Al-Mustansiriyah Journal of Science
ISSN: 1814-635X (print), ISSN: 2521-3520 (online)
Volume 28, Issue 3, 2017
DOI: http://doi.org/10.23851/mjs.v28i3.122

Research Article

Solving Composite Multi objective Single Machine Scheduling Problem Using Branch and Bound and Local Search Algorithms

Tariq S. Abdul – Razaq[1], Hafed M. Motair[2]
[1]Department of Mathematics, College of Science, Mustansiriyah University, IRAQ
[2]Department of Mathematics, Open Educational College, Al- Qadisiya, Ministry of Education, IRAQ
*Correspondent Author Email: hafedmotair@gmail.com

ArticleInfo

Received:
2 Apr. 2017

Accepted:
17 Oct. 2017

:

Abstract

This paper present algorithm for solving a single machine scheduling problem to minimize the sum of total completion times, total tardiness, maximum tardiness, and maximum earliness. The single machine total tardiness problem is already NP-hard, so they consider problem is strongly NP-hard, and several algorithms are used to solve it. Branch and bound algorithm with dominance rule and local search algorithms are proposed for the problem. For the Branch and bound algorithm results- show that using dominance rule improve the performance of the algorithm in both computation times and optimal values, but it needs longer times. Thus we tackle the problem of large sizes with local search algorithms descent method, simulated annealing and tabu search. The performance of these algorithms is evaluated on a large set of test problems and the results are compared. The computational results show that simulated annealing algorithm and Tabu search algorithm are better than descent method with preference to simulated annealing algorithm, and show that the three algorithms find optimal or near optimal solutions in reasonable times.

Keywords: Multicriteria Scheduling, Branch and bound, Dominance rule, Local search algorithms.

الخلاصة

قدم البحث خوارزميات لحل مسألة الجدولة لماكنة واحدة لتصغير مجموع الدوال: مجموع زمن إتمام النتاجات, مجموع أزمان التبكير, اكبر زمن تبكير, وأكبر زمن تأخير. اقترحنا خوارزمية التفرع والتقييد (BAB), مع قواعد الهيمنة (DR) التي حسنت من أداء الخوارزمية من ناحية القيمة المثالية والزمن, واقترحنا خوارزميات البحث المحلي SA , TS, DM , أظهرت النتائج أن خوارزميات SA و TS أفضل من خوارزمية DM مع أفضليه لخوارزمية SA, وأن الخوارزميات الثلاث أوجد الحلول المثالية أو القريبة من الحل المثالي في أزمنة مناسبة.

Introduction

Scheduling concerns the allocation of limit resources to tasks over time. It is a decision-making process that has as a goal the optimization of one or more objectives [11]. Scheduling problems in real life applications generally involves optimization of more than one criteria. These criteria are often conflicting in nature and quite complex, according to how these criteria optimized, multicriteria optimization can be divided into two types:

1. Hierarchical optimization for which if one criteria, say f is more important than other one g, then the problem is to minimize primary criterion f while breaking ties in favor of the schedule that has minimum secondary criterion value g and denoted by Lex(f, g) .
2. Simultaneous optimization for which if no criterion is dominant and all criteria are considered simultaneously.

It is well known that the problem $1||\sum_j T_j$ is NP-hard [6], so any problem containing the cost function $\sum_j T_j$ as subproblem is also NP-hard. Sen and Gupta[14] studies the problem of minimizing a linear combination of flow times and maximum tardiness of a given number of jobs on a single machine and presents BAB

technique to arrive at an optimal solution. The first result involving E_{max} and T_{max} is due to Garey et al.[8], they study a single machine scheduling problem and present an O(nlogn) algorithm to check whether for a given threshold value y there exists a feasible schedule such that each job is executed in the interval$[d_i - p_i - y, d_i - y]$. By applying binary search, they find the minimum such value y. Gareyet al. also show that the problem of finding the set of starting time that minimizes $\sum E_j + \sum T_j$ for a given ordering of the jobs on a single machine is solvable in O(nlogn) time. Verma and Dessouky [15] show that the problem $1|\mathrm{p_j} = \mathrm{p}|\sum_{\mathrm{j}}(\alpha_{\mathrm{j}}\mathrm{E_j} + \beta_{\mathrm{j}}\mathrm{T_j})$ is solvable in polynomial time if the weights are agreeable, that is, the jobs can be renumbered such that: $\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_n, \beta_1 \leq \beta_2 \leq \cdots \leq \beta_n$. Abdul Razaq and Ibrahim[1] studied the problem $1||\mathrm{F}(\mathrm{E_{max}}, \mathrm{T_{max}})$ they propose general algorithm (ET) to find the set of approximate efficient (Pareto optimal) solutions, and for the problem $1||E_{max} + T_{max}$ they find the (near) optimal solution using BAB algorithm . Abdul-Razaq and Mahrooz [2] study problem including three criteria, total completion times, the total tardiness and the maximum tardiness, they propose BAB algorithm for the problem $1||\sum_{\mathrm{j}}\mathrm{C_j} + \sum_{\mathrm{j}}\mathrm{T_j} + \mathrm{T_{max}}$ and find the set ofefficient solutions for the problem$1||(\sum_{\mathrm{j}}\mathrm{C_j}, \sum_{\mathrm{j}}\mathrm{T_j}, \mathrm{T_{max}})$. Lee and Choi in [12], propose a genetic algorithm for solving the scheduling problem with distinct due dates in a single machine general penalty weights which are not necessarily proportional to the processing times are applied to jobs either early or tardy. The computational experiments show that the genetic algorithm finds optimal solutions for small instances. Wan and Benjamin in [16] propose a tabu search algorithm for the problem of minimizing the earliness tardiness cost on a single machine with due windows. Computational experiments indicate that the performance of the proposed approach is quite well, especially for the instances of large size.

Problem formulation

Let $N = \{1,2,\dots,n\}$ be the set of n jobs that must be processed by a machine. Each job i has a processing timep_i, and a due date d_i , for $i = 1,\dots,n$.Initially, all of the jobs are available to be processed by the machine and it starts processing without interrupted, and requires p_i units of time to complete its processing. Thus a schedule for the machine can be completely specified by giving the sequence in which the jobs are processed.

Let σ be a sequence of the jobs in N representedby the n-tuple $(\sigma(1), \sigma(2), \dots, \sigma(\mathrm{n}))$ where σ(i) is the ith job processedby the machine. The completion time of jobσ(i) is given by $\mathrm{C_{\sigma(i)}} = \sum_{\mathrm{j=1}}^{\mathrm{i}} \mathrm{p_{\sigma(j)}}$,the tardiness of the job σ(i) is given by $\mathrm{T_{\sigma(i)}} = \max(\mathrm{C_{\sigma(i)}} - \mathrm{d_{\sigma(i)}}, 0)$, and the earliness of the job σ(i) is given by $\mathrm{E_{\sigma(i)}} = \max(\mathrm{d_{\sigma(i)}} - \mathrm{C_{\sigma(i)}}, 0)$.

We consider the following performance criteria:

- the sum of completion times $\sum_j C_j$.
- the sum of tardiness $\sum_j T_j$.
- the maximum tardiness T_{max} .
- the maximum earliness E_{max} .

Hence this problem is denoted by $1||\sum_j C_j + \sum_j T_j + T_{max} + E_{max}$ (Q) which belong to type (2) of multicriteria optimization and written as :

$$z = min\left(\sum_j C_j + \sum_j T_j + T_{max} + E_{max}\right) \quad S.T \quad \left.\begin{array}{ll} C_i \geq p_i & i = 1,2,\dots,n \\ C_i = C_{i-1} + p_j & i = 2,3,\dots,n \\ T_i \geq C_i - d_i & i = 1,2,\dots,n \\ T_i \geq 0 & i = 1,2,\dots,n \\ E_i \geq d_i - C_i & i = 1,2,\dots,n \\ E_i \geq 0 & i = 1,2,\dots,n \end{array}\right\} \quad (\boldsymbol{Q})$$

Solving the Problem (Q) Using BAB Algorithm

The aim of this problem is to find the minimum value of the of the objective function $\sum_{\mathrm{j}}\mathrm{C_j} + \sum_{\mathrm{j}}\mathrm{T_j} + \mathrm{T_{max}} + \mathrm{E_{max}}$, this function include the cost function $\sum_{\mathrm{j}}\mathrm{T_j}$, as mentioned before any problem including this function as sub problem is NP-hard. Since the feasible set of solutions considered in our problem is finite set, so an

optimal solution can be obtained by a straight forward algorithm that enumerates all feasible solutions, and then outputs the one with the minimum objective value. Complete enumeration method (CEM) is hardly practical because the number of cases to be considered is usually enormous. Solving NP-hard discrete optimization problems to optimality is often an immense job requiring very efficient algorithms, and Branch and bound algorithm (BAB) is one of the main tools in construction of these. A BAB algorithm searches the complete space of solutions for a given problem for the best solution. However, explicit enumeration is normally impossible due to the exponentially increasing number of potential solutions. A BAB algorithm proceeds by repeatedly partitioning the class of all feasible solutions into smaller and smaller subclasses in such a way that ultimately an optimal solution is obtained. The BAB have the following general characteristics:

- A branching rule that defines partitions of the set of feasible solutions into subsets.
- A lower bounding rule that provides a lower bound (LB) on the value of each solution in a subset generated by the branching rule.
- A search strategy that selects a node from which to branch.

To implement the BAB algorithm we decompose the problem (Q) into sub problems:

Decomposition of the Problem $(\boldsymbol{Q})$

Decomposition is a general approach for solving a problem by breaking it up into smaller ones and solving each of the smaller ones separately, either in parallel or sequentially. The problem (Q) can be decomposed into four sub problems Q_i , $i = 1,2,3,4$: $1||Z_i$, where $Z_1 = \min \sum_j C_j$, $Z_2 = \min \sum_j T_j$, $Z_3 \min T_{max}$, $Z_4 = \min E_{max}$.

Derivation of Lower bound (LB) and Upper bound (UB) for the Problem $(\boldsymbol{Q})$

Next theorem help us to derive the lower bound (LB) for problem (Q) using the decomposed problems:

Theorem 1: Let Z_i be the lower bound or the minimum for the objective functions of sub problems Q_i , $i = 1,2,3,4$ and let Z be the minimum of objective function of problemQ, then $\sum_i Z_i \leq Z$

Proof: It is clear

The initial lower bound $ILB = \sum_{i=1}^{n} Z_i$ where Z_1 calculated by sequencing the jobs in SPT order to get the minimum value of total completion times. Z_2 calculated by sequencing the jobs in EDD order to get the minimum maximum tardiness $T_{max}(EDD)$ using that $T_{max}(EDD) \leq \sum_j T_j$ .Z_3 calculated by sequencing the jobs in EDD order to get minimum maximum tardiness $T_{max}(EDD).Z_4$ calculated by sequencing the jobs in MST order to get the minimum maximum earliness $E_{max}(MST)$. Hence the initial lower bound ILB calculated as follwes:

$ILB = \sum_j^n C_j(SPT) + T_{max}(EDD) + T_{max}(EDD) + E_{max}(MST)$

The upper bound (UB) is derived as follows: Compute the first and second upper bound by sequencing the jobs in SPT and EDD order respectively, then

$UB1 = \sum_j C_j(SPT) + \sum_j T_j(SPT) + T_{max}(SPT) + E_{max}(SPT)$,

$UB2 = \sum_j C_j(EDD) + \sum_j T_j(EDD) + T_{max}(EDD) + E_{max}(EDD)$.

The upper bound is computed by: $UB = \min(UB1, UB2)$.

Dominance Rule

A dominance rule is established in order to reduce the solution space either by adding new constraints to the problem, or by writing a procedure that attempts to reduce the domain of variables, or by building interesting solutions directly [10]. This dominance rule is expressed as "There exist at least one optimal solution of S (the set of feasible solutions) having the property A". The strategy is as follows: Any solution which does not satisfy A can be

removed from S, because there is at least one optimal solution of S satisfying A.

Theorem 2: If $p_i \le p_j$ and $d_i \le d_j$ for every $i, j = 1,2, \dots, n$, then job i preceed job j in optimal solution for the problem(Q).

Proof: consider the sequence $\sigma = \sigma_1 i\, j\, \sigma_2$ and the sequence $\acute{o} = \sigma_1\, ji\, \sigma_2$ which is obtained by interchange the position of jobs i and j.

For the sequence σ and ǿ there are two cases

Case 1 : If $p_i \le p_j$ and $d_i \le d_j$ implies $s_i \le s_j$ for every $i, j = 1,2, \dots, n$

From $p_i \le p_j$ we have :$\sum_j C_j(\sigma) \le \sum_j C_j(\acute{o})$

From the condition of slack time $s_i \le s_j$, we have:

$$E_{max}(\sigma) \le E_{max}(\acute{o}).$$

From $p_i \le p_j$ and $d_i \le d_j$, we have:

$$T_{max}(\sigma) \le T_{max}(\acute{o}) \text{ and } \sum_j T_j(\sigma) \le \sum_j T_j(\acute{o}).$$

Hence we have:

$$\sum_j C_j(\sigma) + \sum_j T_j(\sigma) + T_{max}(\sigma) + E_{max}(\sigma) \le \sum_j C_j(\acute{o}) + \sum_j T_j(\acute{o}) + T_{max}(\acute{o}) + E_{max}(\acute{o}).$$

Case 2: If $p_i \le p_j$ and $d_i \le d_j$implies $s_i \ge s_j$ for every $i, j = 1,2, \dots, n$

From $p_i \le p_j$ we have:

$$\sum_j C_j(\sigma) \le \sum_j C_j(\acute{o}) \qquad (1)$$

The condition on processing times insure that (1) is satisfied, and the addition in cost which is obtained from (1) is equal to $p_j - p_i$, this gives :

$$\sum_j C_j(\sigma) + p_j - p_i = \sum_j C_j(\acute{o}) \qquad (2)$$

From the condition of slack times $s_i \ge s_j$ implies $E_{max}(\sigma) \ge E_{max}(\acute{o})$.

Also the addition in cost $s_i - s_j$ gives:

$$E_{max}(\acute{o}) + s_i - s_j = E_{max}(\sigma) \qquad (3)$$

$$s_i - s_j = (d_i - p_i) - (d_j - p_j)$$

$$= (d_i - d_j) + (p_j - p_i)$$

$$\le p_j - p_i \qquad (4)$$

Adding $E_{max}(\acute{o})$ to both side of (4) we have:

$E_{max}(\acute{o}) + s_i - s_j \le E_{max}(\acute{o}) + p_j - p_i$ and from (3) we have:

$$E_{max}(\sigma) \le E_{max}(\acute{o}) + p_j - p_i \qquad (5)$$

Adding $\sum_j C_j(\sigma)$ to both side of (5) and by (2) we have:

$$\sum_j C_j(\sigma) + E_{max}(\sigma) \le \sum_j C_j(\acute{o}) + E_{max}(\acute{o}) \qquad (6)$$

From the conditions $p_i \le p_j$ and $d_i \le d_j$ we have:

$$T_{max}(\sigma) \le T_{max}(\acute{o}) \text{ and } \sum_j T_j(\sigma) \le \sum_j T_j(\acute{o}).$$

By adding this result to (6) :

$$\sum_j C_j(\sigma) + \sum_j T_j(\sigma) + T_{max}(\sigma) + E_{max}(\sigma) \le \sum_j C_j(\acute{o}) + \sum_j T_j(\acute{o}) + T_{max}(\acute{o}) + E_{max}(\acute{o})$$

Hence in both cases the sequence σ is better than the sequence ǿ . Hence job i precede job j in the optimal solution.

Proposition 1: If SPT and EDD rules are identical in one sequence, then this sequence gives optimal solution for problem (Q) .

Proof: It is clear from theorem (2) .

Solving the Problem (Q) Using Local Search Algorithms (LSAs)

Branch and bound algorithm (and dynamic programming) is based on the idea of intelligently enumerating all feasible solutions. Another possibility is to apply (LSAs). These algorithms produce solutions that are guaranteed to be within a fixed percentage of the actual optimum. One of the most successful methods of attacking hard combinatorial optimization problems is the discrete analog of "hill climbing", known as local (or neighborhood) search [4]. In neighborhood search, a current solution is transformed into a new solution according to some neighborhood

structure. An acceptance rule decides whether the move from the current solution to the transformed solution should be accepted, although the decision is sometimes delayed until the complete neighborhood (or a subset of it) is explored. If a move is accepted, then the transformed solution replaces the previous solution and becomes the current solution; otherwise, the move is rejected and the current solution is retained. This process is repeated until some termination criterion is satisfied. The acceptance rule is usually based on the objective function values of the current solution and its neighbor.

Descent Method (DM)

The simplest type of local search algorithms is descent method DM, which is sometimes known as iterative local improvement. In this method, only moves that result in an improvement in the objective function value are accepted. Under a first improve search, the first move that improves the objective function value is accepted [3]. On the other hand, best improve selects a move that yields the best objective function value among all neighbors. When no further improvement can be achieved, a DM terminates with a solution that is a local optimum. The local optimum is not necessarily the true global optimum.

Simulated Annealing (SA)

Simulated annealing is a local search algorithm (metaheuristics) capable of escaping from local optima ease of implementation and convergence properties. At each iteration of SA, the values for two solutions (the current solution and a newly selected solution) are compared. Improving solutions are always accepted; while a fraction of non-improving (inferior) solutions are accepted in the hope of escaping local optima in search of global optima. The probability of accepting non-improving solutions depends on a temperature parameter, which is typically non-increasing with each iteration of the algorithm [13].

Tabu Search (TS)

Tabu search is another example of neighborhood search that, like SA, is capable of avoiding being trapped in local optima, but the operation of TS is quite different than SA's. At each iteration of TS, a subset of the neighbors of the current solution is considered, and a best of these is chosen. This contrasts to SA which choses a neighboring solution at random and accepts or rejects it on the basis of a probabilistic function. The subset of neighboring solution considered at each step is made up of all the solutions in the neighborhood minus some set of solutions which are considered tabu. The tabu solutions (tabu list) are usually solutions or moves that would bring the search back to a solution that has already been visited[9].The tabu list is a form of short term memory that guides the search away from areas that have already been seen.

Simple Heuristic method (SH)

We use the Simple Heuristic method (SH) to find initial solutions for the local search algorithms [7], which is the following steps:

Step 1: order the jobs in SPT rule.

Step 2: set k=2, chose the first two jobs in ordered sequence, schedule them in order to minimize the objective function, and set the better one as the current solution.

Step 3: Increment k by 1, and generate k candidate sequences by inserting the first job in the remaining jobs into each slot of the current solution, select the best one from these solutions that minimize the objective function. Update the selected partial solution as the new current solution.

Step 4: If k=n. stop, otherwise go to step 3.

Materials and Methodology

Computational Experiments:

Test problems: All experimental design tests are conducted on a personal computer intel (R) Core TM i7 CPU @ 2.50 GHz, and 8.00 GB of RAM. To present the efficiency and compared the results for the problem(Q). Instances with different sizes are considered. The processing times p_i , $i = 1,2, \dots, n$ for each problem is generated randomly from uniform distribution on the interval [1, 10], the due dates d_i , $i =$

$1,2,\dots,n$ is drown from uniform distribution on $\left[\left(1-\mathrm{TF}-\frac{\mathrm{RDD}}{2}\right)t,\left(1-\mathrm{TF}+\frac{\mathrm{RDD}}{2}\right)t\right]$[5], where t is total processing times for all jobs. The valueRDD is the relative range of due dates, it determines the length of the interval from which the due dates were taken, TF the tardiness factor determines the relative positions of the center of this interval between 0 and t, these value of TF and RDDare chosen from the set $\{0.2,0.4,0.6,0.8,1.0\}$. From 25 pairs of values of TF and RDD we generate 10 problem instances for each n, from n=4 to n=25 for BAB algorithm, while for the LSAs from n=4 to n=23 for small size problems and from n=50 to n=2000 for large size problems.

Results and Discussions

Comparison of the results of the problem (Q) using BAB algorithm:

Table 1 shows the average of experimental results of the two methods BAB without DR and BAB with DR. This table contains the number of jobs (n), for each of the ten instances, the average number of nodes, the average computational times, the average of optimal values, and the percentage of solved instances for problem(Q). We see that the BAB with DR gives fairly good results in terms of computational times and the number of nodes. For example for n=16 the BAB without DR failed to find optimal solution within 1800 second for some instances and solved 60 % of these instances, while BAB with DR solved problems in average CPU Time = 2.16 seconds and average number of nodes = 46267.9 . Also Table (1) shows that the BAB without DR solved all problem instances from n=4 to n=14, and failed to solve all problems from n=21 to n=25, while the BAB with DR solved the problem in all of instances from n=4 to n=23 and failed to solve problem in one problem when n=24, and two problems when n=25.

Table 1: The average results of algorithms BAB without DR and BAB with DR.

n	BAB without DR.				BAB with DR.			
	Mean Best	Mean Nods	Mean Time	% solved	Mean Best	Mean Nods	Mean Time	% solved
4	55.5	11.5	0.015112	100 %	55.5	4.9	0.013182	100 %
5	97	22.9	0.001144	100 %	97	8.9	0.000785	100 %
6	110.7	52.7	0.002143	100 %	110.7	16.1	0.000951	100 %
7	151.5	137	0.007278	100 %	151.5	31.6	0.002678	100 %
8	195.3	373.2	0.015159	100 %	195.3	47.9	0.002585	100 %
9	255.3	1370.9	0.052516	100 %	255.3	76.8	0.003826	100 %
10	283.5	4383.3	0.163119	100 %	283.5	213	0.010916	100 %
11	349.5	16626.2	0.594771	100 %	349.5	225.6	0.010774	100 %
12	396.5	45324.6	1.671211	100 %	396.5	413.7	0.019953	100 %
13	426.9	207839.2	7.502205	100 %	426.9	784.6	0.036156	100 %
14	529.1	3213032	124.968	100 %	529.1	7461.5	0.353652	100 %
15	696.3	11636173	439.4964	90 %	696.3	16595.8	0.802141	100 %
16	693.1	20610685	764.4811	60 %	693	46267.9	2.157109	100 %
17	716.3	21511161	912.4015	60 %	716.1	6450.9	0.334145	100 %
18	851.4	41939475	1637.207	20 %	850.4	74039.5	3.690589	100 %
19	967.5	43225118	1745.618	10 %	966.1	244294.5	12.30061	100 %
20	928.8	38241525	1443.574	20 %	928.3	892527.4	39.10577	100 %
21	1154.7	45789608	1800	0 %	1152.7	3329758	164.8119	100 %
22	1075.4	43871985	1800	0 %	1258	3025949	142.3837	100 %
23	1285.9	43273922	1800	0 %	1282	1696230	79.19502	100 %
24	1336.4	44816660	1800	0 %	1335.6	14667602	662.7902	90 %
25	1500.2	43181493	1800	0 %	1497.2	12692192	640.4201	80 %

Comparison of results of the problem (Q) **using local search algorithms**

Computational results of (LSAs), (SA), (TS), and (DM) summarized in tables (2), (3). For the implementation of LSAs we generate the initial solution by SH algorithm which described in section 4.4, and we use arrange of number of iterations starts from 20000 iteration for small size problems where$4 \leq n \leq 23$, and use 50000 iterations for medium size problems where $50 \leq n \leq 400$, for large size problems where $n \geq 500$ we use 100000 iterations. The neighborhoods generated using two methods, the API (Adjacent Pairwise Interchange) method and insertion method, and the algorithm interchange between the two neighborhoods at each iteration according to wither the number of iterations is odd ore even. The comparison between BAB algorithm and LSAs is summarized in Table 2, the results shows that SA and TS algorithms solve all problems and reach the optimal solutions for all small size problems where DM algorithm is not for some instances, also the results shows that TS need more time than other LSAs to reach the optimal solutions. In Table 3 we summarize the results of comparison among LSAs themselves, for each algorithm we find the mean of best values and the mean of computation times. Also Table 3 shows that the performance of SA and TS algorithms is better than DM and that SA algorithm better than TS algorithm.

Table 2: The average results of BAB and local search algorithms for small size problems.

n	BAB		SA		TS		DM	
	Mean optimal	Mean Time	Mean value	Mean Time	Mean value	Mean Time	Mean value	Mean Time
4	55.5	0.0170	55.5	0.6663	55.5	0.6566	55.5	0.6412
5	97	0.0008	97	0.7883	97	0.6372	97	0.6318
6	110.7	0.0009	110.7	0.6472	110.7	0.6441	110.7	0.6411
7	151.5	0.0016	151.5	0.6544	151.5	0.6541	151.5	0.6353
8	195.3	0.0026	195.3	0.6548	195.3	0.6720	195.3	0.6559
9	255.3	0.0037	255.3	0.6650	255.3	0.6714	255.3	0.6422
10	283.5	0.0099	283.5	0.6499	283.5	0.6471	283.5	0.6329
11	349.5	0.0109	349.5	0.7321	349.5	0.7101	349.5	0.6853
12	396.5	0.0185	396.5	0.6624	396.5	0.6616	396.5	0.6456
13	426.9	0.0359	426.9	0.7122	426.9	0.7076	427	0.6905
14	529.1	0.3167	529.1	0.6637	529.1	0.6717	529.1	0.6466
15	696.3	0.7606	696.3	0.6981	696.3	0.6900	696.3	0.6804
16	693	1.9578	693	0.6631	693	0.6917	693.2	0.6589
17	716.1	0.3007	716.1	0.6602	716.1	0.7506	716.2	0.6463
18	850.4	3.5619	850.5	0.7051	850.5	0.8422	850.9	0.6883
19	966.1	11.164	966.1	0.6789	966.1	1.0168	966.1	0.6635
20	928.3	38.04	928.4	0.6771	928.4	1.0115	928.5	0.6625
21	1152.7	160.24	1152.7	0.6788	1152.7	0.8426	1153	0.6619
22	1073.7	198.12	1073.7	0.6822	1073.7	0.9717	1073.8	0.6640
23	1282	82.756	1282	0.7448	1282	1.3327	1282.1	0.7540

Table 3: The average results of local search algorithms for large size problems.

n	SA		TS		DM	
	Mean Best	Mean Time	Mean Best	Mean Time	Mean Best	Mean Time
50	5654.7	1.8871	5654.9	4.9127	5656.6	1.8518
100	20078	2.3228	20079.1	7.6682	20079.9	2.2943
150	46948.4	2.4242	46946.2	10.604	46957.9	2.4356
200	81334.5	5.9961	81335.5	73.222	81340.8	6.0744
250	124500.8	7.4341	124491.8	90.313	124502.1	7.4767
300	174800.5	8.9283	174795.3	118.92	174809.8	8.8176
400	312125.8	9.0340	312146.1	101.38	312191.7	8.9497
500	501303.4	12.290	501319.3	126.61	501410.6	12.209
600	716888.5	16.435	716878.1	152.47	716983.3	16.372
700	985236.6	21.683	985328.1	181.93	985365.5	21.617
800	1259230.2	28.439	1259180.1	215.35	1259230.7	28.379
900	1610269.8	36.401	1610361	245.78	1610416	36.297
1000	1956879.4	46.160	1956764.1	284.86	1956735.2	46.007
1500	4439021.1	125.85	4439195.6	767.05	4439669.1	125.45
2000	7868336	272.65	7868393.4	1464.2	7868473.6	272.92

Conclusions

It is well known that the sizes and the results of multicriteria scheduling problems are generally affected by a number of cost functions, and our problem (Q) is the sum of four cost functions. Therefore BAB algorithm failed to solve problems up to 25 jobs, on the other hand the comparison between BAB without and with dominance rule shows that dominance rule improve the performance of the BAB algorithm in both computation times and optimal values. In LSAs if we don't focus our attention on computation times we see that SA is better than TS and DM to solve our problem(Q). Because of its structure TS need more time than other algorithms to find best solution, but this time remain less than 1800 seconds.

References

[1] R. Tavakkoli-Moghaddam, G. Moslehi, M. Vasei, and A. Azaron, "Optimal scheduling for a single machine to minimize the sum of maximum earliness and tardiness considering idle insert," *Applied Mathematics and Computation*, vol. 167, pp. 1430-1450, 2005.

[2] T. S. Abdul-Razaq and Z. M. Ali, "Minimizing the Total Completion Times, the Total Tardiness and the Maximum Tardiness," *Ibn AL-Haitham Journal For Pure and Applied Science*, vol. 28, pp. 155-170, 2017.

[3] B. Chen, C. N. Potts, and G. J. Woeginger, "A review of machine scheduling: Complexity, algorithms and approximability," in *Handbook of combinatorial optimization*, ed: Springer, 1998, pp. 1493-1641.

[4] P. Brucker and P. Brucker, *Scheduling algorithms* vol. 3: Springer, 2007.

[5] H. Crauwels, C. N. Potts, and L. N. Van Wassenhove, "Local search heuristics for the single machine total weighted tardiness scheduling problem," *INFORMS Journal on computing*, vol. 10, pp. 341-350, 1998.

[6] J. Du and J. Y.-T. Leung, "Minimizing total tardiness on one machine is NP-hard," *Mathematics of operations research*, vol. 15, pp. 483-495, 1990.

[7] T. Eren, "A multicriteria scheduling with sequence-dependent setup times," *Applied Mathematical Sciences*, vol. 1, pp. 2883-2894, 2007.

[8] M. R. Garey, R. E. Tarjan, and G. T. Wilfong, "One-processor scheduling with symmetric earliness and tardiness penalties," *Mathematics of Operations Research*, vol. 13, pp. 330-348, 1988.

[9] J. D. Knowles, "Local-search and hybrid evolutionary algorithms for Pareto

optimization," University of Reading UK, 2002.

[10] A. Jouglet and J. Carlier, "Dominance rules in combinatorial optimization problems," *European Journal of Operational Research*, vol. 212, pp. 433-444, 2011.

[11] V. T'kindt and J.-C. Billaut, *Multicriteria scheduling: theory, models and algorithms*: Springer Science & Business Media, 2006.

[12] C.-Y. Lee and J. Y. Choi, "A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights," *Computers & Operations Research*, vol. 22, pp. 857-869, 1995.

[13] M. Gendreau and J.-Y. Potvin, *Handbook of metaheuristics* vol. 2: Springer, 2010.

[14] T. Sen and S. K. Gupta, "A branch-and-bound procedure to solve a bicriterion scheduling problem," *AIIE Transactions*, vol. 15, pp. 84-88, 1983.

[15] S. Verma and M. Dessouky, "Single-machine scheduling of unit-time jobs with earliness and tardiness penalties," *Mathematics of Operations Research*, vol. 23, pp. 930-943, 1998.

[16] G. Wan and B. P.-C. Yen, "Tabu search for single machine scheduling with distinct due windows and weighted earliness/tardiness penalties," *European Journal of Operational Research*, vol. 142, pp. 271-281, 2002.