Using Genetic Algorithm and Local Search to Solve Flow shop NP - complete

[1]Tariq S. Abdul-Razaq and [2]Lika Z. Hummady
[1]Mathematics Department / Al-Mustansiriyah University
[2]Geology Department/ Baghdad University

الخلاصة

هناك عدة مسائل في الجدولة تمتلك الصيغة التوافقية وهذه المسائل يكون من الصعب ايجاد الحل الامثل لها ولذلك نلجا الى الطرق التقريبية لايجاد الحل الامثل او حلول مقبولة قريبة من الحل الامثل

لقد تناولنا في هذاالبحث مسالة جدولة النتاجات على ماكنتين لتصغير دالة الهدف المركبة وهي (اكبر وقت اتمام واكبر تاخير لا سالب) وفي بحثنا هذا قمنا باستعراض وتطبيق بعض الطرق التقريبية وهي

Adjacent pairwise interchange method (APIM),Descent method(DA),
Simulated annealing method(SAM)كما قمنا بتطوير طريقة
Genetic Algorithm (GA)

ولقد تم تقييم كفاءة هذه الطرق على مجموعة كبيرة من مسائل اختيارية.

كما قمنا بايجاد طريقة تقريبية جديدة (New heuristic method(NHM))وعند مقارنة النتائج لهذه الطريقة مع الطرق السابقة وجدت انها الافضل من حيث الكفاءة.

ABATRUCT

There are a lot of scheduling problems that have a combinatorial manner and these problems are difficult to be solved. For these scheduling problems , local search methods are used to find the optimal solution or near optimal solutions.

In this paper we consider the scheduling problem on two machine flow Shop to find the minimum maximum completion time and maximum of tardiness to be compared with some of the local search methods namely (Descent method (DM), Adjacent parwise interchange method(APIM) and Simulated annealing method (SAM) . We developed the Genetic Algorithm (GA)) by using a large number of experimental problems, we proposed a new heuristic method (NHM) and when comparing the results of this method with the preceding methods we found that it is the best in case of qualification.

INTRODUCTION

The general flow shop problem, indicated by $Fm//C_{max}$, can be stated as follows. There are n jobs numbered 1,…….,n, each of which is to be processed on machines 1,…..,m in that order. Each job i (i=1,….,n) has a processing time P_{ik} on machine k (k=1,…..,m). Each machine can process not more than one job at a time and each job can be processed by not more than one machine at a time. The order in which jobs are processed need not be the same on all machines. The objective is to find a processing order on each machine which minimizes C_{max}, the maximum completion time of all the jobs. For $Fm//C_{max}$ problem, Conway *et al*. (1) observe that there exists an optimal schedule with the same processing order of jobs on the first pair of machines and the same order on the last pair of machines. It is also well known that for m=2 , the resulting flow shop problem *(i.e* $F2//C_{max}$ *)*, can be solved using Johnson's algorithm (2) in which job i is sequenced before job j if $\min\{P_{i1},$

$P_{j2}\} \leq \min\{P_{i2}, P_{j1}\}$.Note that using other criteria usually leads to NP-hard problems for example *F2 / / Lmax* (Lenstra et al.(3)). That leads to use local search methods, which ten toward but do not guarantee the finding of optimal solution for any instance of an optimization problem.

Recently, there has been considerable excitement about the success of a new family of algorithms in attacking notoriously difficult (NP-hard) computational problems such as traveling salesman and resource-constrained scheduling,(4). These are so-called "genetic algorithms" (GA,s) use concepts drawn from the theory of evolution to "breed" progressively better solutions to problems with very large solution spaces.

Problem Formulation

To state our scheduling problem more precisely, we are given n jobs which are numbered 1, ……, n. All jobs are available for processing at time zero and are to be processed on machines **A** and **B** in that order during uninterrupted processing times $\mathbf{a_i}$ and $\mathbf{b_i}$ respectively. The objective is to find a processing order of the jobs that minimizes the composition objective function (maximum of completion time and maximum of tardiness on the second machine), this objective function is denoted by $(C_{max} + T_{max})$. Using the three field classification suggested by Graham et al.(5), this problem denoted as $F2 / / C_{max} + T_{max}$. The problem under investigation is known to be NP-complete,

since $F2 / / T_{max}$ is NP-complete (Lenstra *et al*.(3)). Given any sequence $\pi = (\pi(1), \ldots, \pi(n))$, the minimum completion times $C^A_{\pi(1)}$ and $C_{\pi(1)}$ of the first job $\pi(1)$ in the sequence on the first and second machines are equal to $a_{\pi(1)}$ and $a_{\pi(1)} + b_{\pi(1)}$ respectively . The minimum completion times of any other job $\pi(i)$, (i=2,…,n) on the first and second machines are given by

$C^A_{\pi(i)} = C^A_{\pi(i-1)} + a_{\pi(i)}$ *and* $C_{\pi(i)} = max\{C^A_{\pi(i)}, C_{\pi(i-1)}\} + b_{\pi(i)}$

respectively , hence the tardiness for each job $\pi(i)$, i=1,…n on the second machine is given by $T_{\pi(i)} = max\{ C_{\pi(i)} - d_{\pi(i)}, 0 \}$ where the $d_{\pi(i)}$ is the due date of job $\pi(i)$.The objective is to find a schedule σ ($\sigma(1),\ldots,\sigma(n)$) of the jobs that minimize the total cost $Z(\sigma)$. Our problem (P) can formally be stated as

$$\begin{aligned} &\min_{S \in \sigma} \{Z(\sigma)\} = \min \{ C_{max} + T_{max} \} \\ &\text{s.t.} \quad C_{\sigma(1)} = C^A_{\sigma(1)} + b_{\sigma(1)} \\ &C_{\sigma(i)} = \max \{ C^A_{\sigma(i)}, C_{\sigma(i-1)} \} + b_{\sigma(i)} \qquad i=2,\ldots,n \\ &C_{max} = C_{\sigma(n)} \end{aligned} \qquad \textbf{(P)}$$

$T_{\sigma(i)} \geq C_{\sigma(i)} - d_{\sigma(i)}$ i=1,….n

$T_{max} = Max \{T_{\sigma(i)}\}$

Where σ(i) denotes the position of job i in the ordering σ and S denotes the set of all sequences.

The following special cases yields optimal solutions for our problem (P).

Case (1): The sequence which is obtained by Johnson's rule is optimal for $F2 / / C_{max}+T_{max}$ problem if T_i=o for each job i(i=1,…,n).

Proof : Since all jobs are early or completed on time this mean that there are no tardiness (i.e. T_i =o for each i) and hence T_{max}=0, and our problem $F2 / / C_{max} + T_{max}$ is reduced to $F2 / / C_{max}$ which can be solved by Johnson's rule (J.R)(2).

Case (2): The sequence which is obtained by Johnson's rule is optimal for $F2 / / C_{max} + T_{max}$ problem If di =d for each i (i=1,….n).

Proof: Let σ = (1,…,n) be the sequence obtained by Johnson's rule (J.R) and the minimum C_{max} is given by : $C_{max} = \max \{C_{n-1}, C_n^A\} + b_n$

where C_{n-1} is the completion time of job n-1 on machine B, C_n^A is the completion time of job n on machine A, b_n is the processing time of the last job n on machine B. For each job $i \in \sigma$, $T_i = \max\{C_i-d,0\}$ and hence the maximum tardiness $T_{max} = C_{max} - d$, which is less than or equal to the maximum tardiness given by any other sequence $\pi \neq \sigma$. Hence σ is optimal for $F2 / / C_{max} + T_{max}$.

Drived the Upper and Lower bounds for the problem(P)

Consider the two machine flow-shop problem to minimize the composition objective function the maximum of completion time and maximum of tardiness $(C_{max} + T_{max})$. This problem is clearly NP-hard since the simpler version $F2 / / T_{max}$ is already NP-hard (6). It is well known that computation can be reduced by using a heuristic to find a good solution to act as an upper bound(UB). Also a simple techniques is used to obtain a lower bound (LB) for our problem (P).

1. Upper bound (UB)

We can find upper bound (UB) for our problem (p) by using Johnson's rule (J.R), since Johnson's rule gives the optimal solution to the important part C_{max} of this problem.

2. Lower bound (LB)

Decomposition of the problem and derivation of lower bound (LB). Now consider again the formulation of the problem(P), we decompose the problem into two subproblems with a simpler structure. Then the lower bound (LB) of

the problem (P) is the sum of the minimum value of the subproblem (P_1) and the lower bound of the subproblem (P_2).

$Z_1 = \min \{C_{max}\}$

s.t. $\sigma \in S$

$$\mathbf{C_{\sigma(1)} = C^{A}_{\sigma(1)} + b_{\sigma(1)}}$$

$$\mathbf{C_{\sigma(i)} = Max\ \{ C^{A}_{\sigma(i)}, C_{\sigma(i-1)} \} + b_{\sigma(i)} \quad i=2,\ldots,n}$$

$$\mathbf{C_{max} = C_{\sigma(n)}}$$

(P_1)

And

$Z_2 = \min T_{max}$

s.t.

$$\mathbf{T_{\sigma(i)} \geq C_{\sigma(i)} - d_{\sigma(i)} \quad i=1,\ldots.n}$$

$$\mathbf{T_{max} = Max\ \{T_{\sigma(i)}\}}$$

$$\mathbf{T_{\sigma(i)} \geq 0}$$

(P_2)

It is clear from the decomposition that (P_1) and (P_2) have simpler structures than (P) , and thus appear easily to solve optimality for (P_1) (i.e. (P_1) is solved by Johnson's rule) , and a lower bound can be obtained for (P_2) by using the relaxation theqniquies . The lower bound for (P_2) is obtained as follows. Let b_i =0 for each job i and the resulting problem is $1 / / T_{max}$ which is solved by EDD rule (the job i is sequenced before job j if di ≤ dj (i, j=1,…n) , let LBT= min $\{T_{max}\}$ for (P_2) . Hence LB= Z_1 + LBT.

An optimal solution for each test problem (with n <30) is obtained by branch and bounds (BAB) method were applied to scheduling problems by Ignall and Scharge (7) and Lomnicki(8). The branch and bound finds s* by implicit enumeration all s Є S through examination of smaller subsets of the set of feasible solutions S. These subsets can be treated as sets of solutions of corresponding subproblems of the original problem.

This method is determined by the following procedures:

I-The branching procedure*:* This describes the method to partition a subset of possible solution. These subsets can be treated as a set of solutions of corresponding subproblems of the original problem.

II-The bounding procedure*:* This indicates how to calculate a lower bound(LB) on the optimal solution value for each subproblem generated in the branching process.

III-The search strategy*:* This strategy describes the method of choosing a node of the search tree to branch from it, we usually branch from a node with the smallest lower bound (LB) among the recently created nodes.

The New Heuristic Method(NHM)

It is well known that the complete enumeration methods generate one by one all the possible feasible solutions searching for an optimal solution and eliminate the non -optimal solutions. Suppose we are given n jobs to be process on m machines, in this case, there exists $(n!)^m$ possible ordering of the jobs. Clearly, using this method searching for an optimal schedule is not suitable even for small n and m. Thus, we introduce a new heuristic for the two machine flow shop problem where the objective is to minimize the maximum completion time and maximum tardiness of jobs. An improved heuristic method for complete enumeration methods has been constructed. The method appears to be difficult to give through extensive empirical runs failed to generate even a single counter example which has not optimal or near optimal solution. This leads to an interesting observation, that the developed heuristic method, will yield optimal solutions in most instances of the extensive empirical runs performed by us. We expect that this new heuristic would produce optimal solutions for most of other instances too. In the following we shall give the outline of this heuristic:

Step 1: Start with counter =0

Step 2: Find the initial solution s.

Step3: Calculate the values of UB and LB (such that the initial solutions for UB).

Step4: Find the neighborhoods (Using subalgorithm) and compute costs for the neighborhoods.

Step 5: If all costs > UB then go to step (4).

Step 6: Register costs in the result file .

Step7: If costs $\leq$ (UB+LB)/2 then register costs in Quick file and let counter = counter +1 . Otherwise (i.e. Cost >(UB+LB)/2, go to step (10).

Step 8: If counter =5 then go to step (11).

Step9: Go to step (4).

Step10: Choose minimum of result file and go to step (12).

Step11: Chose minimum of Quick file.

Step12: End.

Where in this heuristic,

Cost : The value of the objective function($C_{max} + T_{max}$).

Result: A file for results with cost less than UB.

Quik : A file for results with cost less than (UB+LB)/2.

Counter: A counter for good solutions in Quick file.

Neighborhoods Subalgorithm

Setp1 : for i= 1 to 25

Step2 : Exchange single job.

Step3: Exchange each job with the last one.

Step4: Exchange adjacent jobs.

Step 5:Exchange the first with the last one .

Step6: Exchange each job with the first one.

S**tep7**: Next.

The Genetic Algorithm (GA):

Genetic algorithm (GA) was firstly introduced by Holand , as a highly robust strategy. It may be viewed as population based algorithm, that constructs solutions by combining others. It is based on genetic process of biological organisms. In contrast to the single current solution in neighborhood search, a genetic algorithm works with a population of solutions, where each solution may not be selected and others may appear several times. Solutions are chosen from the mating pool according to fitness values, where better quality solutions are assigned a higher fitness. The following steps describe structure of GA:

Step 1*: Initialization*

Generate random initial population of m individuals (solutions) and evaluate fitness values for each of them. In this paper, we start with different values of m. The initial population can be generated at random or can be constructed by using heuristic methods.

Step 2*: New Population:*

Create a new population by repeating the following steps untill the new population is completed:

a) Selection

The selection of parents to create the next generation is based on the fitness of individual. The probability of selection of any individuals is proportional to its fitness, thus, the better fitness individuals are more likely to be selected for reproduction. The reproduction operator uses the fitness values for the selection of solutions. Solutions with higher fitness values tend to have more of their copies at the next generation. When the objective is to minimize a function, objective function value can not be directly used for the fitness values. One way to overcome this problem is to transform the objective function value (Z_i) to fitness value (f_i). A classic method for transformation is subtracting the objective function value from an arbitrary large constant,(determine this large constant as the largest objective function value in the current population).

The selection operator may be implemented in an algorithmic form in a number of ways. In this study, we have used roulette wheel selection which

assigns a probability to each individual (solution), it is computed as the proportion:

$$Fi = \frac{fi}{\sum_{i=1}^{N} fi} \quad ,i=1,2,\ldots,n \quad \text{---------- ---------- (1)}$$

Once the fitness value (f_i) are calculated, the survival of the fittest principle can be implemented by roulette wheel selection, each solution is given a space on this wheel proportional to its fitness. The probability to select a solution s is given by (1).

Clearly, fitter sequences (solutions) stand a better chance of passing through it, but it is important to note that even sequences with a very low initial fitness have some chance of getting through. This ensures that each has some chance of developing into a better sequence by subsequent breeding. Suppose a sample population of sequences has fitness function

Example-1 : The problem of 4 jobs with the following data :-

value fi as shown in the following table (1) , where $\bar{F} = \sum_{i}^{n} fi \Big/ n$, $[\frac{Fi}{\bar{F}}]$ the integer number represent the number of times that the sequence appear.

i	1	2	3	4
a_i	3	7	5	3
b_i	2	4	8	6
d_i	6	20	15	22

Table -1: Description of the sequences which appear in the new population

i	Sequence s	$Z_i(s)$	$fi = Z_{max}-Z_i$	$Fi = \frac{fi}{\sum_{i=1}^{N} fi}$	$\frac{Fi}{\bar{F}}$	$]\frac{Fi}{\bar{F}}[$	Sequence appear in ne population
1	(3,4,1,2)	40	10	0.25	1	1	(3,4,1,2)
2	(3,1,2,4)	34	16	0.40	1.6	2	(3,1,2,4)
3	(2,3,4,1)	50	0	0	0	0	(3,1,2,4)
4	(4,3,1,2)	36	14	0.35	1.4	1	(4,3,1,2)
Sum		160	40	1			
Averag		40	10	0.25			

i : number

Z_i: The objective function (C_{max} +T_{max}) value

In fitness- proportionate selection, sequence(3 ,1 ,2 ,4) is given probability of 40 % of being selected for each of the four positions in the new population. Thus, we expect that sequence (3 , 1 ,2 ,4) will occupy two of the

four positions in the new population, while sequence (3,4, 1 ,2) will occupy one of the four positions in the new population.

The corresponding weighted roulette wheel for this generation reproduction is shown in fig (1).

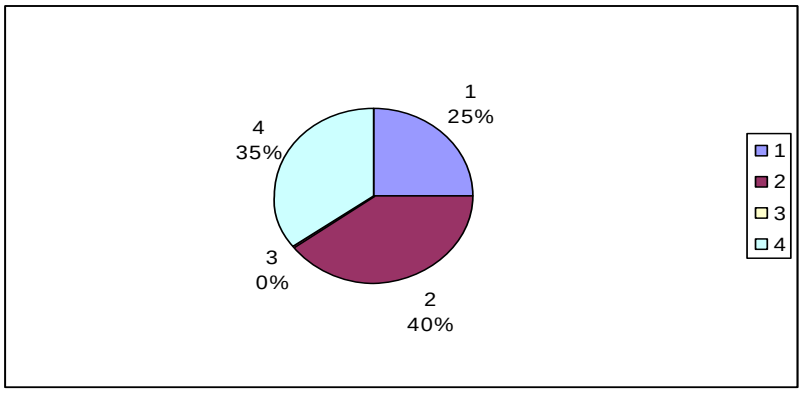


Figure -1: Sample wheel is sized for the example of table(1)

b) Crossover rate

These three different crossover rates are (XRT >0.5 , XRT = 0.5, XRT < 0.5). Results comparing the performances of the different crossover rates are given in table (2). It is clear from table (2) and Fig.(2) that XRT >0.5 is the best one. Where the * indicate that the problem has the best solution in Table (2).

Table -2: Comparative computational results of the different crossover rate and for some value of n

	Crossover Rate of GA		
n	XRT< 0.5	XRT = 0.5	XRT >0.5
10	68	66	58*
20	142	138	136*
30	227	222	221*
40	316	311	310*
50	399	393*	393*
60	491	489*	489*
70	595	595	593*
80	705	702*	703
90	793	791*	791*
100	877	873*	874

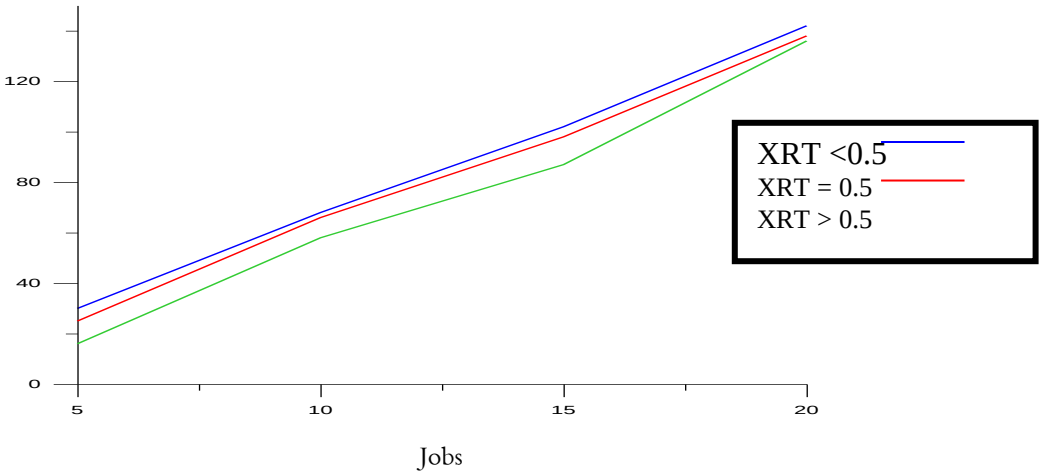
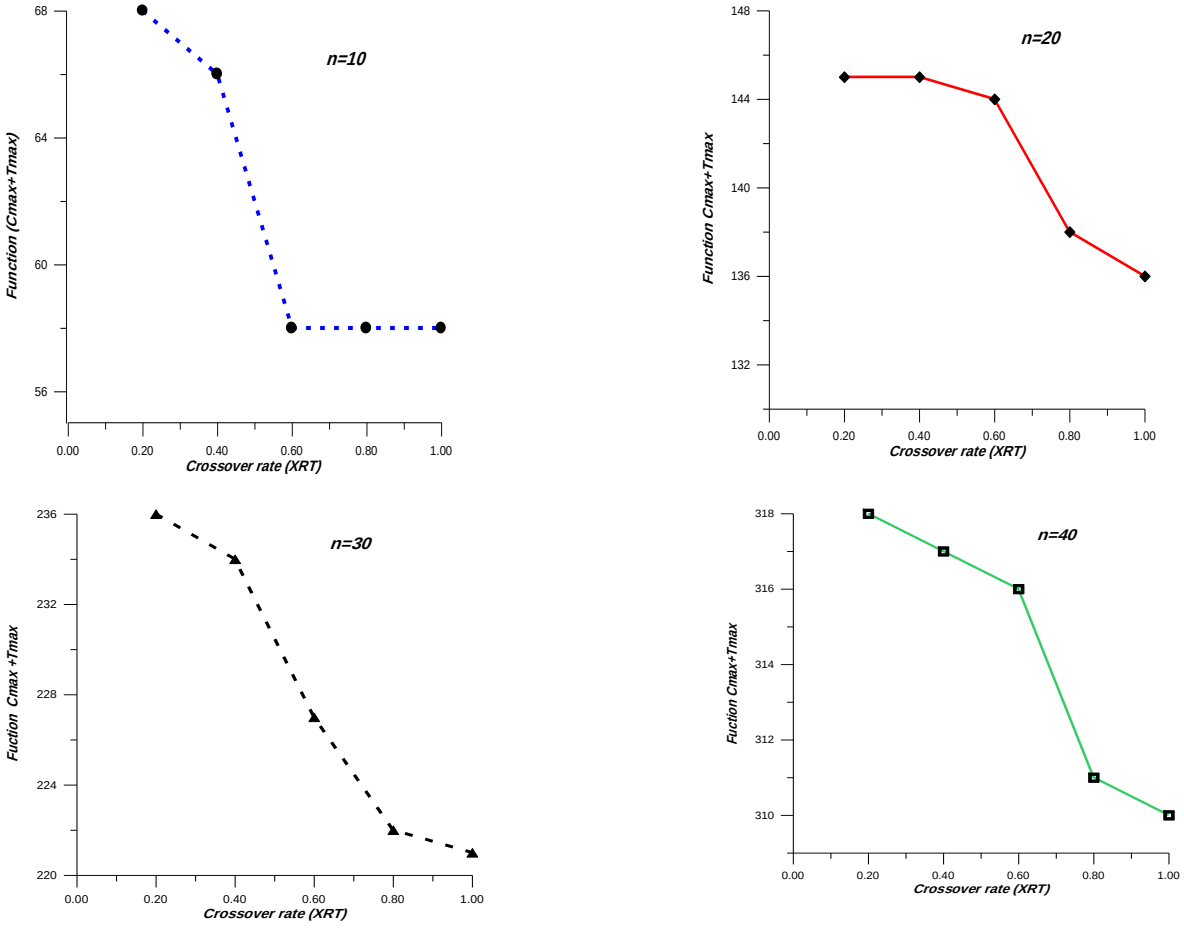


Figure -2: comparison of three different XRT,



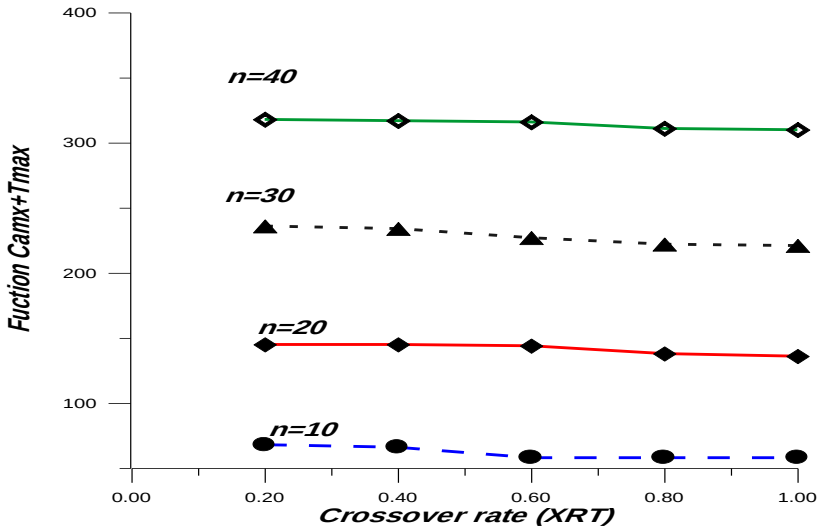


Figure -3: Description of the crossover rates for some values of n.

Figure (2) shows the difference between the crossover rates(XRT) such that crossover rate (XRT > 0.5)gives the best sequence(gen) for most of the time of the cases. Figure (3) shows that crossover rate (XRT) works for some values of n,(n=10 , n=20 ,n=30 and n=40).The results show that when a crossover rate gives the best sequence (gen) then all the rates after this rate give the same sequence.

c) PBX crossover operator

In this step we used position based crossover (PBX) to generate the new population, to find the optimal solution or near optimal solution. Table (3) shows the results for (PBX) for example (1).

Example (2): Consider again the problem of 4 jobs with data given in example (1).

Table -3: Description of the processes of GA for finding solutions

Sequence appear in new population	PBX	Objective value Z(s)	fi = Z_{max}-Zi	$Fi = \frac{fi}{\sum_{i=1}^{N} fi}$	$\frac{Fi}{\overline{F}}$	$[\frac{Fi}{\overline{F}}]$
(3,4,1,2)	(3,4,1,2)	40	0	0	0	0
(3,1,2,4)	(4,1,3,2)	28	12	0.63	2.52	3
(3,1,2,4)	(4,1,2,3)	37	3	0.16	0.63	1
(4,3,1,2)	(4,3,1,2)	36	4	0.21	0.84	1
Sum		141	19	1		
Average		35.25	4.75	0.25		
minimum		28				

d) Mutation Operator:

This operator should alter a solution (sequence) only slightly. The mutation operator is applied to perturb some of these new solutions. Hence it is clear that a small modification in the solution will result in a small deviation of its fitness value only.

Step 3 : Termination

The GA procedure stops when a fixed number of generations (or iterations) are executed here 1000 iterations. This means that the GA procedure continues until the population is converged to a good, if not optimal solution to our problem (P).

Local Search methods

Local search is a family of methods that iteratively search through the set of solutions. Starting from initial solution, a local search procedure moves from one feasible solution to a neighboring solution until some stopping criteria are met the choice of a suitable neighborhood functions has an important influence on the performance of local search.

Adjacent Pairwise Interchange Method (APIM)

This (APIM) depends on interchange elements (jobs) at positions (i) and (i+1) of a given sequence , (i=1,…..,n-1) (9). The following steps describe this method:

1-Initialization:

To obtain an initial current solution jobs are ordered according to Johnson rule (J.R),by sequencing job i with ($a_i \leq b_i$) first in non-decreasing other of a_i followed by the remaining jobs i with ($a_i > b_i$) sequenced in non- increasing order of b_i (i= 1,…,n) to obtain the current

sequence $\sigma = (\sigma(1), \ldots, \sigma(n))$, with its objective function value (UB) , where $UB = C_{max} + T_{max}$.

2-Neighbor Generation :

In order to improve the sequence σ , the position of two adjacent jobs $\sigma(i)$, $\sigma(i+1)$, $(1 \leq i \leq n-1)$ are transposed. Hence a new sequence σ' is obtained with its objective function $UB' = C_{max} + T_{max}$.

3-Evalution :

If the improvement is made (i.e $UB' < UB$) then, the two jobs are left in their new positions. On the other hand, the two jobs are replaced in their original positions. The procedure is then repeated from step (2) and other possibilities are considered in a similar way.

4-Termination Step:

The method is terminated when all possibilities are considered for adjacent jobs $\sigma(i)$, $\sigma(i+1)$, $(i=1,\ldots,n-1)$, without making any improvement.

Descent Method (DM)

This method is a simple form of local search methods. It can be executed as follows:

1- Initialization:

In this step, the feasible solution $\sigma = (\sigma(1), \ldots, \sigma(n))$ obtained from Johnson rule (J.R) , is chosen to be the initial current solution for descent method, with its objective function value (IUB).

2- Neighbor Generation*:*

In this step, a feasible neighbor $\sigma' = (\sigma'(1),\ldots, \sigma'(n))$ of the current solution is generated by choosing randomly two jobs from σ in the first time , (not necessarily adjacent) and transpose their positions, and a feasible neighbor is also generated by choosing randomly a block of jobs from σ at the second time , and transpose their positions, for each case calculate the function values and the minimum value is denoted by (CUB).

3- Acceptance Test:

Now consider the test whether to accept the move from σ to σ' or not , as follows:

(a) If CUB < IUB : then σ' replace σ as the current solution and we set IUB = CUB, and go to step (2) (Neighbor generation).

(b) Other wise (i.e . $IUB \leq CUB$): σ is retained as the current solution, and go to step 2.

4-Termination Test: Repeat step (2), and other possibilities are considered in a similar way. The DM terminates if no neighbor provides an improved objective function value, in which case the current solution IUB is a local minimum.

The Simulated Annealing Method(SAM):

It is well known that in SAM, improving and neutral moves are always accepted, while deteriorating moves are accepted by a controlled probability (10).

The following steps describe SAM.

1-Initialization:

The same technique described in DM is used to obtain an initial current solution $\sigma = (\sigma(1), \ldots, \sigma(n))$ and IUB.

2-Neighbour Generation:

A feasible neighbor of the current solution σ is generated by choosing randomly two jobs $\sigma(i)$, $\sigma(j)$ such that $(1 \leq i < n)$ $(1 \leq j \leq n)$ not necessary adjacent and transpose their positions and compute the function value for this neighbor to be (CUB).

3-Acceptance Test:

In this step, the difference value between the current initial solution IUB and the new value CUB, Δ = CUB-IUB is calculated, then evaluated as follows:

a) If ($\Delta < 0$) : then (CUB) is accepted as the new current solution, and set IUB =CUB and go to step 2

b) If ($\Delta > 0$) : (CUB) is accepted with $P(\Delta) = \exp(-\Delta/t)$, which is the probability of accepting a move, which results in an increase of Δ in the objective function value, where t is a parameter known as the temperature and is usually set to large (i.e . the probability of accepting a worse solution is high) in the beginning of the search, and gradually decreases as the search proceeds.(11)

Let (it) denotes the number of iterations that are considered. Where at each iteration i, $1 \leq i \leq$ it a temperature t_i is derived from the following recursive formula, described by Anderson *et a*l. (12).

$$t_i = t_{i-1}/(1+B*t_1) \qquad i=2,\ldots,it$$

$$\text{Where: } B=(t_1-1)/(it*t_{i-1});\ t_1=10 \text{ and } it=1000$$

4-Termination Test:

After one thousand iterations the algorithm terminates at a feasible solution (SUB).

Test problems

In selecting test problems, one important goal is to create problem instances that are representatives of the general problem class. We generated the test problems with value of n (10,20,30,40,50,60,70,80,90,100,150,200). When comparing the performance of algorithms, it is important to test them on a range of problem instances. The main characteristic of an instance of our scheduling problem is its size , as measured by numbers of jobs n. Our test problems were generated as follows: (13). The processing times $\mathbf{a_i}$ and $\mathbf{b_i}$ in the test problems were randomly sampled from a uniform distribution on the integers defined on (1,10), and due dates were generated from the uniform distribution on [(1-TF-RDD/2) sp , (1-TF+RDD/2) sp] such that

$$sp = \sum_{i=1}^{n} Ci \quad \text{where } C_i = (a_i + b_i)/2 \quad , TF= 0.2 , 0.4 \ , RDD = 0.2,0.4,0.6,0.8,1$$

Five test problem were created for some contributions of TF and RDD.

1. Initial Solutions

The sequence which is obtained by using Johnson's rule (J.R) can be used as an initial solution for adjacent pairwise interchange method(APIM) , descent method (DM) and simulation annealing method (SAM) as well as in the new heuristic method (NHM). But in genetic algorithm we generate old population randomly. Also J.R. is used to generate upper bound (UB) on the optimal solution for our problem.

If such constructive methods can produce initial solutions of sufficiently high quality, then the need for local search methods, with their heavier computational requirements, is reduced.

2. Comparison between the optimal (BAB) solution and the new heuristic method (NHM)

This section shows the efficiency of the new heuristic method given in section (4) and compares it with optimal solution obtained by BAB algorithm for each test problem.

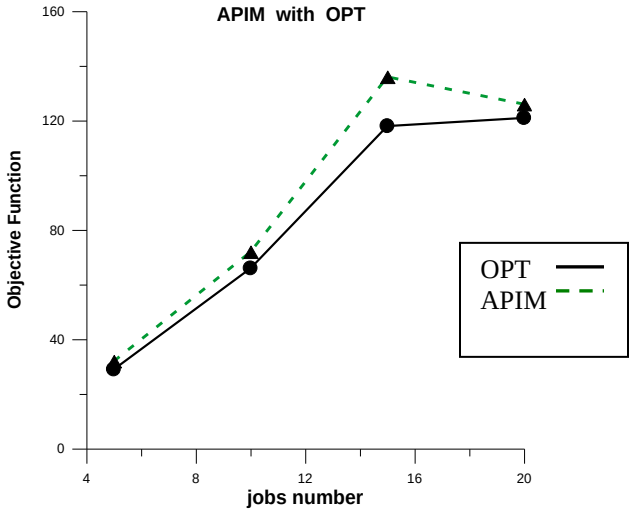
An optimal solution for each test problem (with $n \leq 30$) is obtained by BAB algorithm, these optimal solution values are used to assess the quality of solutions generated by the new heuristic methods. For our BAB algorithm, whenever a problem was not solved within a time of 6000 second, computation was bounded for that problem (11). Since the BAB algorithm can not generate optimal solutions using reasonable limits on computation time for test problems (with $n > 30$). In this case, the new heuristic method (NHM) is used to obtain the best solution value and forms the basis for comparison. It should be noted that this (NHM) for n=80 and n=100, also sometimes a large a mount

of time is needed to give good solutions. Table (4) gives the results of comparison between BAB and the new heuristic method (NHM). Also table (4) shows the results for lower and upper bounds for our problem. It is clear from table (4) that the results show that the new heuristic method NHM have the exact value of the optimal for 18 of the 20 test problems.

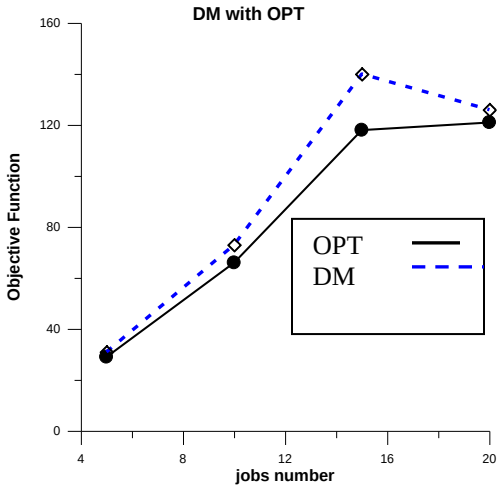
Table -4: Comparative Computational Results

n	No	UB	LB	BAB	NHM	CT of NHM
5	1	35	24	29	29	0.3
	2	31	21	27	27	0.21
	3	31	20	28	28	0.13
	4	40	26	35	35	1.2
	5	56	37	43	43	0.3
10	1	60	49	50	50	1.33
	2	81	64	69	69	1.50
	3	77	60	67	67	1.54
	4	102	66	73	73	1.65
	5	73	58	66	66	1.23
15	1	110	91	91	91	3.25
	2	142	113	118	118	5.13
	3	128	92	97	97	5.2
	4	149	114	123	123	3.25
	5	104	79	89	89	4.25
20	1	126	112	121	121	15.3
	2	159	118	124	126	12.3
	3	193	132	143	143	14.42
	4	176	128	128	128	15.33
	5	190	132	148	152	11.22

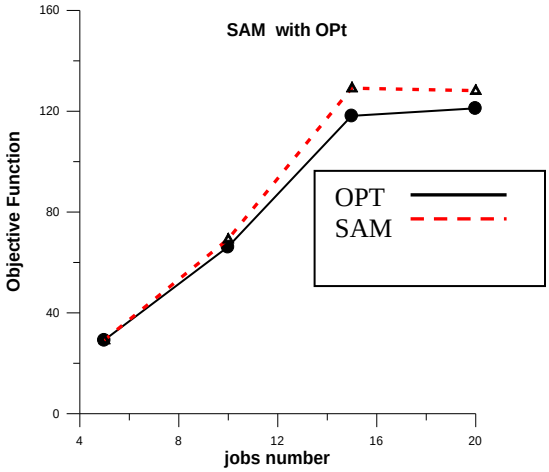
No: test problem number , n : Number of jobs
UB:Upper bound , LB: Lower bound
BAB: Branch and bound algorithm , NHM: New heuristic method
CT: computation time (in seconds)


APIM with OPT
Objective Function
jobs number
OPT
APIM

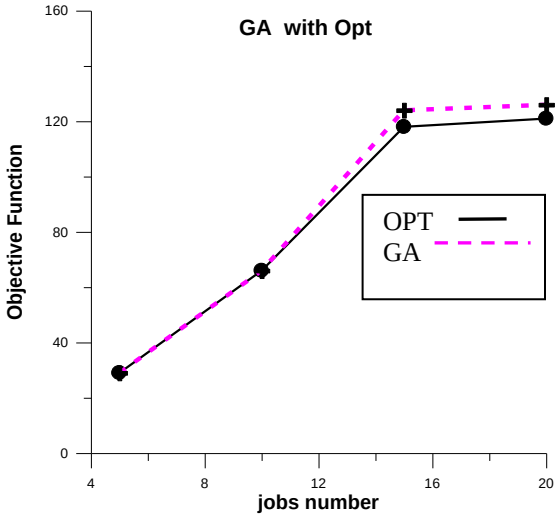
(a)


DM with OPT
Objective Function
jobs number
OPT
DM

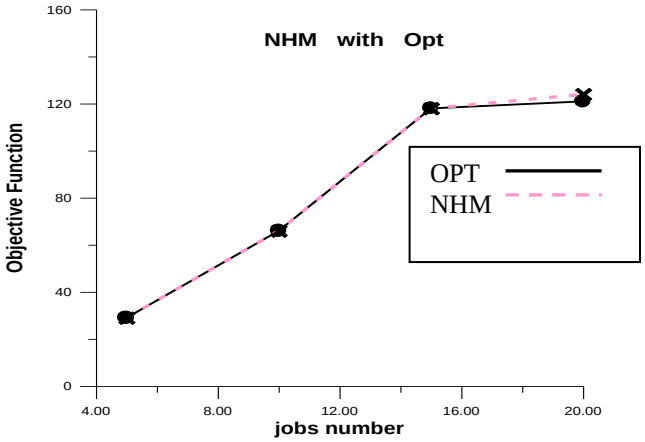
(b)


SAM with OPt
Objective Function
jobs number
OPT
SAM

(c)


GA with Opt
Objective Function
jobs number
OPT
GA

(d)


NHM with Opt
Objective Function
jobs number
OPT
NHM

(e)

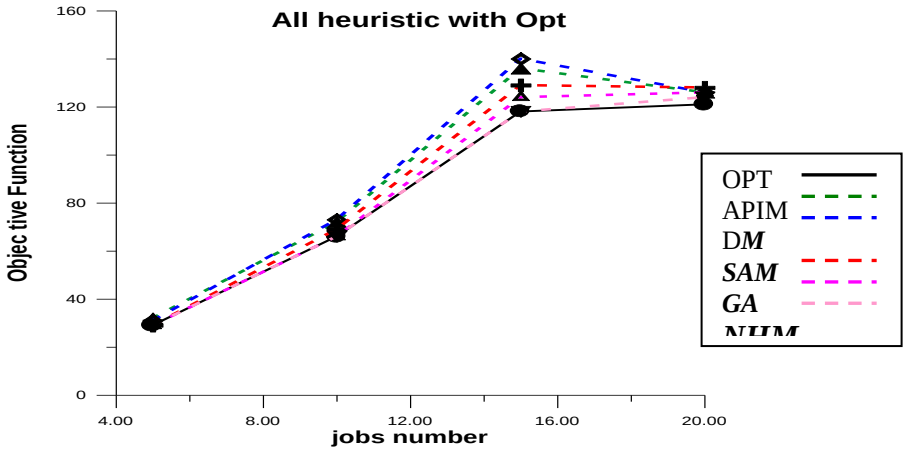


Figure -4:for (a, b, c ,d and e) shows each local search heuristic , Optimal and comparing between them

3. Comparative Computational Results

In this section, we shall report on the computational experience with different local search heuristic methods. For the methods (adjacent parwise interchange method (APIM), descent method (DM), simulated annealing method(SAM), genetic algorithm (GA) and the new heuristic method (NHM) that have been described in this paper, we present a table of results which shows the importance of each method. Table (5) shows that the (NHM) gives the best results than all the other heuristic methods but with longer time, while, the genetic algorithm GA gives the second better value after (NHM) with shorter time. Table (5) describes the comparison between the local search heuristic methods, the upper and lower bounds. Table (5) Compares the computational results.

Table -5: Comparative Computational Results

Number of jobs	Upper bound	Lower bound	API	DM	SAM	GA	NHM
10	70	52	66	68	62	58	58
20	156	129	150	140	142	136	135
30	238	211	231	226	226	221	220
40	325	289	317	316	313	310	301
50	411	379	407	396	396	393	390
60	505	471	501	494	491	489	485
70	613	570	602	598	597	593	592
80	719	633	712	706	707	703	700
90	803	758	800	794	795	791	788
100	885	849	879	877	877	874	870
125	1122	1076	1115	1119	1115	1106	---
150	1363	1311	1349	1353	1350	1346	---
175	1616	1539	1601	1613	1599	1596	---
200	1833	1778	1819	1823	1822	1817	---

REFERENCES

1. Conway , R.W. Maxwell W.L. and Miller L.W. " Theory of Scheduling Addison Wesley, Reading, MA, (1967).
2. Johnson S.M." Optimal two and three stage production schedules with set-up times included". Naral Res. Logist Quart 1:61-68(1954).
3. Lenstra J.K., Rinnooy Kan A.H.G., and Brucker P., " Complexity of Machine Scheduling Problems". Ann. Of Discrete Mathematics ,1:343-362(1977).
4. Robert A.J. Matthews "The Us of Genetic Algorithm In Cryptanalysis Crytologia . XVII (2)Apil (1993).
5. Graham R.L., Lawler E.L., Lenstra J.K., and Rinnooy kan A.H.G., " Optimization and approximation in deterministic sequencing and scheduling theory : a survey ", Discrete math. 5, :287-326(1979).
6. Karp R.M., Reducibility among combinatorial problems. In complexity of computer computations, Miller R.E and Thatcher J.W.Eds. plenum press , New York:95-103 (1972).
7. Ignall E., Schrage L.E., "Application of the branch and bound technique to some flow shop scheduling problem", Oper.Res. 13:400-412 (1965).
8. Lominickiz. A., "A branch and bound algorithm for the exact solution of three- machine scheduling problem", Oper. Res. Quart 16, :89-100(1965).
9. Chen B., A better heuristic for preempive parallel machine scheduling with batch set-up time. SIAM Jornal on computing 22:1303-1318 (1993).
10. Caruwels H.A.J., A comparative study of local search methods for one machine sequencingproblems, Ph.D. thesis, Katholike University Leuren (1998).
11. Reevs C.R., " modern heuristic techniques for combinatorial problems", John Wiley and sons. Inc.,. New York (1993).
12. Anderson E.J., Glass C.A. and Potts C.N., " local search in combinatorial optimization, edited by E.H.L. Aarts and J.K. Lenstra, wiley (1997).
13. Anderson E.J., Glass C.A. and Potts C.N., "Application of local search in machine scheduling" , March (1995).