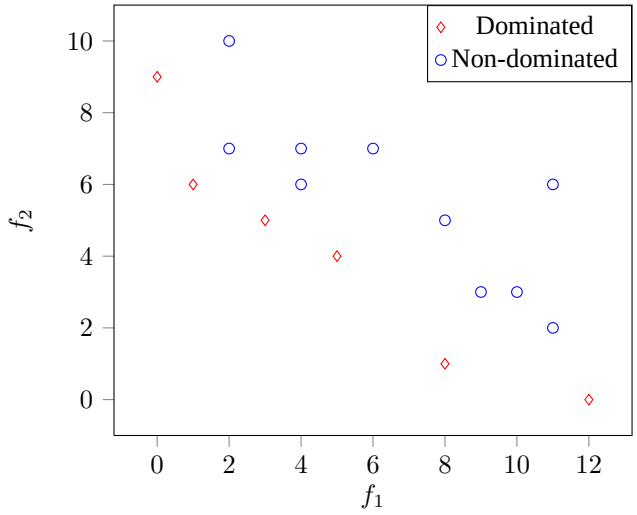


Fig. 3: A bi-objective space with dominated and non-dominated solutions

We speak of dominance of solutions as the way to compare the solutions to a multi-objective problem. If $x_1, x_2 \in X$ and $\forall k, f_k(x_1) \leq f_k(x_2)$, with at least one strict inequality, we say x_1 dominates x_2 ($x_1 \succ x_2$) and $f(x_1)$ dominates $f(x_2)$. A feasible solution $\hat{x} \in X$ is efficient or pareto optimal if there is no other $x \in X$ such that $\forall k, f_k(x) \leq f_k(\hat{x})$ with at least one strict inequality, *i.e.* there is no solution at least as good as x. As $\hat{x}$ is efficient then $f(\hat{x})$ is called non-dominated point (Ehrgott & Gandibleux, 2000). It is generally required to generate a set of efficient solutions. The set of all efficient solutions $\hat{x} \in X$ is denoted by X_E, the representation of X_E in objective space is called the efficient frontier or pareto front, and the set of all non-dominated points $\hat{y} = f(\hat{x}) \in Y$ by Y_E. Fig. 3 illustrates the difference between dominated and non-dominated solutions in a bi-objective space.

There are different approaches, found in literature to solve multi-objective optimization problem.

– **Weighted sum approach:** In this approach, a multi-objective problem is transformed into single objective problem by aggregating all the objectives. This single objective problem is then solved repeatedly with different parameter values. This is the most commonly used technique to find X_E. X_E can be partioned into supported efficient solutions and non-supported efficient solutions, X_{SE} and X_{NE} respectively. $x \in X_{SE}$ is an optimal solution of the following parametrized single objective problem for some $\lambda = (\lambda_1, \lambda_2, ..., \lambda_p), \forall k, \lambda_k > 0$:

$$\min_{x \in X} \sum_{k=1}^{p} \lambda_k f_k(x)$$

– **ϵ-constraint approach:** The ϵ-constraint method is another technique to solve multi-objective optimization problems that consists of minimizing one objective with an upper bound constraints on the other objectives.
– **Lexicographic approach:** In this approach, the objectives are ranked in a priority order and then optimized in that order without degrading the already optimized objectives.
– **Pareto approach:** This approach is aimed at generating the complete set of efficient solutions.

Classes of Schedules

Assumptions have to be made with regard to what the scheduler may and may not do when he generates a schedule. For example, it may be the case that a schedule may not have any unforced idleness on any machine. Two classes of schedules are of utmost importance in scheduling, namely active and semi-active.

A semi-active schedule is a left aligned schedule, it is a feasible schedule that can not be modified without changing the order of operations, without delaying some operations. In a semi-active schedule, it is not possible to finish a task earlier on a machine without reordering tasks.

A feasible schedule is active if no task can be processed earlier without delaying the starting time of another task. This means that the schedule is left-shift, such that there is not enough space available between any two tasks in time domain where some other task can be inserted without violating any constraints. To minimize a regular measure of performance it is only necessary to consider active schedules.

A schedule is called non-delay if no machine is kept idle while an operation is waiting for processing. This implies that if there is one or more tasks waiting for processing in front of a machine, it must start processing one of the waiting tasks as soon as it becomes free.

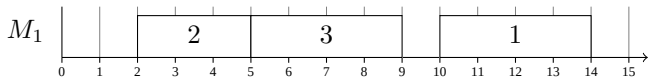
Consider an example of a single machine problem with three jobs given in table 3. Fig. 4 illustrates these classes of schedules for this problem.

Schedule σ_1 (fig. 4(a)) is not a semi-active schedule, because task 1 starts at $t = 10$, then what could start at $t = 9$ without violating any constraint, and without changing the execution order of jobs on the machine. Since this ordering is not semi-active, it is neither active nor non-delay.

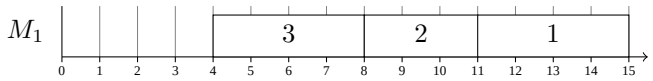
Schedule σ_2 (fig. 4(b)) is a semi-active schedule as no task can be slided towards its left without reordering the tasks. It is not an active schedule. Infact, if task 3 is at first position, the schedule can never be active as there will always be a room for task 1 before task 3. As it is not active, so it is also not non-delay.

Table 3: An example problem to illustrate classes of schedules

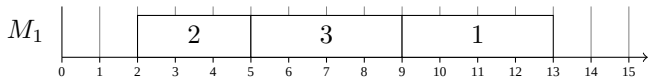
j	p_j	r_j
1	4	0
2	3	2
3	4	4



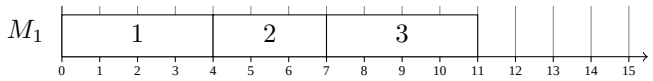
(a) σ_1: Non Semi-active schedule



(b) σ_2: Semi-active schedule



(c) σ_3: Active schedule



(d) σ_4: Non-Delay schedule

Fig. 4: Schedules of the problem described in table 3

Schedule σ_3 (fig. 4(c)) is an active schedule (and a semi-active as well) because no task can be shifted towards its left without delaying other tasks. It is however, not a non delay schedule as job 1 could have been processed on machine at $t = 0$, but machine is kept idle.

Schedule σ_4 (fig. 4(d)) is a non-delay (as well as active and semi-active) as machine is never kept idle when the task was available.

Classification of Scheduling Problems

There are some notations related to scheduling. This notation is decomposed of three field α, β and γ. The scheduling problem is denoted by 3-field notation: $\alpha|\beta|\gamma$.

α field The α field describes the machine environment. It consists of α_1 and α_2 where α_1 is the type of shop and α_2 is the number of machines in the problem and is optional. If this field is empty, then the number of machines is the data of the problem.
Some machine environment in the α field are:

Table 4: Some examples for the field α_1

α_1	**Description**
ϕ	Single machine, the value for the field α is "1"
P	Identical parallel machines
Q	Parallel machines with different speeds
R	Unrelated parallel machines
F	Flow shop
J	Job shop
O	Open shop

Table 5: Some examples for the field β

β	**Description**
$\tilde{d_j}$	The deadline
$prmp$	The preemption constraints
$prec$	The precedence constraints
r_j	the release date

- Single Machine: Only a single machine is available for processing. It is the simplest case of all scheduling environment.
- Identical Parallel Machines: There are m identical machines in parallel. Job j requires only one operation and may be processed on any one of the m machines.
- Parallel Machines with Different Speeds: There are m machines in parallel with different speeds. This environment is referred to as uniform machines.
- Unrelated Parallel Machines: More generalized case of above, where there are m different machines in parallel with speed of machines related to the jobs being processed on them.
- Flow Shop: There are m machines in series. Each job has to processed on each machine with each job follow the same route.
- Job Shop: In a job shop, each job has its own predetermined route to follow. A route may not go through all machines.
- Open Shop: Jobs has to be processed with no restrictions with regard to the routing of each job through the machine environment.

Some notations of α field are given in table 4.

β field The β field describes the constraints and particularities of the problem.

- Deadline: There are those due dates which are imperatives, and called deadlines. A deadline is a due date for which tardiness is not allowed (T'kindt & Billaut, 2006).
- Preemption: Processing of a job on a machine may be interrupted and resumed at a later time even on a different machine and already processed amount is not lost.
- Precedence: One or more jobs have to be completed before another job is allowed to start.
- Release Dates: The release date of job j, is the time when a job is arrived at the system to be processed and the job cannot start its processing before release date.

Some possible entries of β field and their notations are given in table 5.

Table 6a: Some examples for the field γ

γ	**Description**
$C_{\max}$	Makespan
$L_{\max}$	Maximum lateness
$T_{\max}$	Maximum tardiness
$\Sigma_j T_j$	Total tardiness
$\bar{T}$	Mean tardiness

Table 6b: Some MO examples for the field γ

γ	**Description**
$\sharp(f_1, ..., f_K)$	Enumeration of non-dominated solutions
$Lex(f_1, ..., f_K)$	Lexicographic optimisation of K objectives
$Fl(f_1, ..., f_K)$	Convex combination of K objectives
$\epsilon(f_1, ..., f_K)$	ϵ-constraint method of K objectives

γ field The γ field describes the objective function. For example, in a single machine problem with unequal release dates, problem can be denoted as $1|r_j\ |f$, with $f = \Sigma_j T_j$, ΣU_j, $T_{\max}$, *etc*. Some are given in table 6a. For multiobjective problems, list of objectives separated by comma are used in γ field. For instance, a single machine problem with $C_{\max}$ and $\bar{T}$ can be noted as $1||C_{\max}$,$\bar{T}$. Moreover, based on the method to compute pareto optimum, different values for the γ field can be introduced, as shown in table 6b.

2 Literature Review

This section addresses, the relevant single machine problems studied in literature. The approach used in the literature for solving problems with different single and multi-objective one machine scheduling problems are discussed. A review of the dominance rules for single machine problems is also presented.

A large number of approaches have been used for solving shop scheduling problem with varying degree of success. Constructive algorithm is that which buildup a solution from the data of the problem by following a simple set of rules which exactly determines the processing order. Priority dispatching rules (like SPT and EDD) fall in this category. Due to their simplicity, ease of implementaion and minimal computational complexity, they are very often used in industrial problemes. A very few of them provide optimal solution for certain specific single machine problems. Some classical scheduling algorithms for single machine include Lawler's algorithm for $1|prec|f_{\max}$ and Moore's algorithm for $1|d_j|\Sigma U_j$. Enumeration methods list or enumerate a given set of possible schedules and then eliminate the non-optimal schedules from the list. These techniques include dynamic programming and branch and bound method.

Single machine with tardiness problem A dominance rule is a constraint that can be added to the initial problem without changing the value of the optimum. (Emmons, 1969) introduced some powerful dominance rules, on which most of the exact methods

rely for solving $1||\Sigma_j T_j$ problem. This problem is shown to be NP-hard by (Du & Leung, 1990). (Chu & Portmann, 1992) defined a dominant subset of schedules for $1|r_j|\Sigma_j T_j$ and proposed several approximate algorithms belonging to this subset. (Akturk & Yildirim, 1998) proposed a dominance rule that provides a sufficient conditions for local optimality for the $1||\Sigma_j\ w_j T_j$ problem. Then they extended these dominance rules for $1|r_j|\Sigma_j\ w_j T_j$ problem and incorporated them in a branch and bound algorithm (Akturk & Ozdemir, 2000) and two local search algorithms (Akturk & Ozdemir, 2001). (Szwarc & Mukhopadhyay, 1996) developed a branch and bound algorithm for $1||\Sigma_j T_j$ based on their new decomposition rule. Then they reported in (Szwarc *et al.*, 1999) and (Szwarc *et al.*, 2001) some improved performance of their algorithm by considering the impact of deleting lower bound and adding a stronger decomposition rule. (Baptiste *et al.*, 2004) generalized these dominance rules and introduced new lower bounds for branch and bound procedure for $1|r_j|\Sigma_j\ T_j$. (Su & Chen, 2008) has used dominance properties to develop a branch and bound method for $1|r_j|\Sigma_j\ T_j$. (Loukil *et al.*, 2005) reviewed literature for a multi-objective single machine scheduling problem and proposed a simulated annealing based method.

Multi-objective single machine problem Most of the multi-objective single machine scheduling problems found in literature are infact ϵ-constraint problems, where one

Table 7: A brief review of multi-objective single machine problems

Problem	**Reference**	**Complexity**
$1\|\|\epsilon(f_{1_{\max}}/f_{2_{\max}}, ..., f_{K_{\max}})$	(Hoogeveen, 1992)	$O(n^4)$ for k = 2 $O(n^{k(k+1)-6})$ otherwise
$1\|d_j\|\epsilon(\bar{C}/L_{\max})$	(Smith, 1956) (Heck & Roberts, 1972) (VanWassenhove & Gelders, 1980) (Nelson *et al.*, 1986) (Esswein *et al.*, 2001)	$O(nlog(n))$ $O(n^2\overline{p}log(n))$
$1\|\|F_l(T_{\max}, \bar{C})$	(Sen & Gupta, 1983)	
$1\|\|Lex(f_{\max}, C)$ $1\|\|\epsilon(\bar{C}/f_{\max})$	(Emmons, 1975a) (John, 1989) (Hoogeveen & Van de Velde, 1995)	 $O(nlog(n))$
$1\|\|Lex(\bar{U}, \bar{C})$ $1\|\|\epsilon(\bar{C}/\bar{U})$	(Emmons, 1975b) (Nelson *et al.*, 1986) (Kiran & Unal, 2006)	 NP-hard NP-hard
$1\|\|\sharp(\bar{C}, \bar{T})$	(Lin, 1983)	NP-hard
$1\|\|\epsilon(C/T, T_{\max})$	(Nelson *et al.*, 1986)	NP-hard

objective is minimized while upper bound constraints are imposed on other objectives. Table 7 lists a review of relevant multi-objective single machine problems.

(Smith, 1956) studied a particular case of $1|d_j|\epsilon(\bar{C}/L_{\max})$ problem where a condition of $L_{max} = 0$ is imposed. (Heck & Roberts, 1972) and (VanWassenhove & Gelders, 1980) extended this problem by proposing a priori and aposteriori algorithm respectively. (Nelson *et al.*, 1986) proposed a branch and bound algorithm for the same problem by making use of dominance rules to identify a subset of non-dominated schedules. (Sen & Gupta, 1983) proposed a branch and bound procedure for $1||F_l(T_{\max}, \bar{C})$ problem.

(Emmons, 1975a) studied the generalized problem $1||Lex(f_{\max}, \bar{C})$ and proposed a greedy algorithm. Later, (John, 1989) and (Hoogeveen & Van de Velde, 1995) provided some improved results for $1||\epsilon(\bar{C}/f_{\max})$ problem.

(Hoogeveen, 1992) studied the problem of minimizing the K increasing functions of the completion times with two cases of $K = 2$ and $K > 2$ and specified the cardinality of the set of efficient solutions.

In the class of NP hard problems, (Emmons, 1975b) proposed a branch and bound algorithm for $1||Lex(\bar{U}, \bar{C})$ problem and provided some dominance conditions inspired by the Moore algorithm. (Nelson *et al.*, 1986) provided the enumeration of Pareto optima for $1||\epsilon(\bar{C}/\bar{U})$ employing a branch and bound algoritm. Later, (Kiran & Unal, 2006) extended the study on same problem and proposed some general conditions for these optima. $1||\sharp(\bar{C}, \bar{T})$ was studied by (Lin, 1983), who proposed a posteriori algorithm based on dynamic programming and integerated in it some new dominance conditions for the problem.

(Nelson *et al.*, 1986) was interested in $1||\epsilon(\bar{C}/\bar{T}, T_{\max})$ problem, for which they determined a subset of non-dominated schedules using branch and bound algorithm.

3 Branch and Bound Procedure

Branch and Bound methods have been the most successful of the exact approaches for solving scheduling problems. Branch and Bound (B & B) algorithms are enumeration schemes that use a dynamically constructed tree structure as a means of representing the solution space of all feasible sequences. As implied by their name a branching as well as a bounding scheme is applied to perform the search, starting by considering the topmost node of the search tree representing the root problem (the original problem with the complete feasible region).

Branching The branching procedure describes how to split a problem into two or more subproblems (subsets of the problem) such that their union returns the main problem as illustrated in fig. 5.

Each node at different levels of the search tree represents a partial solution of the problem. The algorithm is applied recursively to these subproblems. From an unselected (active) node the branching operation determines the next set of possible nodes from which the search could progress.

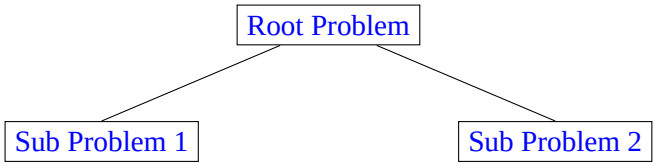


Fig. 5: Branching Procedure

Bounding Bounding procedure is aimed at computing the lower and upper bounds. The upper bound gives the quality of the best solution found during the search while the lower bound represens the best possible quality to find, at a given node. These bounds are an essential tool in a branch and bound procedure for solving combinatorial optimization problem. The lower and upper bounds are used to reduce the search space. The lower-bounding and upper-bounding procedures are applied, starting from the root problem to every subproblem obtained through branching process. If the lower bound for a node exceeds the best known feasible solution, no globally optimal solution can exist in the subspace of the feasible region represented by that node. Therefore, the node can be removed from further consideration. The search proceeds until all nodes have been solved or pruned. If an optimal solution is found for a subproblem, it is only a feasible solution to the whole problem, but not necessarily a globally optimal.

A Single Objective Branch and Bound Procedure In a single objective optimization problem, the branching and bounding procedures can be applied easily as stated above. A relaxation of the original problem can be used to compute the bounds. For example, a branch and bound procedure for $1|r_j|L_{max}$ can be constructed as follows.

In the branching process, at first check whether job is eligible for a particular position or not. For this let job c has to be considered as a candidate for position k only if

$$r_c < \min_l(\max(t, r_l) + p_l),$$

where $J_l \in J$, J being the set of jobs that are not yet scheduled and t denote the time the machine completes its previous job. If any job that does not satisfy this inequality then it will be pruned from the tree (Pinedo & Chao, 1999).

There are various ways to compute bounds, *e.g.* using a preemtive EDD (Earliest Due Date) rule. The preemtive EDD rule gives optimal schedule for $1|r_j, prmp|L_{\max}$. So $1|r_j, prmp|L_{\max}$ is the relaxation of $1|r_j|L_{max}$. Calculate LB at each node. If LB for a node exceeds the previously found UB, then this node will not branch further (Pinedo & Chao, 1999).

Branch and Bound in Multi-Objective Scheduling Context In multi-objective branch-and-bound procedures, one has to find the Pareto front of solutions (in fact one solution for each Pareto point in the objective space). Therefore, at each time of the search, one keeps the set of non-dominated solutions found so far instead of a single incumbent. Futhermore, unlike the single-objective case, there possibly exist several Pareto optimal solutions (more precisely, pareto optimal solutions with distinct images in the

objective space) that can be reached from a given node in the search tree. Hence, a natural extension of conventional branch-and-bounds can associate each node with a set of lower bounds.

Quality of bounds plays an important role in the success of any branch and bound method. Generally accepted and well known bounds in multiobjective optimization are the ideal point y^I and nadir point y^N, with $y^I < y < y^N$. Ideal Point represents a lower bound and is defind as

$$y_k^I = \min_{y \in Y_N} y_k, k = 1, ...n$$

Nadir point y^N, representing the upper bound on the value of any efficient point, is defined as

$$y_k^N = \max_{y \in Y_N} y_k, k = 1, ...n$$

Unfortunately these bounds are generally far away from the non-dominated points.

The concept of bounds can be generalized to bound sets for use in multi-objective optimization problems. For instance, local ideal points and local nadir points may represent a lower and uper bound set, where these local points are derived from two adjacent supported solutions in the objective space.

In the next section, we present the application of branch and bound procedure on our problem, $1|r_j|\sharp\{T_j\}$.

4 $1|r_j|\sharp\{T_j\}$ Problem

$1|r_j|\sharp\{T_j\}$ problem is an n-objective combinitorial optimization problem with n tasks to be executed on a single machine. Considering the tardiness of each task T_j as a performance measure for the scheduling problem with n tasks, makes the problem an n-objective scheduling problem. Note that $1|r_j|\sharp\{C_j\}$ problem is a special case of $1|r_j|\sharp\{T_j\}$ problem, obtained by setting $\forall j, d_j = 0$.

In $1||\sharp(f_1, ..., f_K)$ problem, all non-dominated solutions are enumerated without using a special objective. This notation is always related to a posteriori resolution context where the provided algorithm proceeds by enumerating all the solutions in order to retain the pareto optima (T'kindt & Billaut, 2006). This differs from the problem under study, as the number of objectives here are dependent on the number of tasks *i.e.* n. Therfore, a different notation of $\sharp\{T_j\}$ is used, representing the individual tardiness of each task as an objective.

As this is a multi-objective problem, there may not be a single schedule superior to all others. Thus the goal is to find the set of non-dominated schedules of the problem. A branch and bound procedure is developed to find the set of non-dominated solutions for $1|r_j|\sharp\{T_j\}$. Active schedule generation procedure is employed for the branching phase. Depth first strategy is used for generating the nodes of the tree for this procedure.

Two different bounding schemes are applied. First bounding scheme is based on solving $1|r_j, prmp|\sharp T_j$ instances at each node, where a node represents a partial schedule. The set of tardiness values for all n tasks, thus obtained, is used as a lower bound set for the original problem. This node is discarded if it is not dominated by any of the earlier lower bound set. This assumes that all the solutions of the subproblem of a node corresponding to a dominated solution are dominated as well.

Second bounding scheme is based upon local ideal points.

For an example $1|r_j|\sharp\{T_j\}$ problem as given in table 8, the procedure is illustrated in table 9 and fig. 6. Table 9 lists all the solutions, partial or complete, at every stage of the procedure. At the root node (*,*,*,*), *e.g.* there are two possible active branches. The procedure explores both the branches as this node is non-dominated with bound of (0,5,4,0). Note that, node 20, 21, 22 and 23 are listed in the table (for the sake of clarity), although these nodes are not explored by the procedure as their root node (19) is dominated. This can easily be seen in fig. 6 as well. Tables 10a and 10b list all the non-delay and active but not non-delay schedules respectively, for the problem given in table 8. There are 12 active schedules with 8 schedules as non-dominated.

Table 8: A $1|r_j|\sharp\{T_j\}$ problem P

j	r_j	p_j	d_j
1	0	4	8
2	1	2	12
3	3	6	11
4	5	5	10

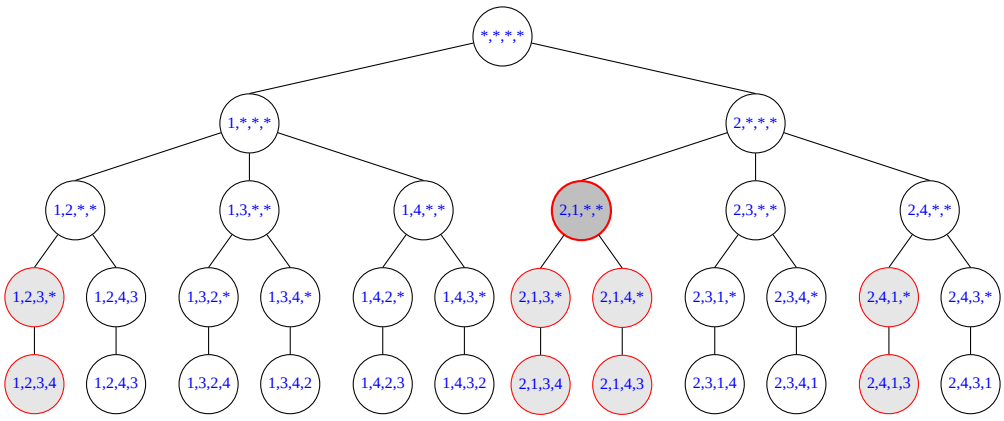


Fig. 6: Multi-objective branch and bound tree of active schedules

Table 9: Branch and bound procedure

Node no.	Node	No. 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,,7,20)	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

Table 10a: Non-delay schedules

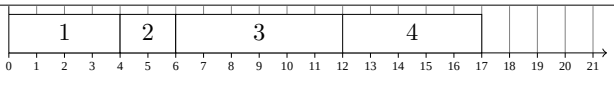
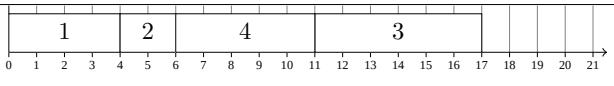
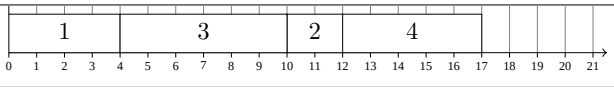
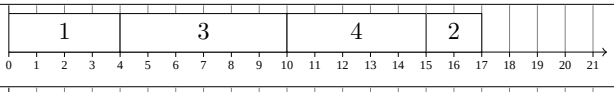
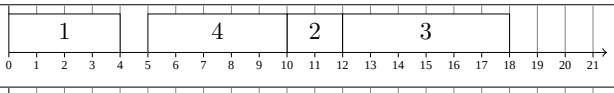
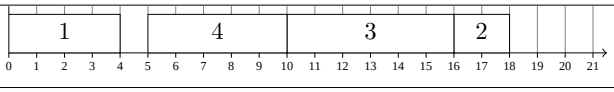
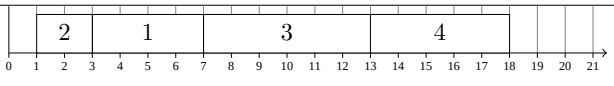
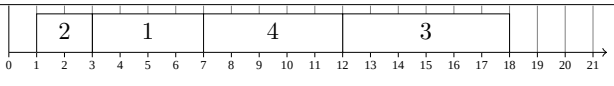
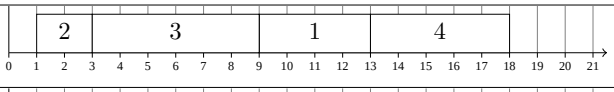
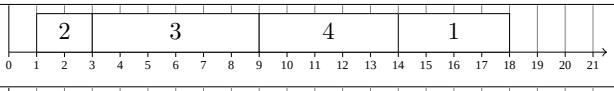
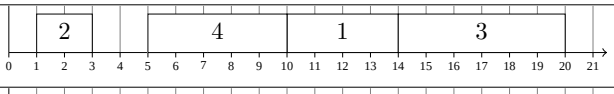
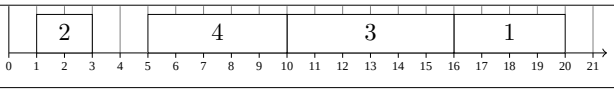
No.	σ	T_j	Dominance	Dom.	Dom. by
1		0,0,1,7	D	7,8	3
2		0,0,6,1	N	-	-
3		0,0,0,7	N	1,7,9	-
4		0,5,0,5	N	-	-
5		0,0,7,0	N	11	-
6		0,6,5,0	N	-	-

Table 10b: Active schedules

No.	σ	T_j	Dominance	Dom.	Dom. by
7		0,0,2,8	D	-	1,3,7
8		0,0,7,2	D	7	-
9		5,0,0,8	N	-	3
10		10,0,0,4	N	-	-
11		6,0,9,0	D	-	5
12		12,0,5,0	N	-	-

5 Conclusions & Perspectives

A single machine n-objective scheduling problem with individual tardiness of each task as an objective, denoted as $1|r_j|\sharp\{T_j\}$ is presented. A branch and bound procedure is built for this problem. Branching procedure is based on the enumeration of active schedules. The branch and bound tree is explored using depth first strategy. Two bounding schemes are used in the procedure. The main perspective of this study is to find a set of dominance rules in order to reduce the search space. Some better quality bounds for this problem are to be explored further. Computationaly efficient data structures for such n-objective problem may improve the performance of the procedure as well.

GLOSSARY OF NOTATION

$\mathbb{J}$	set of tasks.
$\mathbb{M}$	set of machines.
π	A sequence.
σ	A schedule.
r_j	Ready time, Ready time of j^{th} task.
p_j	Processing time of j^{th} task
d_j	Due date of j^{th} task.
$\tilde{d}_j$	The deadline of j^{th} task.
$C_j(\sigma)$	Completion time of j^{th} task in a schedule σ.
$L_j(\sigma)$	Lateness of j^{th} task in a schedule σ.
$T_j(\sigma)$	Tardiness time of j^{th} task in a schedule σ.
f	A performance measure.
$C_{\max}(\sigma)$	Makespan, or total completion time of a schedule σ.
$T_{\max}(\sigma)$	Maximum tardiness of a schedule σ.
$L_{\max}(\sigma)$	Maximum lateness of a schedule σ.
$\bar{T}(\sigma)$	Mean tardiness of a schedule σ.
$\mathrm{CMT}(\sigma)$	Conditional mean tardiness of a schedule σ.

Bibliography

Akturk, M. S. & Ozdemir, D., 2000, "An exact approach to minimizing total weighted tardiness with release dates," *IIE Transactions* 32, no. 11, pp. 1091–1101, 10.1023/A:1013741325877.

Akturk, M. S. & Ozdemir, D., 2001, "A new dominance rule to minimize total weighted tardiness with unequal release dates," *European Journal of Operational Research* 135, no. 2, pp. 394–412, doi: DOI: 10.1016/S0377-2217(00)00319-2.

Akturk, M. S. & Yildirim, M. B., 1998, "A new lower bounding scheme for the total weighted tardiness problem," *Comput. Oper. Res.* 25, no. 4, pp. 265–278, 290642.

Baptiste, P., Carlier, J., & Jouglet, A., 2004, "A Branch-and-Bound procedure to minimize total tardiness on one machine with arbitrary release dates," *European Journal of Operational Research* 158, no. 3, pp. 595–608, doi: DOI: 10.1016/S0377-2217(03)00378-3.

Chu, C. & Portmann, M. C., 1992, “Some new efficient methods to solve the n/1/ri/[epsilon]Ti scheduling problem,” *European Journal of Operational Research* 58, no. 3, pp. 404–413, doi: DOI: 10.1016/0377-2217(92)90071-G.

Du, J. & Leung, J. Y.-T., 1990, “Minimizing Total Tardiness on One Machine is NP-Hard,” *Mathematics of operations research* 15, no. 3, pp. 483–495.

Ehrgott, M. & Gandibleux, X., 2000, “A survey and annotated bibliography of multiobjective combinatorial optimization,” *OR Spectrum* 22, no. 4, pp. 425–460, 10.1007/s002910000046.

Emmons, H., 1969, “One-Machine Sequencing to Minimize Certain Functions of Job Tardiness,” *OPERATIONS RESEARCH* 17, no. 4, pp. 701–715.

Emmons, H., 1975a, “A note on a scheduling problem with dual criteria,” *Naval Research Logistics Quarterly* 22, no. 3, pp. 615–616.

Emmons, H., 1975b, “One machine sequencing to minimize mean flow time with minimum number tardy,” *Naval Research Logistics Quarterly* 22, no. 3, pp. 585–592.

Esswein, C., T’kindt, V., & Billaut, J., 2001, “A polynomial time algorithm for solving a single machine bicriteria scheduling problem,” Tech. rep., Laboratory of Computer Science, University of Tours (France).

Heck, H. & Roberts, S., 1972, “A note on the extension of a result on scheduling with secondary criteria,” *Naval Research Logistics Quarterly*, 19, pp. 59–66.

Hoogeveen, J. & Van de Velde, S., 1995, “Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time,” *Operations Research Letters* 17, no. 5, pp. 205–208.

Hoogeveen, J. A., 1992, *Single-Machine Bicriteria Scheduling*, Ph.D. Thesis, CWI, Amsterdam, The Netherlands.

John, T., 1989, “Tradeoff solutions in single machine production scheduling for minimizing flow time and maximum penalty,” *Computers & Operations Research* 16, no. 5, pp. 471–479.

Kiran, A. & Unal, A., 2006, “A single-machine problem with multiple criteria,” *Naval Research Logistics* 38, no. 5, pp. 721–727.

Lin, K., 1983, “Hybrid algorithm for sequencing with bicriteria,” *Journal of Optimization Theory and Applications* 39, no. 1, pp. 105–124.

Loukil, T., Teghem, J., & Tuyttens, D., 2005, “Solving multi-objective production scheduling problems using metaheuristics,” *European Journal of Operational Research* 161, no. 1, pp. 42–61, doi: DOI: 10.1016/j.ejor.2003.08.029.

Nelson, R., Sarin, R., & Daniels, R., 1986, “Scheduling with multiple performance measures: the one-machine case,” *Management Science* 32, no. 4, pp. 464–479.

Pinedo, M. & Chao, X., 1999, *Operations Scheduling with Applications in Manufacturing and Services*.

Sen, T. & Gupta, S., 1983, “A branch-and-bound procedure to solve a bicriterion scheduling problem,” *IIE Transactions* 15, no. 1, pp. 84–88.

S.French, 1982, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job Shop*, Ellis Horwood Ltd.

Smith, W. E., 1956, “Various optimizers for single-stage production,” *Naval Research Logistics Quarterly* 3, no. 1, pp. 59–66.

Squirrel, P. & Lopez, P., 1999, *L’advancement*, Economic, Paris, ISBN 2-7178-3798-1.

Su, L.-H. & Chen, C.-J., 2008, “Minimizing total tardiness on a single machine with unequal release dates,” *European Journal of Operational Research* 186, no. 2, pp. 496–503, doi: DOI: 10.1016/j.ejor.2006.07.051.

Szwarc, W., Croce, F. D., & Grosso, A., 1999, “Solution of the single machine total tardiness problem,” *Journal of Scheduling* 2, no. 2, pp. 55–71, 10.1002/(SICI)1099-1425(199903/04)2:2¡55::AID-JOS14¿3.0.CO;2-5.

Szwarc, W., Grosso, A., & Croce, F. D., 2001, “Algorithmic paradoxes of the single-machine total tardiness problem,” *Journal of Scheduling* 4, no. 2, pp. 93–104, 10.1002/jos.69.

Szwarc, W. & Mukhopadhyay, S. K., 1996, “Decomposition of the single machine total tardiness problem,” *Operations Research Letters* 19, no. 5, pp. 243–250, doi: DOI: 10.1016/S0167-6377(96)00031-4.

T’kindt, V. & Billaut, J., 2006, *Multicriteria scheduling: theory, models and algorithms*, Springer Verlag.

VanWassenhove, L. & Gelders, L. F., 1980, “Solving a bicriterion scheduling problem,” *European Journal of Operational Research* 4, pp. 42–48.