ISSN: 0067-2904

Using Heuristic and Branch and Bound Methods to Solve a Multi-Criteria Machine Scheduling Problem

Aseel Aboud Jawad*, Faez Hassan Ali, Wafaa Sayyid Hasanain
Mustansiryah University, College of Science, Mathematics Department, Baghdad, Iraq

Received: 12/9/2019 Accepted: 19/11/2019

Abstract

In this paper, we investigate some methods to solve one of the multi-criteria machine scheduling problems. The discussed problem is the total completion time and the total earliness jobs $(1//(\sum C_j, \sum E_j))$. To solve this problem, some heuristic methods are proposed which provided good results. The Branch and Bound (BAB) method is applied with new suggested upper and lower bounds to solve the discussed problem, which produced exact results for $n \leq 20$ in a reasonable time.

Keywords: Machine Scheduling problem, completion time, earliness, Branch and Bound.

استخدام الطرق التقريبية وطريقة التفرع والتقييد لحل مسألة جدولة الماكنة متعددة المعايير

اسيل عبود جواد* ، فائز حسن علي، وفاء سيد حسنين
قسم الرياضيات, كلية العلوم، الجامعه المستنصريه، بغداد, العراق.

الخلاصه

في هذا البحث سنتناول بعض الطرق لحل واحدة من مسائل جدولة الماكنة متعددة المعايير. ان المسألة المراد مناقشتها هي مسألة وقت الاتمام الكلي ووقت التبكير الكلي $(1//(\sum C_j, \sum E_j))$. لحل هذه المسألة, تم اقتراح بعض الطرق التقريبية التي اعطت نتائج جيدة. وقد تم تطبيق طريقة التفرع والتقيد المقترحة لحل المسألة والتي اعطت نتائج مضبوطة ل $n \leq 20$ ضمن وقت مقبول.

1. Introduction

There are many definitions for machine scheduling; scheduling problem is the allocation of resources over time to perform a collection of tasks. Resources and tasks are called machine and jobs respectively and both of than can take many forms [1]. The two concepts can take many forms. For example, we can consider computers as machines and the programs that are to be run on these computers as jobs.

There are two general constraints in the classical scheduling theory [2]. Each job can be processed by one machine each time and every machine is processing at most one job each time. A schedule is called feasible schedule if it satisfies the two main constraints, and if it satisfies many requirements relating to the discussed problem type. The type of the problem is related to the machine environment, the job characteristics, and the criterion of optimality.

There are many known exact and approximation solution methods to solve the Machine Scheduling Problem (MSP). The exact solutions are obtained using the Complete Enumeration Method (CEM), Branch and Bound (BAB) method, and Dynamic Programming (DP) method [3].

*Email: aseelabood6@gmail.com

Multi-criteria optimization depending on conflicting objective functions establishes a set of Pareto optimal solutions (Efficient solutions), instead of one optimal solution. This set includes one (many) solution(s) that no other solution(s) is better with respect to objective functions. In the literature, there are two approaches for multi-criteria scheduling problems:

1. **The hierarchical approach**: the first criterion is considered as the primary criterion and the second one is considered as the secondary criterion. The objective is to minimize the primary criterion while breaking all ties in favor of the schedule which has a minimum secondary criterion value [4].
2. **The simultaneous approach**: there are, at most, two types; the first one generates all efficient schedules then selects the one that yields the best composite objective function value of the criteria. The second one is to find the sum of these objectives [4]. Van Wessenhove and Gelder [5] studied the efficiency with respect to the criteria of the total completion time and the maximum tardiness in a single machine problem. A survey on multicriteria one-machine scheduling problems can be found elsewhere [6].

The most important literature survey for the last five years is that of Mahmood (2014) [7], who discussed the multicriteria scheduling problems which are studied on a single machine to find a set of efficient solutions for the general problems $1//F(\sum C_j, T_{max}, V_{max})$, $1//F(\sum C_j, \sum V_j, V_{max})$, $1//F(\sum C_j, \sum V_j, T_{max})$, and others. Some efficient algorithms were proposed for solving these problems. Ali (2015) [3], in his thesis, solved the sum of completion time and sum of tardiness ($\sum C_j, \sum T_j$). He proposed a new BAB, two local search methods, and a Neural Network (NN) to solve this problem. Ali and Abdul-Kareem (2017) [8], attempted to solve a single MSP to simultaneously minimize the maximum tardiness, maximum late work, and total late work. They used the exact methods (CEM and BAB), suggested some heuristic methods for a large number of jobs, and used the best results as a good upper bound for BAB. Chachan and Hameed (2019) [9] studied the multiobjective problem, which is the sum of completion time, the tardiness, the earliness, and the late work ($1//\left(\sum(C_j + T_j + E_j + V_j\right)$). They suggested the use of BAB for solving this problem, where four upper bounds and one lower bound are proposed and a number of dominance rules are considered to reduce the number of branches in the search tree.

In section two, we will discuss the mathematical formulation of $1//\left(\sum C_j, \sum E_j\right)$ problem and its special cases. Section three introduces some heuristic methods for $1//\left(\sum E_j\right)$ and our problem, while in section Four the BAB is applied with a newly suggested upper and lower bound. The practical and comparative results are introduced in section five. While in section six we present an analysis and discussion for the results which are introduced in section five. Lastly, in section seven we introduce the most important conclusions and some recommendations.

1.1 Some Important Notations

There are some notations which are used in this paper:

n	:	Number of jobs.
p_j	:	Processing time of jobs j.
d_j	:	Due date of jobs j.
s_j	:	Slack time of job j s.t. $s_j = d_j - p_j$.
C_j	:	Completion time of job j, where $C_j = \sum_{k=1}^{j} p_k$.
E_j	:	Earliness of job j, $E_j = \max\{d_j - C_j, 0\}$.
$\sum C_j$	:	Total completion time.
$\sum E_j$	:	Total earliness.
OP	:	Optimal Value of P_1-problem.
F	:	Objective Function of P-problem.
F_1	:	Objective Function of P_1-problem.

1.2 Machine Scheduling Problem

In this paper, we need some basic definitions.

Definition(1): Shortest Processing Time (SPT) rule[10]: Jobs are sequenced in a non-decreasing order of processing times (p_j), where this rule is used to solve the problem $1//\sum C_j$.

Definition(2): Earliest Due Date (EDD) rule [11]: Jobs are sequenced in a non-decreasing order of due date (d_j), where this rule is used to minimize the problem $1//T_{max}$.

Definition(3): Minimum Slack Time (MST) rule [12]: Jobs are sequenced in a non-decreasing order of slack times (s_j) s_j=$d_j - p_j$, where this rule is well known for solving the problem $1//E_{max}$.

Definition(4) [11]: The term "**optimize**" in a multi-criteria resolution-making problem indicates a solution about which there is no way of developing or improving any objective without worsening the other objective.

Definition(5) [13]: A schedule S is said to be **an efficient schedule** if we cannot found another schedule S' satisfying $f_j(S') \leq f_j(S), j = 1, \dots, k$, with at least one of the above holding as a strict inequality. Another way is that S is said to be dominated by S'.

Remark(1): Let S be a set of efficient schedules and let σ be a schedule. We use the symbol $\sigma \notin S$ if all efficient solutions of *S* do not dominates σ.

1.3 Dominance rule (DR)

Reducing the current sequence may be achieved by using several Dominance Rules (DR's). DR's usually specify some (all) parts of the path to obtain a good value for objective function so that they can be useful to determine whether a node in BAB method can be ignored before its lower bound (LB) is calculated. Clearly, DR's are particularly useful when a node can be ignored although it has a LB that is less than the optimum solution. The DR's are also useful within the BAB method to cut all nodes that are dominated by others. These improvements lead to a very large decrease in the number of nodes to obtain the optimal solution**.**

Definition(6) [**14**]**:** If *G* is a graph that has *n* vertices, then the **matrix *A*(*G*)=[a_{ij}]**, whose $i^{th}$and j^{th} elements are equal to 1 if there is at least one edge between V_i and V_j whereas they are equal to zero otherwise, this matrix is called the **adjacency matrix** of *G*, where.

$$a_{ij} = \begin{cases} 0, & \text{if } i = j \text{ or } j \nrightarrow i, \\ 1, & \text{if } i \rightarrow j, \\ a_{ij} \text{ and } \bar{a}_{ij}, & i \leftrightarrow j. \end{cases}$$

2. Mathematical Formulation for $1//(\sum C_j, \sum E_j)$ Problem

The object can be described as a set of *n* jobs *N={1,2,…,n}* on a single machine to find $\sigma \in S$ (where S is the set of all feasible schedules), so they can be fully used to specify whether that minimizes the multi-criteria $(\sum C_j, \sum E_j)$. The $1//(\sum C_j, \sum E_j)$ problem can be written as:

$$\left.\begin{array}{ll} Min\{\sum C_j, \sum E_j\} & \\ \text{Subject to,} & \\ C_j \geq p_j, & j = 1,2,\dots,n. \\ C_j = C_{(j-1)} + p_j, & j = 2,3,\dots,n. \\ E_j \geq d_j - C_j, & j = 1,2,\dots,n. \\ E_j \geq 0, & j = 1,2,\dots,n. \end{array}\right\} \quad \dots(P)$$

2.1 Sub problems of P-problem

For P-problem, we can deduce two sub problems:

1. The $1//\sum C_j + \sum E_j$ Problem:

$$\left.\begin{array}{ll} Min\{\sum C_j + \sum E_j\} & \\ \text{Subject to,} & \\ C_j \geq p_j, & j = 1,2,\dots,n. \\ C_j = C_{(j-1)} + p_j, & j = 1,2,\dots,n. \\ E_j \geq d_j - C_j, & j = 1,2,\dots,n. \\ E_j \geq 0, & j = 1,2,\dots,n. \end{array}\right\} \quad \dots(P_1)$$

The aim for the P_1- problem is to find a suitable processing order of the jobs on a one single machine to minimize the sum of completion time and sum of earliness jobs, which is a single object.

Proposition (1): Each optimal solution for P_1- problem is an efficient solution for P-problem.

Proof: Let π be an optimal schedule for P_1- problem. Suppose that π is not an efficient solution for P-problem, then there is an efficient schedule, say δ, for P-problem such that:

$\sum C_j(\delta) \leq \sum C_j(\pi)$ and $\sum E_j(\delta) \leq \sum E_j(\pi)$

where at least one of the inequalities is strict. This implies that $\sum C_j(\delta) + \sum E_j(\delta) < \sum C_j(\pi) + \sum E_j(\pi)$, then δ is a schedule that gives a better solution than π for P_1- problem. But π is an optimal schedule, which is a contradiction with our assumption, then π must be an efficient schedule for P-problem.

2. The $1//Lex(\sum C_j, \sum E_j)$ Problem

$$\left.\begin{array}{ll} Min\{\sum E_j\} & \\ Subject\ to & \\ C_j \geq p_j, & j = 1,2,\dots,n. \\ C_j = C_{(j-1)} + p_j, & j = 2,3,\dots,n. \\ E_j \geq d_j - C_j, & j = 1,2,\dots,n. \\ E_j \geq 0, & j = 1,2,\dots,n. \\ \sum C_j = \Delta\ , where\ \Delta = \sum C_j(SPT) & \end{array}\right\} \quad \dots(P_2)$$

For the P_2-problem, the objective $\sum C_j$ is more important than the other objectives since the multi-criteria object is $Lex(\sum C_j, \sum E_j)$.

2.2 Special Cases For P-problem

Case (1): If $d_j = d$, $\forall j$, then the sequence σ obtained by SPT rule gives an efficient solution for P-problem.

Proof: Since $d_j = d$ and since p_j are different, then the sequence σ obtained by SPT rule gives an optimal schedule for $\sum C_j$, then the sequence σ gives an efficient solution for P-problem.

Case (2): If $p_j \geq d_j, \forall j$, (and automatically for $C_j \geq d_j$), then the sequence σ obtained by SPT rule gives:

1. A unique efficient solution for different p_j, such that $(min\sum C_j(\sigma),0)$.
2. If there is m similar p_i in the schedule, then the SPT rule gives $m!$ optimal solution for $\sum C_j$, and obtains m! efficient solutions for P-problem.

Proof: Since $p_j \geq d_j$, (and automatically $C_j \geq d_j$) $\forall_j$, then $E_j = 0$, $\forall j$, and $\sum E_j = 0$, then it is constant.

1. If all p_j are different, then σ gives an optimal solution for $\sum C_j$, and gives a unique efficient solution for P-problem.
2. If p_j is similar to $m \geq 2$, then σ gives an optimal solution for $\sum C_j$, and σ gives $m!$ efficient solutions for P-problem.

Case (3): If $d_j \geq C_{max}$, for some j, then the job j is sequenced last, then we obtain an efficient solution with SPT rule for other jobs in the schedule for P-problem.

Proof: Since $d_j \geq C_{max}$, if the job j is sequenced last in some order, then if $E_{\sigma(j)} = 0$, then the sequence σ gives an efficient solution for P-problem.

Theorem (1): In general, there exist efficient solution(s) for P-problem that satisfies the SPT rule.

Proof:

1. First, suppose that all processing times (p_j) are different. Then their exists a unique SPT sequence (SPT*) that gives the unique value of minimum of $\sum C_j$. Hence, there is no sequence $\sigma \neq$ SPT* such that:

$$\sum_{j=1}^{n} C_j(\sigma) \leq \sum_{j=1}^{n} C_j(SPT^*)\ and\ \sum_{j=1}^{n} E_j(\sigma) \leq \sum_{j=1}^{n} E_j(SPT^*) \quad \dots(1)$$

with at least one strict inequality.

2. If many jobs have equal processing times, then more than one SPT sequence exists. Let SPT* be one sequence satisfying the SPT rule, and the jobs which have equal processing times are ordered in EDD rule in order to minimize $\sum E_j$(SPT*). Then we have to prove that each SPT* sequence is efficient. It is clear that any sequence that do not satisfy the SPT rule cannot dominate an SPT* sequence, as in relation (1). So if σ is an SPT but not SPT* sequence then it cannot dominate SPT* since:

$$\sum_{j=1}^{n} C_j(\sigma) = \sum_{j=1}^{n} C_j(SPT^*) \text{and } \sum_{j=1}^{n} E_j(\sigma) \leq \sum_{j=1}^{n} E_j(SPT^*) \quad \dots(2)$$

Hence all SPT* sequences give efficient solutions for P-problem.

3. Heuristic Methods for P and P_1-Problems

As most of the scheduling problems are NP-hard and the computational requirement to solve such problems using BAB or DP methods might require more time, many researchers have developed heuristic algorithms to solve them in an efficient and effective way.

The heuristic method is defined as follows (Reeves [15]): A heuristic is a technique that seeks a good (i.e. optimal or near optimal) solution at a reasonable computational cost without guarantee for either feasibility or optimality, or even in many cases, to state how close this solution is to optimality in a particular feasible solution?

In the next section, we discuss some heuristic methods for $(1//\sum E_j)$, P and P_1-problems.

3.1 Heuristic Methods for $1//\sum E_j$

The problem $\mathbf{1//\sum E_j}$ is considered as NP-hard, so we suggested two heuristic methods to solve this problem that give good results. The first suggested heuristic method is using MST for sum of earliness (MST-SE). The idea of this method is to arrange the jobs by MST rules and calculate the objective function, then putting the second job in the first place, while the other jobs are arranged by the MST rules and the objective function is calculated. We continue this work until obtaining the *n* sequences. The main steps of MST-SE (**mst rule-sum of earliness**) algorithm are as follows:

Algorithm (1): MST_SE Heuristic Method.

Step(1): INPUT n, p_j and $d_j, j = 1,2,3, \dots, n$.

Step(2): Arrange jobs in MST rule (γ_1), and calculate $F_1(\gamma_1) = \sum E_j(\gamma_1)$, let $\gamma = \gamma_i$.

Step(3): FOR $i=2,\dots,n$, job i in the first position of $\gamma_{i-1}$to obtain γ_i, then calculate $F_i(\gamma_i)$.

Step(4): If $F_i(\gamma_i) \leq F(\gamma)$ THEN $\gamma = \gamma_i$

ELSE GO TO Step(3)

END IF.

Step(5): OUTPUT: The options of sequence γ with $F(\gamma)$ value.

Step(6): END.

The second method depends on using DR of Sum Earliness (DR-SE).

Remark (2) [16]: For $1//E_{max}$ problem, if $p_i \leq p_j$ and $s_i \leq s_j$, then there exists an efficient solution in which job *i* is sequenced before job *j*.

Remark (1) may be useful to obtain a good solution for $1//\sum E_i$ problem.

Example (1): Let's have the following data for *n*=4:

n	1	2	3	4
p_i	6	1	3	4
d_i	25	7	8	6
s_i	19	6	5	2

We obtain the following DR: 2→1, 3→1, 4→1, then the adjacency matrix:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & a_{23} & a_{24} \\ 1 & \bar{a}_{23} & 0 & a_{34} \\ 1 & \bar{a}_{24} & \bar{a}_{34} & 0 \end{bmatrix}$$

The optimal solution is $\sum E_i = 14$ with the sequence 4 , 3 , 2 , 1.

The DR-SE method is summarized by finding a sequence sort with a minimum s_j which is not in a contradiction with DR and then the objective function will be calculated. The main steps of DR-SE (**Dominance rule-sum of earliness**) are as follows:

Algorithm (2): DR_SE Heuristic Method.

Step(1): INPUT: n, p_j and $d_j, j = 1,2, \dots, n$.

Step(2): Apply remark (2.1) to find adjacency matrix A of DR, let $\sigma = \emptyset$. $N = \{1,2, \dots, n\}$. Calculate $s_j = d_j - p_j, \forall j \in N$. Let σ be any sequence.

Step(3): For i=1,2,…,n, find a sequence σ with minimum s_j which is not a contradiction with DR (matrix A), if ∃ is more than one job that breaks the tie arbitrarily.

Step(4): If $F_i(\sigma_i) \leq F(\sigma)$, THEN $\sigma = \sigma_i$

ELSE GO TO Step(3)
END.
Step(5): OUTPUT: The optioned sequence σ with $F(\sigma)$ value.
Step(6): END.

3.2 Heuristic Method for P and P_1- problems

The heuristic methods for P and P_1-problems are considered as a development of the heuristic methods for $1//\sum E_j$ problem.

3.2.1 Heuristic Methods for P-problem

The first heuristic method depends on SPT and MST. Since the SPT rule solves the $1//\sum C_j$ problem, we have improved the heuristic method MST-SE by ordering the jobs by the SPT rule and then calculating the objective function. Then, the second job was put in the first place and the other jobs were still arranged by SPT rule. The objective function was calculated, and so on until obtaining the n sequences, The main steps of SPT-MST-SCSE(F) (**spt rule-mst-rule-sum of completion time and sum of earliness for function F**) are as follows:

Algorithm (3): SPT_MST_SCSE(F) Heuristic Method.
Step(1): INPUT n, p_j and $d_j, j = 1,2,3,\dots,n$, $\delta = \phi$.
Step(2): Arrange jobs in SPT rule (σ_1) and calculate $F_{11}(\sigma_1) = \left(\sum C_j(\sigma_1), \sum E_j(\sigma_1)\right)$;
$\delta = \delta \cup \{F_{11}(\sigma_1)\}$.
Step(3): FOR $i=2,\dots,n$, put job i in the first position of $\sigma_{i-1}$to obtain σ_i and calculate $F_{1i}(\sigma_i) = \left(\sum C_j(\sigma_i), \sum E_j(\sigma_i)\right)$;
$\delta = \delta \cup \{F_{1i}(\sigma_i)\}$.
END;
Step(4): Arrange jobs in MST rule (π_1) and calculate $F_{21}(\pi_1) = \left(\sum C_j(\pi_1), \sum E_j(\pi_1)\right)$;
$\delta = \delta \cup \{F_{21}(\pi_1)\}$.
Step (5): FOR $i=2,\dots,n$, put job i in the first position of $\pi_{i-1}$to obtain π_i and calculate $F_{2i}(\pi_i) = \left(\sum C_j(\pi_i), \sum E_j(\pi_i)\right)$;
$\delta = \delta \cup \{F_{2i}(\pi_i)\}$.
END;
Step(6): A filter set δ to obtain a set of efficient solutions of *P*-problem.
Step(7): OUTPUT The set of efficient solutions δ.
Step(8): END.

The idea of the second heuristic method is dependent on the heuristic DR-SE and it is summarized by finding a sequence sort with minimum p_j, d_j and s_j which is not a contradiction with DR, and calculating the objective function. The main steps of DR-SCSE(F) (**Dominance rule-sum of completion time and sum of earliness for function F)** are as follows:

Algorithm (4): DR_SCSE(F) Heuristic Method.
Step(1): INPUT: n, p_j and $d_j, j = 1,2,\dots,n$.
Step(2): Apply remark(1) to find DR and adjacency matrix A;
$\sigma = \emptyset, N = \{1,2,\dots,n\}$.
Calculate $s_j = d_j - p_j, \forall j \in N, \delta = \emptyset$.
Step(3): Find a sequence σ_1 with minimum p_j which is not contradiction with DR (matrix A), if ∃ is more than one job that breaks ties arbitrarily, $\delta = \delta \cup \{\sigma_1\}$.
Step(4): Find a sequence σ_2 with minimum d_j which is not contradiction with DR (matrix A), if ∃ is more than one job that breaks ties arbitrarily, $\delta = \delta \cup \{\sigma_2\}$.
Step(5): Find a sequence σ_3 with minimum s_j which is not contradiction with DR (matrix A), if ∃ is more than one job that breaks ties arbitrarily, $\delta = \delta \cup \{\sigma_3\}$.
Step(6): Find the dominated sequence set δ' from δ.
Step(7): Calculate $F(\delta)$.
Step(5): OUTPUT The set of efficient solution δ.
Step(8): END.

3.2.2 Heuristic Methods for P_1-problem

For P_1-problem, we can use the same two heuristic methods which are discussed in section 3.2.1. The Tree Type Heuristic Method (TTHM) can be considered as the 3rd heuristic method for P_1-problem.

Algorithm (5): TTHM(F_1)

Step(1): INPUT: n, p_j and $d_j, j = 1,2, \dots, n$.

Step(2): CALL SPT_MST_SCSE(F_1);

Calculate UB =min(F_{1i}), obtained by a schedule, say τ_i. Compute $\sum C_j(\tau_i)$ and $\sum E_j(\tau_i)$ s.t. UB=$\sum C_j(\tau_i)$+$\sum E_j(\tau_i)$. *i=0.*

Step(3): *i*=*i*+1, for any node in the search tree calculate the lower bound (LB)=cost of sequencing jobs + cost of unsequencing jobs, where the cost of unsequencing jobs is obtained by σ=SPT rule.

Step(4): Branch from the minimum node with LB $\leq$ UB for level *i.*

Step(5): If $i < n$ then go to **Step(3).**

Step(6): At the last level (*i*=*n*) we get the best solution (BS).

Step(7): OUTPUT The Best solution (BS).

Step(8): END.

4. Branch and Bound Method for P and P_1-problems

BAB method is one of the most effective exact ways to solve scheduling problems, but it is classically difficult because of the elapsed time.

We notice that P-problem has many similar efficient solutions. This implies when we apply BAB for P-problem. Many nodes are open and this implies that the BAB may be not efficient to obtain an accurate solution for large *n* in a reasonable time. Table-1 describes this problem.

4.1 Branch and Bound Method for P-problem

We tested three models of upper bound (UB); let σ_1=SPT, σ_2=EDD, σ_3=MST, then UB_1= F(σ_1), UB_2= F(σ_2), and UB_3= F(σ_3) UB=min{UB_1,UB_2,UB_3}. We also tested two models of lower bound (LB); let γ_1=SPT and γ_2=MST, then LB_1=F(γ_1) and LB_2=F(γ_2).

Algorithm (6): BAB(F)

Step(1): INPUT: n, p_j and $d_j, j = 1,2, \dots, n$.

Step(2): Find the UB=UB_k by σ_k s.t. UB=($\sum C_j(\sigma_k)$,$\sum E_j(\sigma_k)$), (*k*=1,2,3), *i=0.*

Step(3): Set the upper bound *S*={ σ_k }.

Step(4): *i*=*i*+1, in level *i*, for any node in the search tree, compute the LB=cost of sequencing jobs +cost of unsequencing jobs, where the cost of unsequencing jobs is obtained by γ_m, (*m*=1,2) for the P-problem.

Step(5): Check that the sequence γ_m of this node is $\gamma_n \notin S$ if $\gamma_m \not\supset S$, $S = S \cup \{\gamma_m\}$, cancel the dominated sequence, and branch from this node; else ignore this node.

Step(6): After finishing all nodes of this level, find the efficient solutions s_i of *S*, say S', such that $S' \subseteq S$.

Step(7): State that UB=$\{F(s_i)\}$, $\forall s_i \in S'$.

Step(8): GO TO Step(3) until finishing checking all levels *(i=n).*

Step(9): Calculate $O_i = F(s_i)$, for $s_i \in S'$, for *i=1,…, k (k* is the number of the efficient points).

Step(10): OUTPUT The set of efficient solution S'.

Step(11): END.

4.2 Branch and Bound Method for P_1-problem

For the P_1-problem, we applied what is known as the classical BAB to obtain optimal sequence τ=SPT since we search about the single objective function. Firstly, UB= $\sum C_j(\tau) + \sum E_j(\tau)$. Then calculate the LB for any node which consists of sequence and unsequence parts (obtained by SPT rule). Repeat these steps until obtaining the optimal solution at the root. The BAB algorithm steps for P_1-problem are as follows:

Algorithm (7): BAB(F_1)

Step(1): INPUT: n, p_j and $d_j, j = 1,2, \dots, n$.

Step(2): Find the UB by τ=SPT, obtain schedule say τ compute $\sum C_j(\tau)$ and $\sum E_j(\tau)$ then UB=$\sum C_j(\tau)$+$\sum E_j(\tau)$.

Step(3): Set the UB *S*=$\{\tau\}$, when *i=0.*

Step(4): $i=i+1$, for any node in the search tree calculate the LB=cost of sequencing jobs + cost of unsequencing jobs, where the cost of unsequencing jobs is obtained by σ=SPT rule.
Step(5): Branch each node with LB $\leq$ UB for level *i.*
Step(6): If $i < n$ then go to **Step(4).**
Step(7): At the last level ($i=n$) we get the optimal solution (OP).
Step(8): OUTPUT The optimal solution (OP).
Step(9): END.

5. Practical Results of P and P_1-problems

The values of p_j and d_j for all example are generated randomly s.t. $p_j \in$ [1,10] and

$$d_j \in \begin{cases} [1,30], & 1 \leq n \leq 29 \\ [1,40], & 30 \leq n \leq 99 \\ [1,50], & 100 \leq n \leq 999 \\ [1,70], & \text{otherewise,} \end{cases}$$

under condition $d_j \geq p_j$, for j=1,..,n.

Before showing tables of all the results, we introduce some important abbreviations:

Ex	:	Example Number.
Av	:	Average.
NES	:	Number of efficient Solution.
ANES	:	Average number of efficient solution.
T/S	:	Time per second.
AT/S	:	Average Time per second.
AAE	:	Average Absolute Error.
MAE	:	Mean Absolute Error s.t. MAE=$\lvert a_1-a_2\rvert/a_1$.
AMAE	:	Average of MAE.
T(Av)	:	Total Average.
AE(Av)	:	Absolut error (Average).
SOF	:	Single Objective Function.
ASOF	:	Average Single Objective Function.
MOF	:	Multi Objective Function.
AMOF	:	Average Multi Objective Function.
R	:	0 < Real < 1.

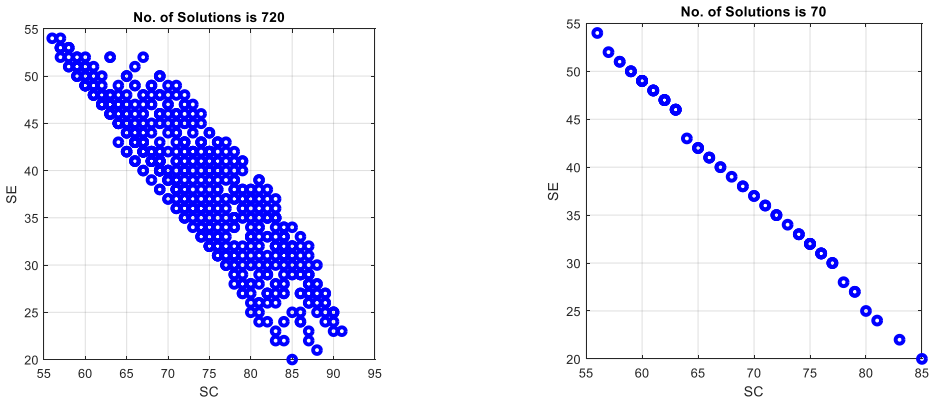
It is important to mention that the results of applying all solving methods are revered for 5 experiments.

For P-problem, we note that the number of similar feasible and efficient solutions is very large. This number will be increased as n is increased, because of the function $\sum \boldsymbol{E_i}$. Table-1 shows the number of feasible solutions (NF) as well as the averages and percentages of similar (SS) and different (DS) feasible solutions, and the number of the similar (NESS) and different (NEDS) efficient solutions for different n.

Table 1-NF and the averages and percentages of SS, DS, NESS and NEDS for different n

n	NF	Av(SS)	P(SS)%	Av(DS)	P(DS)%	Av(NESS)	Av(NEDS)
4	24	0.8	3.3	23.2	96.7	0.4	5.6
5	120	23.8	19.8	96.2	80.2	8.4	14.2
6	720	389	54	331	46	20.4	14.8
7	5040	3795.4	75.3	1244.6	24.7	26.8	21.2
8	40320	38015.6	94.3	2304.4	5.7	76.8	26
9	362880	359262.6	99	3617.4	1	132.6	29.6
10	3628800	3622601	99.8	6199	0.2	---	43.6

Notice that the P(SS) and Av(NESS) are increased as n increased.
Figure-1 Shows the difference between feasible and efficient solutions for n=6, where the curve of the efficient solutions is approximate to $f(x)= -x$.



(a) (b)

Figure 1-(a) All feasible solutions. (b) Efficient solutions for n=6.

5.1 Comparison Results of $1//\sum E_j$

Table-2 shows a comparison between the results of the heuristics MST_SE and DR_SE compared with the CEM method to solve $1//\sum E_j$ problem, *n*=4:10.

Table 2-Comparison of the results of MST-SE, DR-SE with CEM for $1//\sum E_j$ problem, *n*=4:10.

N*n*	CEM		MST-SE			DR-SE		
	ASOF	AT/S	ASOF	AT/S	AAE	ASOF	AT/S	AAE
4	14.6	R	8.6	R	0.41	8.8	R	0.39
5	27	R	11.8	R	0.56	11.8	R	0.56
6	25.6	R	6.4	R	0.75	7.2	R	0.72
7	21.6	R	7.4	R	0.66	9.2	R	0.57
8	40.2	1.5	6.2	R	0.85	6.4	R	0.84
9	30	12.04	5.6	R	0.81	5.8	R	0.81
10	28	124.6	4	R	0.86	12.4	R	0.56
Av	26.71		7.14		0.70	8.8		0.64

where the AASE is the average of the averages of SE.

Table-3 shows a comparison of the results between heuristics MST_SE and DR_SE methods to solve $1//\sum E_j$ problem for *n*=30,70,100,300,700 and 1000.

Table 3-comparison of the results between MST_SE and DR_SE methods for $1//\sum E_j$ problem for *n*=30,70,100,300,700 and 1000.

n	MST-SE		DR-SE	
	ASOF	AT/S	ASOF	AT/S
30	2	R	2.6	R
70	1.2	R	1.2	1
100	1.4	R	2.2	2
300	1	1.2	1.6	14
700	0	5.6	1.6	71
1000	0	11.11	1.4	148

5.2 Comparison of the Results of P-problem.

Comparison of the results between SPT-MST-SCSE(F) and DR-SCSE(F) with efficient results of CEM(F) for P-problem are shown in Table-4, *n*=4:10.

Table 4-Comparison between SPT-MST-SCSE(F), DR-SCSE(F) with CEM(F) for P-problem, *n*=4:10.

n	CEM(F)			SPT-MST-SCSE(F)				DR-SCSE(F)			
	AMOF	AT/S	ANES	AMOF	AT/S	ANES	AAE	AMOF	AT/S	ANES	AAE
4	(58.6,16.3)	R	5.6	(59.4,17)	R	4.8	(0.01,0.04)	(58.6,14.8)	R	2.6	(0.0,0.09)
5	(72.8,24.4)	R	14.2	(73.7,25.6)	R	7.2	(0.01,0.05)	(73.8,24.5)	R	2	(0.01,0.004)
6	(93.4,19.8)	R	14.8	(93.7,22.3)	R	6.8	(0.003,0.13)	(97.1,16.9)	R	2.8	(0.04,0.15)
7	(134.6,24.5)	R	21.2	(140,26.5)	R	9.4	(0.04,0.08)	(146.2,21.3)	R	2.6	(0.09,0.13)
8	(143.3,27.5)	R	26	(148.8,32.5)	R	9	(0.04,0.18)	(155.3,24.6)	R	3	(0.08,0.11)
9	(199.3,28.2)	4.4	29.6	(215.8,31.1)	R	11	(0.08,0.10)	(217.7,26.1)	R	2.6	(0.09,0.07)
10	(235.8,34.1)	43.1	43.6	(256.4,37.7)	R	12.2	(0.09,0.11)	(277,26.7)	R	3	(0.18,0.22)
	AE(AV)		22.143			8.628	(0.04,0.10)		R	2.66	(0.07,0.11)

Notice that the Heuristics SPT-MST-SCSE(F) and DR-SCSE(F) yield good results compared with CEM(F) for P-problem.

The results of applying BAB(F) compared with CEM(F) for P-problem, *n*=4:10 are shown in Table -5.

Table 5-Comparison between BAB(F) with CEM(F) for P-problem, *n*=4:10.

n	CEM(F)			BAB(F)		
	AMOF	AT/S	ANES	AMOF	AT/S	ANES
4	(58.6,16.3)	R	5.6	(58.8,16.0)	R	5
5	(72.8,24.4)	R	14.2	(72.8,24.4)	R	13.4
6	(93.4,19.8)	R	14.8	(93.2,20.1)	R	14.2
7	(134.6,24.5)	R	21.2	(135.6,23.9)	R	18.2
8	(143.3,27.5)	R	26	(144.4,27.0)	R	22.6
9	(199.3,28.2)	4.4	29.6	(199.7,28.2)	R	26.4
10	(235.8,34.1)	43.1	43.6	(238.7,31.8)	R	39
TAV			22.1429			19.8286
AE(AV)			-----			0.1045

The comparison of the results of CEM(F) and BAB(F) which are shown in Table-5, for P-problem, *n*=4:10, are depicted in Figure-2.

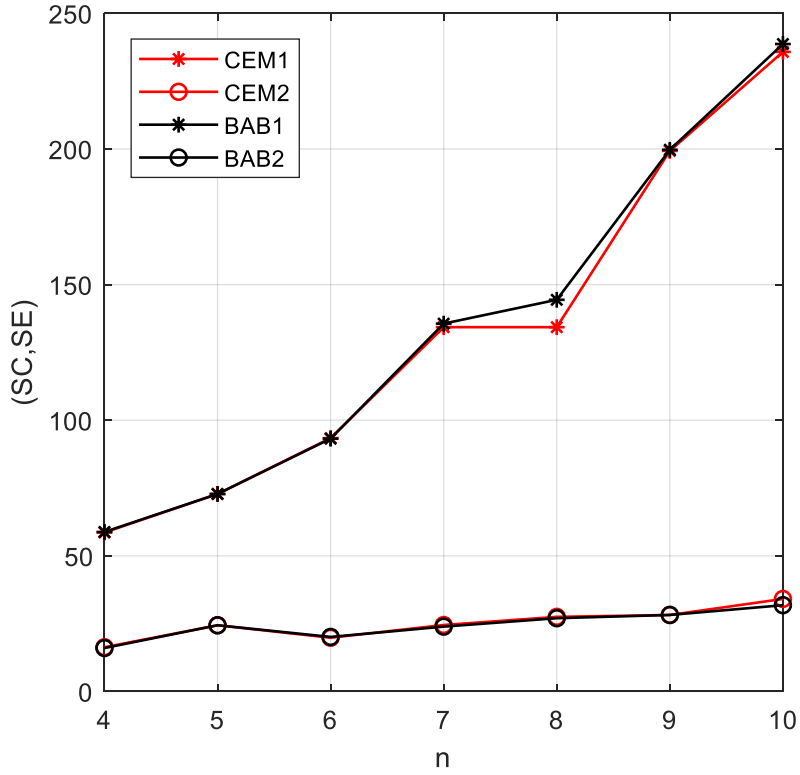


Figure 2-Comparison of the results of CEM(F) and BAB(F) for P-problem, *n*=4:10.

Table-6 shows a comparison of the results of CEM(F) and BAB(F) models NS for P-problem for n=4:10.

Table 6-Comparison of the results of CEM(F) and BAB(F) methods for NS for P-problem, *n*=4:10.

n	**AV(NES) Using CEM**	**AV(NES) using BAB**		
		UB=F(SPT) LB=F(SPT)	**UB=F(SPT) LB=F(MST)**	**UB=min(F(SPT),F(EDD),F(MST)) LB=F(SPT)**
4	5.6	5.0	5.2	5.4
5	14.2	13.2	12.2	13.0
6	14.8	13.6	10.6	14.2
7	21.2	15.0	13.2	17.6
8	26.0	17.0	14.0	21.8
9	29.6	19.0	12.4	26.4
10	43.6	33.0	8.0	41.2
TAV	22.143	16.543	10.8	19.943
AE(AV)	--------	0.253	0.512	0.099

From the above table, we conclude that the best model is UB=min{F(SPT), followed by F(EDD), F(MST)}, then LB=F(SPT).

In Table-7, we compare the results obtained from heuristic SPT-MST-SCSE(F), DR-SCSE(F) and BAB(F) for P-problem, *n*=11:16.

Table 7-Results of the comparison of BAB(F), SPT-MST-SCSE(F) and DR-SCSE(F) for (P), *n*=11:20.

n	BAB(F)		SPT-MST-SCSE(F)		DR-SCSE(F)	
	AMOF	AT/S	AMOF	AT/S	AMOF	AT/S
11	(286,28.2)	0.7	(312,33.5)	R	(329.2,25.5)	R
12	(366.2,18.1)	0.4	(386.8,26.4)	R	(408.1,17.3)	R
13	(441.3,28.9)	1	(487.4,34.2)	R	(525.5,22.1)	R
14	(452,29.9)	0.9	(519.2,33.8)	R	(556.3,25.7)	R
15	(531.7,33.2)	6.3	(581.4,38.9)	R	(636,25.7)	R
16	(660.4,24.9)	11.4	(732.6,30.4)	R	(766.3,22.4)	R
17	(657,37.7)	21.5	(766.3,41.8)	R	(823.7,28.1)	R
18	(729,36.8)	56.4	(829.2,44.2)	R	(871.3,32.9)	R
19	(855.5,30.6)	128.5	(980.9,35.01)	R	(1030.5,25.7)	R
20	(919.9,31)	211.5	(1062.2,37.8)	R	(1127.6,26.2)	R

Table-8 shows a comparison of the results between heuristics methods SPT-MST-SCSE(F) and DR-SCSE(F) to solve P-problem for *n*=30,70,100,300,700 and 1000.

Table 8-Comparison of the results between SPT-MST-SCSE(F) and DR-SCSE(F) for P-problem for different *n*.

n	SPT-MST-SCSE(F)		DR-SCSE(F)	
	AMOF	AT/S	AMOF	AT/S
30	(2343.8,56.2)	0.2	(2569.9,36.9)	0.3
70	(11985.8,96.5)	0.3	(12673.4,68.9)	0.3
100	(24316.2,155.01)	0.3	(25484.2,109)	0.3
300	(241361.9,123.1)	1	(235964.5,129.7)	0.7
700	(1286395.7,98.9)	6.7	(1221127.3,24.7)	3.5
1000	(2681881.1,160.8)	16.8	(2544587.7,26.9)	7.7

5.3 Comparison of the Results of P_1-problem

In Table-9, we show a comparison between the optimal results of CEM(F_1) and the results of the heuristics SPT-MST-SCSE(F_1), DR-SCSE(F_1) and TREE(F_1) *n*=4:10 for P_1-problem.

Table 9-Comparison between CEM(F_1) and SPT-MST-SCSE(F_1), DR-SCSE(F_1) and Tree(F_1) for P_1-problem, *n*=4:10.

n	CEM(F_1)		SPT-MST-SCSE(F_1)			DR-SCSE(F_1)			TREE(F_1)		
	ASOF	AT/S	ASOF	AT/S	AAE	ASOF	AT/S	AAE	ASOF	AT/S	AAE
4	73	R	73.4	R	0.01	73.4	R	0.01	73.4	R	0.01
5	95.6	R	96.8	R	0.01	96.6	R	0.01	96.8	R	0.01
6	110.4	R	113.2	R	0.03	112.6	R	0.02	112.4	R	0.02
7	156	R	160.8	R	0.03	161	R	0.03	160.2	R	0.03
8	166.8	R	171.8	R	0.03	172.8	R	0.04	171.8	R	0.03
9	224.4	4.5	231.4	R	0.03	231.6	R	0.03	229.6	R	0.02
10	267.4	46.2	271.2	R	0.01	274.2	R	0.03	271	R	0.01
AE(AV)	156.229		159.8		0.02	160.31		0.02	159.31		0.02

Notice that the heuristics Tree (F_1), SPT-MST-SCSE(F_1) and DR-SCSE(F_1) produced good results compared with CEM(F_1), and this is obvious from AAE, for P_1-problem.

Table-10 shows the results of CEM(F_1) for P_1-problem compared with those of BAB(F_1) and Tree(F_1) methods for P_1-problem, n=4:10.

Table 10-Comparison of the results of BAB(F_1), Tree(F_1) with CEM(F_1) for (P_1), n=4:10.

n	CEM(F_1)		BAB(F_1)		TREE(F_1)	
	ASOF	AT/S	ASOF	AT/S	ASOF	AT/S
4	73	R	73	R	73.4	R
5	95.6	R	95.6	R	96.8	R
6	110.4	R	110.4	R	112.4	R
7	156	R	156	R	160.2	R
8	166.8	R	166.8	R	171.8	R
9	224.2	4.5	224.2	1.3	229.6	R
10	267.4	46.2	267.4	2.8	271	R
TAV	156.2		156.2		159.31	

Where TAV is the total average.

The results of applying BAB(F_1) and the three heuristics for P_1-problem, n=11:15, which are given in a reasonable CPU time (Time ≤ 600 seconds), are described in Table-11.

Table 11-The results of the comparison of BAB(F_1), with three heuristics for (P_1), for n=11:15.

n	BAB(F_1)		TREE(F_1)		SPT-MST-SCSE(F_1)		DR-SCSE(F_1)	
	ASOF	AT/S	ASOF	AT/S	ASOF	AT/S	ASOF	AT/S
11	309.4	0.4	313.2	R	316.2	R	318.8	R
12	379	1.2	384.6	R	387.6	R	390.6	R
13	465.4	6.6	468.8	R	473.6	R	475.8	R
14	476.6	45.4	481.2	R	485	R	486.6	R
15	559	66.2	564.8	R	567.2	R	571.8	R
TAV	437.88		442.52		445.92		448.72	

Figure-3 shows the comparison of the results of BAB(F_1), Tree(F_1) and SPT-MST-SCSE(F_1) which are obtained from Table-11 for P_1-problem, for n=4:15.

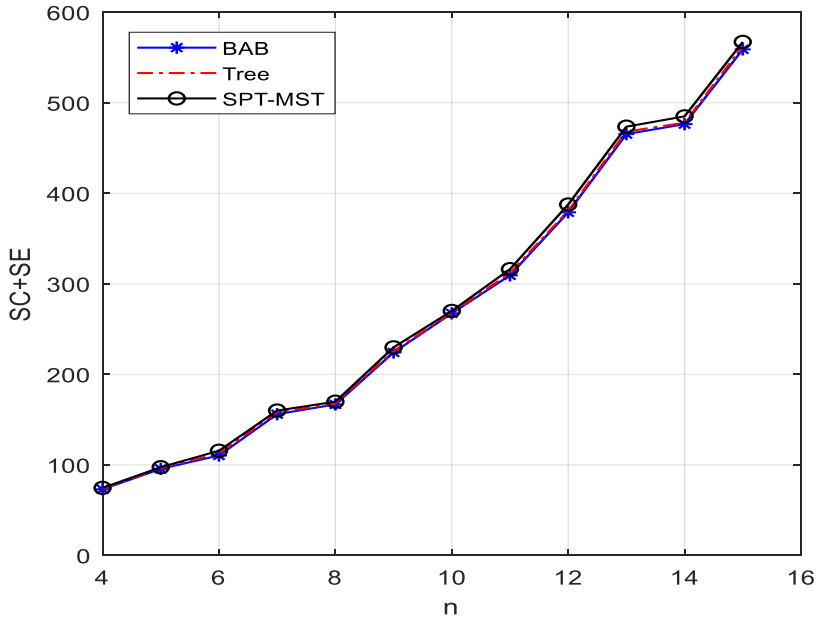


Figure 3-Comparison of the results of BAB(F_1), Tree(F_1) and SPT-MST-SCSE(F_1) for n=4:15.

Table-12 describes the average of the efficient solution for P_1-problem for *n*=20:10:100;200:100:1000, using Tree(F_1), compared with the heuristic methods SPT-MST-SCSE(F_1) and DR-SCSE(F_1).

Table 12-Results of the comparison of Tree(F_1) with SPT-MST-SCSE(F_1) and DR-SCSE(F_1) for (P_1), *n*=20:10:100;200:100:1000.

n	TREE(F_1)		SPT-MST-SCSE(F_1)		DR-SCSE(F_1)	
	ASOF	AT/S	ASOF	AT/S	ASOF	AT/S
20	950.6	R	959	R	959	R
30	2065.4	R	2079.4	R	2083.6	R
40	3269.2	R	3297.4	R	3299.8	R
50	4803.8	R	4836	R	4828.6	R
60	7192	R	7225.2	R	7220.8	R
70	10007.4	R	10043.8	R	10038.6	R
80	13311.4	R	13349	R	13345.8	R
90	16617	R	16659.6	R	16644	R
100	19929.8	R	20006.2	R	19995.4	R
200	78921.4	4	79030.2	0.5	78989.8	R
300	180740.2	11.4	180888.2	0.8	180842.8	R
400	306064	19.1	306284	0.9	306119.6	R
500	482286	37.9	482609.4	1.2	482311.4	R
600	703009	58.5	703309.6	1.3	703043.2	R
700	921031.6	81.4	921374	1.7	921031.6	R
800	1243269	120.7	1243590.8	2	1243269	1.3
900	1583756	175.8	1584086.2	2.3	1583756	1.7
1000	1934981.8	234.4	1935622.4	2.8	1934981.8	2.3

6. Analysis and Discussion of Results

1. For $1//\sum E_j$ problem, we noticed that MST-SE is better than DR-SE for $n \leq 1000$, both in accuracy and CPU-time (see Tables-2 and 3).
2. For P-problem:

a. The SPT-MST-SCSE (F) is better than DR-SCSE(F) in accuracy and CPU-time (see Tables-4,7 and 8) for all *n***.**

b. We notice that BAB(F) has a good accuracy in a reasonable time for $n \leq 20$ (see Tables-10 and 11)**.**

3. For P_1-problem:

a. In accuracy, we ensure that TTHM(F_1) is the best method, but SPT-MST-SCSE(F_1) is better than DR-SCSE(F_1) for $n \leq 40$. DR-SCSE(F_1) is better for $40 \leq n \leq 1000$, while it is the best as related to CPU-time (see Tables-9, 11 and 12).

b. For $n \leq 15$, the BAB(F_1) gives accurate results in a reasonable time (see Tables-10 and 11).

7. Conclusions

From this paper, we obtained the following conclusions:

1. New different heuristics are suggested to solve $1//\sum E_j$ problem for one machine, where the experimental results proved that the MST-SE and DR-SE methods produced good results.
2. We develop the heuristic methods of $1//\sum E_j$ for convenient $1//(\sum C_j, \sum E_j)$ and we obtained two new heuristics, SPT-MST-SCSE and DR-SCSE, with good performance.
3. For P_1-problem, we used SPT-MST-SCSE(F_1) and DR-SCSE(F_1) and suggest the use of the 3rd heuristic method (TTHM) which gives good results for $n \leq 1000$.
4. For some examples, when the solving methods are applied, there are some extreme CPU times, especially when applying BAB for P1-problem. This is because of the similarity in pi or/and di which causes a large number of nodes that must be branched.
5. As a future work, we suggest the use of local search methods (such as simulated annealing, particle swarm optimization, genetic algorithm, Bees algorithm,…, etc.) to find efficient approximation solutions for P and P_1 problem for $n > 100$.

References

1. Hoogeven, J. A. **2005**. "Invited Review Multicriteria Scheduling", *European Journal of Operation Research*, **167**: 592-623.
2. Blazewics, J., K.H.P. Ecker, E.Esch, G.Schmidt and J.Weglarz, **1996**. "*Scheduling Computer and Manufacture Processes*", Spring Verlag Berlin. Heidelberg.
3. Faez H. Ali, **2015**. "Improving Exact and Local Search Algorithms for Solving Some Combinatorial Optimization Problems", Ph. D., Thesis, Mustansiriyah University, College of Science, Dept. of Mathematics.
4. Tariq, S., Abdul-Razaq and Faez H. **2013**. "Algorithm for Scheduling a Single Machine to Minimize Total Completion Time and Total Tardiness", The 2nd International Conference on Mathematical Sciences-ICMA, 23-24.
5. Van Wassenhove, L.N. and Gelders, F**. 1988**. "Solving a Bi-Criteria Scheduling Problem", *European Journal of Operation Research*, **4/1**: 42-48.
6. Hoogeveen, J.A. **1992**. "Single-machine bi-criteria scheduling", Ph. D. Dissertation, Center for mathematics and Computer science, Amsterdam. The Netherlands.
7. Mahmood A. A. **2014**. "Multicriteria Scheduling: Mathematical Models, Exact and Approximation Algorithms", Ph.D. Thesis, University of Al-Mustansiriyah, College of Science, Dept. of Mathematics.
8. Faez H. and Shrmeen B. **2017**. "Scheduling a Single Machine to Minimize Max Tardiness, Max Late Work and Total Late Work". *Mathematics and Statistics Journal*, **3**(1): 1-17.
9. Hanan A. and Alaa S. **2019**. "*Exact Methods for Solving Multi-Objective Problem on Single Machine Scheduling*", *Iraqi Journal of Science*, 2019, **60**(8): 1802-1813, DOI: 10.24996/ijs.2019.60.8.17.
10. Smith W.E. **1956**. "Various Optimizers for Single Stage Production", *Naval Research Logistics Quarterly*, **3/1**: 59-66.
11. Jouni L. **2000**. "Multi-Objective Nonlinear Pareto-Optimization" *Lappeenranta University of Technology*.
12. Hoogeveen J.A. **1991**. "*Minimizing maximum earliness and maximum lateness on a single machine*", Center for Mathematics and Computer science, P.O. Box 4079, 1009 AB Amsterdam, The Netherland.
13. Hoogeveen J. A. **1996**. "Single Machine Scheduling to Minimize a Function of Two or Three Maximum Cost Criteria", *Journal of Algorithms*, **21**: 415-433.
14. Kolman, B. **1988**. "*Introductory Linear Algebra with Applications*", Macmillan Publishing company.
15. Reeves C. R. **1993**. "*Modern Heuristic Techniques for Combinatorial Problems*", John Wiley and sons, New York.
16. Tariq, S. and Sally A. **2016**. "A Comparison of Local Search Algorithm for Multicriteria Scheduling Problems", M.Sc. thesis, University of Al-Mustansiriyah, College of Science, Dept. of Mathematics.