Int. J. Nonlinear Anal. Appl. 13 (2022) No. 1, 1649–1658
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.5780

Solving multi-objectives function problem using branch and bound and local search methods

Manal Hashim Ibrahim[a,*], Faez Hassan Ali[a], Hanan Ali Chachan[a]

[a]*Mathematics Dept, Mustansiriyah University, College of Science/ Baghdad, Iraq*

(Communicated by Madjid Eshaghi Gordji)

Abstract

In this paper we consider $1//\sum_{j=1}^{n}(E_j + T_j + C_j + U_j + V_j)$ problem, the discussed problem is called a Multi objectives Function (MOF) problem, As objective is to find a sequence that minimizes the multiple objective functions, the sum earliness, the tardiness, the completion time, the number of late jobs and the late work. The NP-hard nature of the problem, hence the existence of a polynomial time method for finding an optimal solution is unlikely. This complexity result leads us to use an enumeration solution approach. In this paper we propose a branch and bound method to solve this problem. Also, we use fast local search methods yielding near optimal solution. We report on computation experience; the performances of exact and local search methods are tested on large class of test problems.

Keywords: Machine Scheduling with Multi-Objective problem, Branch and Bound, Simulated Annealing, Genetic Algorithm. Optimization, Firefly algorithm.

1. Introduction

In the Multi-Objective Functions problems, we are given a set of n jobs $N = \{1, 2, \ldots, n\}$ which are available for processing at time zero has to be scheduled without preemption on a single machine that can handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards and machine idle time is not allowed. Each job j, $j \in N$, requires a processing time p_j and should be completed on its due date d_j.
Machine Scheduling Problem (MSP) in recent years attracted the attention of many researchers in both academic and industrial environments and large number of papers confirms the application of these matters in logistics, production planning and supply chain management [1].

*Corresponding author
Email address: manalhashimibrahim@uomustansiriyah.edu.iq (Manal Hashim Ibrahim)

Received: April 2021 *Accepted:* September 2021

The considered problem in this research was inspired from a real-world case that many companies can frequently face. The problem under study widely used in engineering disciplines to solve problems, with multiple conflicting design objectives ([2, 3]).

In this paper, first, we solve the discussed problem using Branch and Bound (BAB) method as an exact method, secondly, we will discuss some approximate Local Search Methods (LSMs) for MSP. Many scheduling problems have proved to be of this type. Therefore, we will apply approximate methods of a large volume of test problems to one of the scheduling problems and to a void the problem of solving problems that require large arithmetic times. The approximate method can be defined as a good solution method used to find a solution close to the optimal solution and with a suitable calculation time, but the disadvantage of this method is that it does not guarantee that the solution will be optimal [4].

In recent years, this type of problem has been discussed by Mohammad (2017) [5] using BAB algorithm was used to solve the problem $1//\sum_{j=1}^{n} C_j + ET_{max}$ for $n \leq 40$ and his Tree Type Heuristic (TTH) gave excellent times for jobs. While Chachan and Hameed (2019) [6] studies the problem $1//\sum_{j=1}^{n} (C_j + T_j + E_j + V_j)$ and get the optimal solution by BAB for $n \leq 18$. Also found a near optimal solution for this type of problem by LSMs, Such as the problem $1//\sum C_j + \sum T_j + \sum E_j + T_{max} + E_{max}$ is considered to be strongly NP-hard by Abdul-Razaq and Akram (2018) [7], but they applied two local search algorithms; descent (DM) method and simulated annealing (SA) method to solve the problem for $n \leq 5000$ jobs. Abdul-Razaq and Ali (2016) [8], propose the Particle Swarm Optimization (PSO) and GA as heuristic methods to find approximation solutions for $1//Lex(\sum C_j\ ,\ \sum T_j)$ and they found that these LSM solve the problem for jobs with reasonable time. Abdul-Razaq and Motair reach jobs $n \leq 10000$ by using three local search techniques; DM, SA and tabu search (TS) in solving the problem $1//(\sum C_j + \sum T_j + T_{max} + E_{max})$, where they showed that the performance of the algorithms is evaluated on a large set of test problems and the results which are compared showed that SA and TS algorithms are better than DM with preference to SA, and showed that the three algorithms find optimal or near optimal solutions in a reasonable times [9].

The remainder of the paper is organized as follows: Firstly, we introducing the formulation of mathematical models of the proposed problem in section 2 and the derivation of lower bound (LB) in section 3. Next, we introduced and developed the heuristic methods in section 4. Also, we use the Exact (Complete Enumeration Method (CEM) and Branch and Bound (BAB)) method to get the optimal solution for the study problem in section 5. In this work we introduced LSMs to solve the study problem in section 6. The computational experiments of the proposed (models) is given in section 7. Section 8 contained the remarking conclusion and future extensions.

2. Problem Formulation

In this section, we consider MSP with n jobs on a single machine which is always available can do them, where each one of these jobs can be executed on that machine at its special time (i.e. only one job can be executed at a time), and the machine can do only one job at a time to minimize the total earliness time, total tardiness time, total completion time, total number of tardy jobs, and total late work (i.e., to minimize the multi-objective) .This problem is defined by $1//\sum_{j=1}^{n} (E_j + T_j + C_j + U_j + V_j)$. Using the standard scheduling problem classification notation,

the main problem is denoted by (P) and can be formulated as follows:

$$\left.\begin{array}{ll}
\text{Min}Z = \text{Min}\sum_{j=1}^{n}(E_j + T_j + C_j + U_j + V_j) & \\
\text{Subject to :} & \\
C_j \geq p_j, & j = 1, \ldots, n \\
C_j = C_{j-1} + p_j \ ; & j = 2, \ldots, n \\
E_j = \max\{d_j - C_j \ , \ 0\} \ , & j = 1, \ldots, n \\
T_j = \max\{C_j - d_j, \ 0\}, & j = 1, \ldots, n \\
U_j = \begin{cases} 0 & \text{if } C_j \leq d_j \\ 1 & \text{otherwise} \end{cases} & \\
V_j = \max\{p_j, T_j\} &
\end{array}\right\} \text{(P)}$$

3. Derivation of Lower Bound

Deriving a lower bound (LB) for a problem that has a multiple objective function is very difficult since it is not easy to find a sequence that give the minimum for two objectives. Since our problem is NP-hard we may find a sequence that gives minimum value for one of them but not both. The problem (P) can be decomposed into three subproblems (P_1), (P_2) and (P_3).

$$\left.\begin{array}{ll}
V_1 = \min_{\sigma \epsilon S}\{Z_1(\sigma)\} = \min_{\sigma \in S}\sum_{j=1}^{n}(E_j + T_j + C_j) & \\
\text{subject to :} & \\
C_j \geq p_{\sigma j} & j = 1, \ldots, n \\
C_j = C_{\sigma j-1} + p_{\sigma j} & j = 2, \ldots, n \\
E_j \geq d_{\sigma j} - C_j & j = 1, \ldots, n \\
E_j \geq 0 & j = 1, \ldots, n \\
T_j \geq C_j - d_{\sigma j} & j = 1, \ldots, n \\
T_j \geq 0 & j = 1, \ldots, n
\end{array}\right\} (P_1)$$

Also the subproblem:

$$\left.\begin{array}{ll}
V_2 = \min_{\sigma \epsilon S}\{Z_2(\sigma)\} \ = \min_{\sigma \epsilon S}\sum_{j=1}^{n} U_{\sigma j} & \\
\text{subject to :} & \\
C_j \geq p_{\sigma j} & j = 1, \ldots, n \\
C_j = C_{j-1} + p_{\sigma j} & j = 2, \ldots, n \\
U_j \ = \begin{cases} 0 \ , & \text{if } C_j \leq d_{\sigma j} \\ 1, & \text{if } C_j > d_{\sigma j} \end{cases} & j = 1, \ldots, n
\end{array}\right\} (P_2)$$

And the subproblem:

$$\left.\begin{array}{ll}
V_3 = \min_{\sigma \epsilon S}\{Z_3(\sigma)\} \ = \min_{\sigma \epsilon S}\sum_{j=1}^{n} V_{\sigma j} & \\
\text{Subject to :} & \\
C_j \geq p_{\sigma j}, & j = 1, \ldots, n \\
C_j = C_{j-1} + p_{\sigma j}, & j = 2, \ldots, n \\
V_j = \min\{T_j, p_{\sigma j}\} \ , & j = 1, \ldots, n
\end{array}\right\} (P_3)$$

This decomposition has simpler structure than (P), and thus appear easily first to solve optimality for (P_1) to get Z_1 by Mohammed[5] which is obtain as follows:

$$LB_1 = \sum_{j=1}^{n} (E_j + T_j + C_j) = \max(\sum_{j=1}^{n} d_j, \sum_{j=1}^{n} \max(2C_j - d_j, C_j))$$

Second, to get the minimum value (Z_2) for (P_2), using Moor's Algorithm [10] below:

Algorithm (1): Moor's Algorithm (MA)

Step (1): Order the jobs by (EDD) rule i.e., jobs are sequenced in non-decreasing order of their due dates, set $E = L = \emptyset$ and$k = t = 0$.
Step (2): Set $k = k + 1$, $if\ k > n$ go to step (4).
Step (3): Set $t = t + P_k$, $E = E \cup \{K\}$, If $t \leq d_k$ go to step(2), otherwise (i.e., If $t > d_k$) then find a job $j \in E$ such that P_j is as large as possible and let $t = tp_j, L = L \cup \{j\}$, $E = E - \{j\}$, and go to step (2).
Step (4): E is the set of early jobs and L is the set of late jobs.
Thus get $LB_2 = \sum_{j=1}^{n} U_{\sigma j}$

Third, to obtain optimal solution for (P_3) to get Z_3, first we have to introduce the following theorem:

Theorem 3.1 (Lawler 1973). *[11] The $1//f_{\max}$ problem is minimized as follows: While there are unassigned jobs, assign the job that has minimum cost when scheduled in the last unassigned position in that position.*

So Lawler's Algorithm (LA) solved the $1//f_{\max}$ problem where $f_{\max} \in \{L_{max}, T_{\max}, V_{max}\}$[12] to find minimum $f_{\max}$. LA is described by the following steps:

Algorithm (2): Lawler's Algorithm (LA)

Step (1): Let $N = \{1, 2, \ldots, n\}$, F is the set of all jobs with no successors and $\pi = \emptyset$.
Step (2): Let j^* be a job such that $f_{j^*}(\sum_{i \in N} p_i) = \min_{j \in F}\{f_j(\sum_{i \in N} p_i)\}$.
Step (3): Set $N = N - \{j^*\}$ and sequence jobj^* in last position of, i.e. π =(j*, π).
Step (4): Modify F with respect to the new set of schedulable jobs.
Step (5): If $N = \emptyset$ stop, otherwise go to step (2).
Thus get $LB_3 = \sum_{j=1}^{n} V_{\sigma j}$.

Theorem 3.2. *[11] $Z_1 + Z_2 \leq Z$ where Z_1, Z_2 and Z are the minimum objective function values for the problems (P_1), (P_2) and (P).*

Lemma 3.3. *[11] If L_1 is LB for (P_1) and L_2 is LB for (P_2), then $L_1 + L_2$ is LB for (P) problem. Hence LB=Z_1+Z_2+Z_3 is a LB for the (P) problem since:*

$$LB = \min_{\sigma \in S} \sum_{j=1}^{n} (E_{\sigma_j} + T_{\sigma_j} + C_{\sigma_j} + U_{\sigma_j} + V_{\sigma_j}) \geq Z_1 + Z_2 + Z_3 \qquad \ldots \qquad (3.1)$$

4. Dispatch Rules and Heuristic Methods

In this section we describe several dispatch heuristics that were used to generate initial sequences for the upper bounding procedures. Some of these heuristics gives optimal solutions for some problems and their main characteristics are summarized in Table 1.

Table 1: Dispatch rules used in upper bounding procedures.

Rule	Description
SPT	Smith or the Shortest Processing Time rule, that is, sequencing the jobs in non-decreasing order of their processing time. This rule solves the $1//\sum C_j$ problem [13]. More general is the SWPT rule, that is, sequencing the jobs in non-decreasing order of their processing time to weight ratio which solves the problem $1//\sum W_j C_j$.
MDD	the Modified Due Date (MDD) heuristic the due date of each job j is modified to $\max\{t+p_j, d_j\}$[14].
ATC	the Apparent Tardiness Cost (ATC) heuristic selects, whenever the machine becomes available, the unscheduled job with the highest priority index $\frac{1}{p_j}\exp\{\frac{-\max(0,\ d_j-t-p_j)}{\varphi\overline{p}}\}$, where $\overline{\text{p}}$ the average processing time, t is the current time and φ is look ahead empirical parameter [14].
EDD	The earliest due date rule, that is, sequencing the jobs in non-decreasing order of their due date, which solves the $1//\text{T}_{max}$ problem [15].
MST	The minimum slack time rule, that is, sequencing the jobs in non-decreasing order of their slack time $d_j - p_j$, which solves the $1//\text{E}_{max}$ problem [16].
MA	Described in section (3)
LA	Described in section (3)

5. Exact Solution Methods

There are many methods are developed to give an exact solution. The focus in our paper will be on the complete enumeration, and branch and bound methods

5.1. Complete Enumeration Method

Complete enumeration method (CEM) generates all the feasible solutions and then pick the best one. For example, for a single machine problem with n jobs there are $n!$ different alternatives. Hence for the corresponding m machines problem, there are $(n!)^m$ different sequences. This method may take considerable time as the number $(n!)^m$ is very large even for relatively small values of n and m.

5.2. Using Branch and Bound Method for Solving P-problem

Branch and bound (BAB) method represents implicit enumeration technique aim to find an optimal solution by testing the subsets of the feasible solutions systematically. BAB is usually described as a search tree with nodes corresponding to its subsets. This technique is applied and used in many optimization problems (see [17, 18]).

In this work, we propose the BAB algorithm which described below:

Algorithm (3): Branch and Bound (BAB) Method

Step(1): INPUT: n, p_j and d_j , $j = 1, 2, \ldots, n$ for P-problem.

Step (2): Let $Z(\sigma_i) = \sum_{j=1}^{n} (E_{\sigma ij} + T_{\sigma ij} + C_{\sigma ij} + U_{\sigma ij} + V_{\sigma ij})$, let σ_i be the sequence of jobs ordered according to the rules $R(i)$, where: R = {SPT,MDD,ATC,EDD,MST,MA,LA} and $UB_i = Z(\sigma_i)$, $i = 1, 2, \ldots, 7$.

Step (3): Set the upper bound $UB = \min\{UB_1, UB_2, UB_3, UB_4, UB_5, UB_6, UB_7\}$ at first level of BAB.

Step (4): For each node IN, compute the lower bound $LB(IN)$ =cost of sequencing jobs+cost of unsequencing jobs, where the cost of unsequencing jobs is obtained by the procedure described in section (3).
Step (5): Branch each node IN with $LB(IN)<UB$.
Step (6): At last level of BAB applying Backtracking to improve the UB.
Step (7): If $LB \leq$ the best UB, then LB is the optimal solution.
Step (8): Stop.

6. Local Search Algorithms: The Basic Notation

Local Search Methods (LSMs) share the following feature:

- ***Initialization***: The initial solution is the point from which the local search procedure is started, this could be a solution obtained from a heuristic or generated randomly, since a random solution may not satisfy the minimum of objective function [19].
- ***Neighborhood generation:*** A "move" is made through the solution space **S** from one neighbor to another to select a neighbor **s'** of **s.**
- ***Acceptance test:*** Each LSM has its own acceptance test to decide whether **s'** replace **s** as the current solution.
- ***Stopping criteria:*** The stopping criterion is the method used to terminate the search process.

LSMs are widely to obtain approximate solutions to compatibility problems. Neighborhood research is a kind of LSMs from which one can study the method of changing adjacent pairs we will also study the approximate method of the search tree. Clearly, if the algorithm selects always the best or at least a better-cost neighbor, the algorithm will end up in a local minimum [4]. In this paper, we propose to use two LSM's (SA and GA) to solve problem (P) in order to obtain near optimal solutions for the large sized instances in a small consuming time.

6.1. Simulated Annealing

Simulated annealing (SA) is an algorithmic method that is able to escape from local minima. It is a randomized LSM for two reasons: First, from the neighborhood of a solution a neighbor is randomly selected. Secondly, in addition to better-cost neighbors, which are always accepted if they are selected, worse-cost neighbors are also accepted, although with a probability that is gradually decreased in the course of the algorithm's execution. For more information see. The main steps of SA are as follows [20]:

Algorithm (4): Simulated Annealing (SA)

Step(1): Select an initial solution $s \in S$, $s^* = s$; select an initial temperature $t_o > 0$; $k = 0$,$G = 1$;
Step (2): Define B; choose $s' \in N^*(s)$; $\Delta = f(s') - f(s)$; $p(\Delta, t_k) = \exp(-\Delta/t_k)$; If $\Delta \leq 0$, then $s = s'$, and if $f(s) < f(s^*)$, then $s^* = s$; else $(\Delta > 0)$; If a random number of $[0, 1] \leq p(\Delta, t_k)$, then $s = s'$; $G = G + 1$,
Step (3): If $G \leq B$ return to step (2),
Step (4): Update temperature; $k = k + 1$; return to step (2) until some Stopping criteria are met.

6.2. Genetic Algorithms

Genetic Algorithms (GA) are invented by Holland (1975). The algorithms have been used for a wild variety of problems including machine learning, game Playing, and combinatorial optimization. GAs use a population of possible solutions to conduct a robust search of search space. The main steps of GA are as follows [21]:

Algorithm (5): Genetic Algorithms (GA)

Step (1): Create an initial population of m parents.
Step (2): Compute and save the fitness value $f(i)$ for each individual (i).
Step (3): Define selection probabilities $p(i)$ for each parent i so that $p(i)$ is proportional to $f(i)$.
Step (4): Generate m offspring by probabilistically selecting parents to produce offspring.
Step (5): Select only the offspring to survive.
Step (6): Repeat step (2) until a stopping criterion has been met.

7. Computational Results and Comparison Results for Solving P-problem

7.1. Computational Experiments with Exact Solution

The exact solution (CEM and BAB) were test on P-problem with $n = 6, \ldots, 17$ jobs. Number of jobs refers to the problem size. Job i become available for processing at a time zero, requires integers processing times p_i , $i = 1, \ldots, n$ were generated from the uniform distribution [1,100], and requires due dates d_i , $i = 1, \ldots, n$, were generated from the uniform distribution $[\left(1 - T - \frac{RDD}{2}\right) TP, \left(1 + T + \frac{RDD}{2}\right) TP]$ as it has been showed in the literature [22], where $T = \sum_{j=1}^{n} p_j$ depending on the relative range of due date (RDD) and on the average tardiness (T). For both parameters, the values 0.2, 0.4, 0.6, 0.8 and 1 are considered. These yields (10) test problems, for each value of (n).

7.2. Test Problems for the Suggested LSM's

To introduced LSM (SA and GA) we generated in section 7.1 were tested on the problem (P) with five functions for generating best solution. These algorithms were tested on problem (P) with (50, 100, 250, 500, 750, 1000, 2500, 5000, 10000).

In our computational, we use the condition that: if the solution of an example with "n" jobs for any algorithm is not appear after (600) seconds.

Shows for each algorithm, how many it can catch the optimal value for each value of n (problem size). where $n \in \{6, 7, \ldots, 17\}$. The optimal solution for examples of small size $n \in \{6, 7, \ldots, 17\}$, was found by using BAB algorithm, and for problems of large size $n \in \{50, 100, 250, 500, 1000, 2000, 10000\}$ to get near optimal solution using LSMs.

7.3. Computational results

In our computational results are given in tables, Table 2, shows the comparison between CEM and BAB with the discussed LSM (SA and GA). . These results that the value average for $n = 6 : 10$ Tables3 shows the comparison between BAB and SA and GA for $n = 11 : 17$. Table 4, shows the comparison between BAB and the discussed LSM for the study problem for $n = 50, 100, \ldots, 10000$.

Table 2: Comparison between CEM and BAB with SA and GA for $n = 6 : 10$.

n	CEM		BAB		SA		GA	
	AV of Z	AT	AV of Z	AT	AV of Z	AT	AV of Z	AT
6	194.2	R	194.2	R	194.2	R	194.2	R
7	249.9	R	249.9	R	249.9	R	249.9	R
8	271.4	R	271.4	R	271.4	R	271.4	R
9	274.6	R	274.6	R	274.6	1.0	274.6	R
10	428.5	60.2	428.5	R	428.5	R	428.5	R
AV	**283.7**	**12.3**	**283.7**	**R**	**283.7**	**R**	**283.7**	**R**

Table 3: shows the comparison between BAB and SA and GA for n=11:17.

n	BAB		SA		GA	
	AV of Z	AT	AV of Z	AT	AV of Z	AT
11	518.5	1.5	520.1	1.6	518.5	R
12	577.3	3.6	582.7	1.1	577.3	R
13	640.8	10.8	650.2	1.2	642.8	R
14	782.7	48.9	796.4	1.1	693.6	R
15	846.5	118.5	870.3	1.1	847.1	R
16	871.0	820.1	892.6	1.1	873.2	R
17	1.089.1	1220.9	1118	1.2	1091.0	R
AV	**606.8**	**466.9**	**775.8**	**1.2**	**749.1**	**R**

Table 4: Comparison results between SA and GA for $n = 50, 100, \ldots, 10000$.

n	SA		GA	
	AV of Z	AT	AV of Z	AT
50	8924.3	2.5	8549.1	R
100	34257.5	4.1	33057.4	R
250	211651.3	8.7	206944.5	3.7
500	825408.2	13.8	804246.6	11.2
750	1759259.6	18.0	1722926	21.4
1000	3318071.8	22.9	3242985.7	37.5
2500	20348052	59.7	20027327	237.5
AV.	**3786517.8**	**18.5**	**3720862.3**	**44.5**
5000	79082191.9	133.4	-	-
10000	152935237	275.1	-	-

Note: The symbols which used in the tables are:
n: no. of jobs
AV of Z: The optimal value of the function using[(BAB), (CEM)].
AV of Z: The best value of the function using LSMs [(SA, GA)].
AV of Z (CEM): The optimal value of the function using (CEM)
AT: The average of the execution time of the problem (by second).
R: $0 \leq R \leq 1$
sign (-) refers to the unsolved examples.

8. Conclusion and Future Extensions

In this work we propose to study the single MSP in order to minimize the objective function of problem (P). We proposed and discussed the exact and some LSM to solve the NP-hard problem. The proposed algorithms more tested and compared with the problem (P) for small instance sizes greater than 17. The computational results show that the SA and GA are very efficient both in term of quality of the objective function values and computational time.

In Future Extensions, it will be extension of the problem (P) by driving a good LB and generate more instances or using the Dominances rule in BAB method. Also, using the LSM should be explored for finding an improvement potential of various polynomially bounded scheduling algorithms (Heuristic).

References

[1] M. Rajabzadeh, H. Vahdani, Z. Arabasadi, *Single machine scheduling with different types of transportation facilities in batch delivery system*, CIE44 & IMSS14 Proceedings, 14-16 October 2014, Istanbul / Turkey, Pages: 1441-1450.

[2] C. A. C. Coello, G. B. Lamont, *Applications of multi-objective evolutionary algorithms*, Vol.1, World Scientific, 2004.

[3] S. Garcia, C. T. Trinh, *Comparison of multi-objective evolutionary algorithms to solve the modular cell design problem for novel biocatalysis*, Processes, 7(6) (2019) 361.

[4] K. Steinhöfel, A. Albrecht, C. K. Wong, *An experimental analysis of local minima to improve neighbourhood search*, Computers & Operations Research, 30(14)(2003) 2157-2173.

[5] H. A. Mohammed , *Exact and Heuristic Algorithms for Solving Combinatorial Optimization Problems*, Ph.D. Thesis, Mustansiriyah University, College of Science, Dept. of Mathematics 2017.

[6] H. A. Chachan, A. S. Hameed, *Exact Methods for Solving Multi-Objective Problem on Single Machine Scheduling*, Iraqi Journal of Science, (2019) 1802-1813. .

[7] T. S. Abdul-Razaq, A. O. Akram, *Local Search Algorithms for Multi-Criteria Single Machine Scheduling Problem*, Ibn AL-Haitham Journal for Pure and Applied Science, (2018) 436-451.

[8] T. S. Abdul-Razaq, F. H. Ali, *Algorithms for Scheduling a Single Machine to Minimize Total Completion Time and Total Tardiness*. Basrah Journal of Science, 34(A (2)) (2016) 113-132.

[9] T.S. Abdul-Razaq, H. M. Motair, *Solving Composite Multi-objective Single Machine Scheduling Problem Using Branch and Bound and Local Search Algorithms*. Al-Mustansiriyah Journal of Science, 28(3) (2018) 200-208.

[10] L. P. Michael, *Scheduling: theory, algorithms, and systems*, Springer 2018 .

[11] E. L. Lawler, *Optimal sequencing of a single machine subject to precedence constraints. Management science*, 19(5)(1973) 544-546.

[12] A. A. Mahmood, *Solution procedures for scheduling job families with setups and due dates*, Doctoral dissertation, M. Sc. Thesis, University of AL-Mustansiriyah, College of Science, Dept. of Mathematics 2001.

[13] W. E. Smith, *Various optimizer for single-stage production*, Naval Research Logistics Quarterly, 3(1-2) (1956) 59-66.

[14] I. M. Alharkan, *Algorithms for sequencing and scheduling*, Industrial Engineering Department, King Saud University, Riyadh, Saudi Arabia 2005.

[15] J. R. Jackson, *Scheduling a production line to minimize maximum tardiness*, management science research project 1955.

[16] J. A. Hoogeveen, S. L. van deVelde, *Polynomial-time algorithms for single-machine multicriteria scheduling*, Department of Operations Research, Statistics, and System Theory [BS], (R 9008) (1990).

[17] C. P. Tomazella, M. S. Nagano, *A comprehensive review of Branch-and-Bound algorithms: Guidelines and directions for further research on the flowshop scheduling problem.* Expert Systems with Applications, 158 (2020) 113556.

[18] H. He, H. Daume III, J. M. Eisner, Learning to search in branch and bound algorithms. *Advances in neural information processing systems*, 27(2014) 3293-3301.

[19] J. N. Gupta, K. Hennig, F. Werner, *Local search heuristics for two-stage flow shop problems with secondary criterion.* Computers & Operations Research, 29(2) (2002) 123-149.

[20] X. J. Wang, C. Y. Zhang, L. Gao, P. G. Li, *A survey and future trend of study on multi-objective scheduling*, In 2008 Fourth International Conference on Natural Computation (Vol. 6, pp. 382-391)(2008, October), IEEE.

[21] S. M. Jasim, F. H. Ali, *Exact and local search methods for solving travelling salesman problem with practical application*, Iraqi Journal of Science, 60(5)(2019) 1138-1153. https://doi.org/10.24996/ijs.2019.60.5.22

[22] S. Stöppler, C. Bierwirth, *The application of a parallel genetic algorithm to the n/m/P/C max flowshop problem*, In New Directions for Operations Research in Manufacturing (pp. 161-175) (1992), Springer, Berlin, Heidelberg.