Approximation Solutions For Multicriteria Scheduling Problems

Adawiyah A. Mahmood Al-Nuaimy

adawiyaali@yahoo.com
Diyala University
College of Science - Mathematics Department
Dr. Tariq S.Abdul-Razaq
dr.tariqsalih@yahoo.com
Al-Mustansiriyah University
College of Science- Mathematics Department

Abstract: *This paper presents local search algorithms for finding approximation solutions of the multicriteria scheduling problems within the single machine context, where the first problem is the sum of maximum tardiness and maximum late work and the second problem is the sum of total late work and maximum late work.*

Late work criterion estimates the quality of a schedule based on durations of late parts of jobs. Local search algorithms (descent method (DM), simulated annealing (SA) and genetic algorithm (GA))are implemented. Based on results of computational experiments, conclusions are

formulated on the efficiency of the local search algorithms.

Keywords: Local search, genetic algorithm, late work criterion, multicriteria scheduling.

1.Introduction

The scheduling problem is defined as a problem of assigning a set of jobs to a set of machines in time under given constraints ([1],[2],[3]). In single machine case, jobs j (j=1,2,…,n) are mainly characterized by processing times (p_j), due dates (d_j), define expected completion times ($C_j = \sum_{i=1}^{j} p_i$) for particular schedule of jobs. The quality of an assignment, i.e. a schedule, can be evaluated from different points of view, which are represented by different performance measures. Most objective functions based on due dates are regular ones, i.e. non-decreasing with increase in completion times of jobs. This group includes criteria based on lateness ($L_j = C_j - d_j$), tardiness ($T_j = \max\{0, C_j - d_j\}$) or the number of tardy jobs ($U_j = 1$, if $C_j > d_j$, otherwise $U_j = 0$). The criteria based on earliness ($E_j = \max\{0, d_j - C_j\}$) are non-regular ones.

Late work combines the features of two parameters: tardiness and the number of tardy jobs. Formally speaking, in the non-preemptive case the late work parameter for job j in a given schedule is defined as

$V_j = \min\{\max\{0, C_j - d_j\}, p_j\} = \min\{T_j, p_j\}$ or, in a more extensive way, as:

$$V_j = \begin{cases} 0 & \text{if } C_j \le d_j \\ C_j - d_j & \text{if } d_j < C_j < d_j + p_j \\ p_j & \text{if } C_j \ge d_j + p_j \end{cases} \quad , j = 1,2,\ldots,n.$$

The parameter V_j was first introduced by Blazewicz [4], who called it "information loss", referring to a possible application of the performance measures based on it. The phrase "late work" was proposed by Potts and Van Wassenhove [5]. Some researchers, e.g. Hochbaum and Shamir [6], use a descriptive name for this schedule parameter-the number of tardy job units.

The relation between late work and other performance measures was established by Blazewicz et al. [7]. Interesting applications of the late work criteria arise in agriculture, where performance measures based on due-dates are especially useful [8]. Late work criteria can be applied in any situation where a perishable commodity is involved [5]. Leung [9] pointed out another application of late work scheduling in computerized control systems, where data are collected and processed periodically.

The organization of this paper is as follows. Section 2 presents the problems formulation. Section 3 provides local search algorithms incorporating a solution representation of a scheduling problem. Section 4 summarizes results of computational experiments and it is followed by conclusions in section 5.

2.Problems formulation

A set of n independent jobs N = {1,2,…,n} are available for processing at time zero, job j (j=1,2,…,n) is to be processed without interruption on a single machine that can be handle only one job at a time, requires processing time p_j and due date d_j. For a given schedule σ of the jobs, completion time $C_{\sigma(j)} = \sum_{i=1}^{j} p_{\sigma(i)}$, maximum late work $V_{max}(\sigma) = \max\{V_{\sigma(1)}, V_{\sigma(2)}, \dots, V_{\sigma(n)}\}$ can be computed where:

$$V_{\sigma(j)} = \begin{cases} 0 & \text{if } C_{\sigma(j)} \le d_{\sigma(j)} \\ C_{\sigma(j)} - d_{\sigma(j)} & \text{if } d_{\sigma(j)} < C_{\sigma(j)} < d_{\sigma(j)} + p_{\sigma(j)} \\ p_{\sigma(j)} & \text{if } C_{\sigma(j)} \ge d_{\sigma(j)} + p_{\sigma(j)} \end{cases}$$

and maximum tardiness $T_{max}(\sigma) = \max\{T_{\sigma(1)}, T_{\sigma(2)}, \dots, T_{\sigma(n)}\}$, where:

$T_{\sigma(j)} = \max\{0, C_{\sigma(j)} - d_{\sigma(j)}\}$.

The first problem of minimizing a linear function of maximum tardiness (T_{max}) and maximum late work (V_{max}) is denoted by $1// T_{max} + V_{max}$ and called it (P1). Also this problem is a special case of the general $1// F(T_{max} , V_{max})$ problem. The problem (P1) from the class of simultaneous minimization can be formulated as follows:

$$Z1 = \text{Min}\{ T_{max}(\sigma) + V_{max}(\sigma) \}$$

$$\sigma \in S$$

Subject to

$$T_{\sigma(j)} \geq C_{\sigma(j)} - d_{\sigma(j)} \qquad j = 1,2,\dots,n$$

$$T_{\sigma(j)} \geq 0 \qquad j = 1,2,\dots,n$$

$$V_{\sigma(j)} \leq C_{\sigma(j)} - d_{\sigma(j)} \qquad j = 1,2,\dots,n$$

$$V_{\sigma(j)} \leq p_{\sigma(j)} \qquad j = 1,2,\dots,n$$

$$V_{\sigma(j)} \geq 0 \qquad j = 1,2,\dots,n$$

------- (P1)

Where S denotes the set of all feasible schedules.

The aim in problem (P1) is to find a processing order of the jobs on a single machine to minimize the sum of maximum tardiness and the maximum late work (i.e., $1// T_{max} + V_{max}$).

The second problem of minimizing a linear function of total late work ($\sum_{j=1}^{n} V_j$) and maximum late work (V_{max}) is denoted by $1// \sum_{j=1}^{n} V_j + V_{max}$ and called it (P2). Also this problem is a special case of the general $1// F(\sum_{j=1}^{n} V_j , V_{max})$ problem. The problem (P2)

from the class of simultaneous minimization can be formulated as follows:

$$Z2 = \text{Min}\{ \sum_{j=1}^{n} V_{\sigma(j)} + V_{max}(\sigma) \}$$

$$\sigma \in S$$

Subject to

$$V_{\sigma(j)} = \begin{cases} 0 & \text{if } C_{\sigma(j)} \le d_{\sigma(j)} \\ C_{\sigma(j)} - d_{\sigma(j)} & \text{if } d_{\sigma(j)} < C_{\sigma(j)} < d_{\sigma(j)} + p_{\sigma(j)} \\ p_{\sigma(j)} & \text{if } C_{\sigma(j)} \ge d_{\sigma(j)} + p_{\sigma(j)} \end{cases} , j = 1,2,\ldots,n \quad \text{---(P2)}$$

The aim in problem (P2) is to find a processing order of the jobs on a single machine to minimize the sum of total late work and the maximum late work (i.e., $1// \sum_{j=1}^{n} V_j + V_{max}$).

3.Local search algorithms

First application of local search to NP-hard problems stems from a time as early as the late 1950's and early 1960's [10]. Local search methods can find the approximation solution within a reasonable running time. Local search methods have the same feature: they explore the search space by iteratively moving from one feasible solution to another according to some defined rules. Many of these methods are inspired from nature and they explore neighborhood of feasible solution. Local search methods differ from each other in the problem representation they adopt, their definition of neighborhood on this representation and the strategy they employ while searching through this neighborhood. Some local search methods are inspired from evolution. This suggests the need for the following definition:

3.1 Solution Representation[11]

Solution representation depends on the problem specification. In a scheduling problem of n jobs, a solution is represented by a permutation of the integers 1,2,…,n.

Definition I [11]: An instance of a combinatorial optimization problem is a pair (S , f), where the solution set S is the set of all feasible solutions and the cost function f is a mapping $f:S \longrightarrow R$. The problem is to find a globally optimal (minimal) solution, i.e., an $s^* \in S$, such that $f(s^*) \leq f(s)$ for all $s \in S$.

Definition II [12]: A neighborhood function N* is a mapping

$N^*:S \longrightarrow P(S)$ which specifies for each $s \in S$ a subset $N^*(s)$ of S neighbors of s.

Three typical neighborhoods exist for the permutation representation. They can be defined by applying certain moves to a sequence of jobs [13]:

(1) Insert(shift): This neighborhood is obtained by removing a job from one position in the sequence (1,2,3,4,5,6,7,8) and insert it at another position either before (left insert) or after (right insert) the original position. For example the schedules (1,**5**,2,3,4,6,7,8) and (1,2,3,4,6,7,**5**,8) are both neighborhoods.
(2) Swap(interchange): Swap two jobs that may not be adjacent. For example the schedule (1,**6**,3,4,5,**2**,7,8) is a neighbor.
(3) Block insert: Insert a subsequence of jobs in a new position. For example the schedule (1,**4**,**5**,2,3,6,7,8) is a neighbor.

Definition III [8]: Let (S , f) be an instance of a combinatorial optimization problem and let N* be a neighborhood function. A solution $s^* \in S$ is called a local optimal (minimal) solution with respect to N* if $f(s^*) \leq f(s)$ for all $s \in N^*(s^*)$. The neighborhood function N* is called exact if every local minimum with respect to N* is also a global minimum.

Local search methods share the following features:

(1) Initialization: An initial feasible solution s is generated randomly or by applying a heuristic method and declared as the current solution. The objective function value of the current solution is computed.

(2) Neighborhood generation: A "move" is made through the solution space S from neighbor to neighbor to select a neighbor s' of s.

(3) Acceptance test: Each local search method has its own acceptance test to decide whether s' replace s as the current solution.

(4) Termination criteria: The algorithm is repeated until some termination criteria are satisfied. The output will be the best solution generated.

3.2 Descent Method (DM)

The descent method (DM) is a simple form of neighborhood search methods in which only improving moves are allowed. The resulting solution is a local optimum, not necessarily a global optimum. The structure of a descent algorithm is presented in figure (3.1).

Step (1): Select an initial solution $s \in S$;

Step (2):Choose an element $s' \in N^*(s)$;$\Delta = f(s') - f(s)$;if$\Delta < 0$

then $s = s'$,

Step (3): If $f(s') \geq f(s)$, $\forall\ s' \in N^*(s)$, then stop; else return to step (2).

Figure (3.1) structure of a descent algorithm

3.3 Simulated Annealing (SA) Algorithm

Simulated annealing (SA) has its origin in statistical physics, where the process of cooling solids slowly until they reach a low energy state is called annealing. It was originally proposed by Metropolis et al. [13] and was first applied to combinatorial optimization problems by Kirkpatrick et al. [14]. In such algorithm, the sequence of the objective function values does not necessarily monotonically decrease. Starting with an initial sequence s, a neighbor s' is generated (usually randomly) in a certain neighborhood. Then the difference $\Delta = f(s') - f(s)$, in the values of the objective function f is calculated. When $\Delta \leq 0$, sequence s' is accepted as the new starting solution for the next iteration. In the case of $\Delta > 0$, sequence s' is accepted as new starting solution with probability $\exp(-\Delta / T)$, where T is a parameter known as the temperature. Typically, in the initial stages, the temperature is rather high so that escaping from a local optimum in the first iterations is rather easy. After having generated a certain number of sequences, the temperature usually decreases. Often, this is done by a geometric cooling scheme which we will also apply.

In this case the new temperature T^{new} is chosen such that $T^{new} = \lambda T^{old}$, where : $0 < \lambda < 1$ and T^{old} denote the old temperature. A possible stopping criterion would then be a cycle of a final temperature, which is sufficiently close to zero. Since we need to be unbiased we use for all heuristics a fixed given number (20,000 in this paper) of generated solutions as stopping criterion, we determine on the base of the initial temperature T = 10 as in [15].

3.4 Genetic Algorithm (GA)

GA based on simplification of natural evolutionary process, genetic algorithms operate on a population of solutions rather than a single solution and employ heuristics such as selection, crossover and mutation to evolve better solutions. Each individual solution is represented by a string includes a set of random numbers called chromosomes. GA applies genetic operations on the individual

chromosomes which are reproduced by exchanging genes utilizing crossover. An offspring inherit some characteristics from each of its parents.

The general steps of GA are as follows:

(1) Generate random numbers for initial population (individual chromosomes) (solutions).
(2) Select two parents from a current population according to their fitness.
(3) Apply crossover to them to produce new offspring.
(4) Enter the new offspring to the population utilizing replacement strategy.
(5) Apply mutation to the random selected chromosomes.
(6) Enter the mutated chromosomes to the population using replacement strategy.
(7) Terminate the algorithm after a fixed number of generations.

Note that in scheduling problem, a chromosome includes genes that are a number of jobs that should be applied on one machine with permutation. In the following paragraph we describe each of the GA mechanisms.

(1) Initial population

Initially many individual solutions (chromosomes) are generated randomly or produced with a good heuristic to form an initial population.

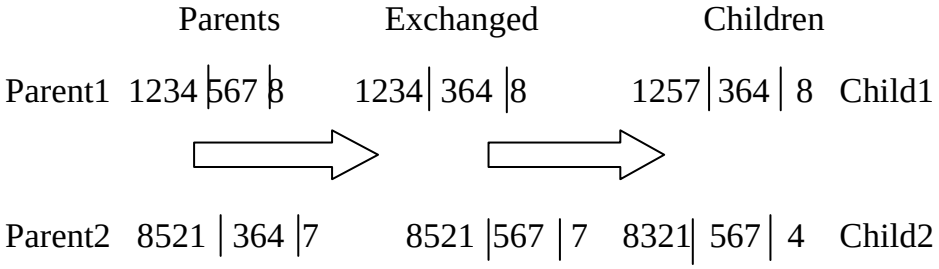
(2) Parent selection

The selection strategy is a procedure to choose the individuals in the current population for creating offspring of subsequent generation. Two parent solutions (chromosomes) are selected from a current population according to their values, the better fitness, the bigger chance to be selected.

(3) The crossover operator

The crossover operation is the most important operator in GA. We used two-point crossover (partially matched crossover) (PMX) on each pair of parent solutions to generate two new solutions.

Two-point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms. For example with crossover two point after the 4^{th} and 7^{th} element:



In other words, we made the swaps, 3 ⟷ 5, 6 ⟷ 6, 4 ⟷ 7 and then other elements remainder the same.

(4) The mutation operator

We randomly choose a chromosome and two of its genes (jobs) as a candidate to be mutated.

(5) Replacement strategy

The elitist strategy is applied in this problem. In each generation we select the best chromosome as one of the chromosomes in the next population, in addition to the chromosomes (parents and offsprings) which are ranked after crossover and mutation according to their fitness values.

(6) Stopping condition

GA is terminated if the best chromosome of the population doesn't change for 50 consecutive generations.

4. Computational experiments

This section views the computational results of local search algorithms (i.e., descent method (DM), simulated annealing (SA) and genetic algorithm (GA)) which introduced in section (3) for the problems ($1//T_{max} + V_{max}$) and ($1//\sum_{j=1}^{n} V_j + V_{max}$). Test problems are generated as follows: for each job j, an integer processing time p_j is generated from the discrete uniform distribution [1,10]. Also, for each job j, an integer due date d_j is generated from the discrete uniform distribution [P(1-TF-RDD/2),P(1-TF+RDD/2)], where $P = \sum_{j=1}^{n} p_j$, depending on the relative range of due date (RDD) and on the average tardiness factor (TF). For both parameters, the values 0.2,0.4,0.6,0.8.1.0 are considered. For each selected value of n, two problems are generated for each of the five values of parameters producing 10 problems for each value of n, where the number of jobs n = 50,100,200. All local search algorithms in this paper (descent method (DM), simulated annealing (SA) and genetic algorithm (GA)) are tested by coding it in Matlab R2009b and running on a personal computer hp with Ram 2.50 GB. Each algorithm stops when it attends to a fixed number of iteration. The (DM) and (SA) stop after 20,000 iteration and (GA) stops after 100 new generation of population.

For the problem (P1) tables (4.1) to (4.3) show the values of each local search algorithm and how many times each of them catches the best value. For the problem (P2) tables (4.4) to (4.6) show the values of each local search algorithm and how many times each of them catches the best value. Each table contains 10 problems where:

EX = the number of example.

DM = descent method performance.

SA = simulated annealing performance.

GA = genetic algorithm performance.

Best = the best performance.

No. best = number of examples that catch the best performance.

Av. Time = the average of time in second.

Table (4.1): The performance of local search algorithms for n=50.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	33	33	0.803423	33	0.485376	33	2.2149
2	14	14	0.485881	14	0.476210	14	2.1998
3	55	55	0.482367	59	0.478526	55	2.2146
4	24	24	0.476799	24	0.480008	24	2.2085
5	0	0	0.481222	0	0.488164	0	2.2074
6	55	55	0.468660	56	0.492309	55	2.2515
7	37	37	0.473350	38	0.482789	37	2.2172
8	64	64	0.481144	69	0.478321	64	2.2074
9	94	94	0.468486	94	0.486744	94	2.2218
10	133	133	0.476539	133	0.495805	133	2.2011
No. best		10	Av. Time 0.5097871	6	Av. Time 0.4844252	10	Av. Time 2.21442

Table (4.2): The performance of local search algorithms for n = 100.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	281	281	0.752799	281	0.618218	281	5.2176
2	345	345	0.632327	345	0.620342	345	5.1908
3	334	334	0.630096	334	0.642404	334	5.1451
4	183	183	0.623157	183	0.756629	183	5.1197
5	496	496	0.628466	496	0.702720	496	5.1533
6	133	133	0.626593	133	0.636938	133	5.1359
7	22	22	0.628908	22	0.633847	22	5.2142
8	120	120	0.624681	120	0.643624	120	5.1146
9	345	345	0.631848	345	0.636874	345	5.1641
10	95	95	0.629514	95	0.640907	95	5.1668
No. best		10	Av. Time 0.6408389	10	Av. Time 0.6532503	10	Av. Time 5.16221

Table (4.3): The performance of local search algorithms for n = 200.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	319	319	0.920927	319	1.089923	319	16.4951
2	138	138	0.963153	138	0.969849	138	16.2566
3	830	830	0.934056	830	0.937151	830	15.5157
4	30	31	0.936466	31	0.944838	30	15.4509
5	0	0	0.936650	0	0.962992	0	15.3982
6	357	357	0.950734	357	0.987751	357	15.5027
7	352	352	0.936504	352	0.949816	352	15.5463
8	146	146	0.940228	146	0.925345	146	15.3809
9	57	57	0.929991	57	0.928336	57	14.7551
10	233	233	0.929631	233	0.941485	233	15.4948
No. best		9	Av. Time 0.937834	9	Av. Time 0.9637486	10	Av. Time 15.57963

Results of the above tables show that the (GA) performs very well and then the (DM) gives reasonable results. The average computation time of (SA) is close to that of (DM) while for (GA) is large.

Table (4.4): The performance of local search algorithms for n = 50.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	38	56	0.621646	62	0.462542	38	1.9022
2	14	18	0.465414	17	0.466724	14	1.8758
3	61	80	0.448073	71	0.468660	61	1.8781
4	24	35	0.462600	33	0.453508	24	1.9018
5	0	0	0.455166	0	0.454749	0	1.8815
6	62	74	0.451512	64	0.447186	62	1.9003
7	41	41	0.490248	50	0.455103	44	1.8713
8	65	72	0.447783	66	0.462686	65	1.8766
9	94	104	0.450303	98	0.456187	94	1.8606
10	133	136	0.446130	138	0.468994	133	1.8821
No. best		2	Av. Time 0.4738875	1	Av. Time 0.4596339	10	Av. Time 1.88303

Table (4.5): The performance of local search algorithms for n = 100.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	281	309	0.579330	289	0.697297	281	4.7017
2	348	378	0.591284	358	0.608357	348	4.8789
3	337	355	0.588728	347	0.597732	337	4.6866
4	191	201	0.599128	197	0.601212	191	4.9383
5	496	497	0.589819	496	0.600085	497	4.6737
6	158	203	0.596384	180	0.602028	158	4.6113
7	27	32	0.598575	42	0.605930	27	4.6847
8	135	150	0.595046	160	0.599502	135	4.9194
9	345	348	0.603438	360	0.592063	345	4.6762
10	111	163	0.597757	140	0.620136	111	4.6745
No. best		0	Av. Time 0.5939489	1	Av. Time 0.6124342	9	Av. Time 4.74453

Table (4.6): The performance of local search algorithms for n = 200.

EX	Best	DM	Time for DM	SA	Time for SA	GA	Time for GA
1	369	438	1.087852	424	0.897394	369	15.3706
2	301	311	0.887854	343	0.888566	301	14.1156
3	834	910	0.925646	876	0.892262	834	14.0318
4	30	56	0.883290	66	0.890235	30	15.7047
5	0	0	0.899873	7	0.857499	0	14.5430
6	404	470	0.890592	453	0.888713	404	13.7467
7	397	459	0.896993	453	0.895019	397	14.7598
8	304	342	0.893320	355	0.913652	304	14.7038
9	91	127	0.889341	136	0.896170	91	14.9116
10	302	370	0.882687	371	0.900059	302	14.7030
No. best		1	Av. Time 0.9137448	0	Av. Time 0.8919569	10	Av. Time 14.65906

5.Conclusions

In this paper local search algorithms (descent method (DM), simulated annealing (SA) and genetic algorithm (GA)) are proposed to find approximation solutions for the problems of minimizing $T_{max} + V_{max}$ and $\sum_{j=1}^{n} V_j + V_{max}$. A computational experiment for the local search algorithms on a large set of test problems are given. The main result is that the genetic algorithm (GA) is more effective for our problems.

An interesting future research topic would involve experimentation with the approximation algorithms for large n and experimentation with the following multicriteria problems :

(1) $1//Lex(T_{max}, V_{max})$.
(2) $1//F(T_{max}, V_{max})$.
(3) $1//Lex(\sum_{j=1}^{n} V_j, V_{max})$.

References

1. Blazewicz J., Ecker K., Pesch E., Schmidt G., Weglarz J., "Handbook on scheduling. From theory to applications", Berlin- Heidelberg -New York : Springer; 2007.
2. Brucker P., "Scheduling algorithms", Berlin-Heidelberg-New York : Springer; 2007.
3. Pinedo M., "Scheduling : theory, algorithms and systems", New York : Springer; 2008.
4. Blazewicz J., "Scheduling preemptible tasks on parallel processors with information loss", Technique et Science Informatiques , 3 (6):415-20; 1984.
5. Potts CN., Van Wassenhove LN., "Single machine scheduling to minimize total late work", Operations Research, 40 (3): 586-95; 1991.

6. Hochbaum DS, Shamir R., "Minimizing the number of tardy job units under release time constraints", Discrete Applied Mathematics 28 (1) : 45-57; 1990.
7. Blazewicz J., Pesch E., Sterna M., Werner F., "Total late work criteria for shop scheduling problems". In : Inderfurth K., Schwoediauer G., Domschke W., Juhnke F., Kleinschmidt P., Waescher G., editors. "Operations Research Proceedings 1999", Berlin : Springer P.354 – 9; 2000.
8. Alaminana M., Escudero LF., Landete M., Monge JF., Rabasa A., Sanchez-Soriano J., WISCHE ; "A DSS for water irrigation scheduling", Omega 38 (6): 492-500; 2010.
9. Leung JY-T., "Minimizing total weighted error for imprecise computation tasks and related problems", In : JY-T Leung. editor. "Handbook of scheduling : algorithms, models and performance analysis", Boca Raton : CRC Press 34: 1-16; 2004.
10. Evans G. W., "An overview at techniques for solving multi objective mathematical programs", Management Science 30: 1268-1282; 1984.
11. Arats E. H. L. and Lenstra J. K., "Local search in combinatorial optimization", John Wiley and Sons. Chichest; 1997.
12. Vaessens R. J. M., Arats E. H. L., and Lenstra J. K., "A local search template", Computers and Operations Research 25 : 969-979; 1998.
13. Mahmood A. A., "Solution procedures for scheduling job families with setups and due dates" M.Sc. Thesis, Univ. of Al-Mustansiriyah, College of Science, Dept. of Mathematics 2001.
14. Kirkpatrick S., Gelatt JR. CD. and Vecchi MP., "Optimization by simulated annealing", Science 220 : 671-80; 1983.
15. Mohammed H. A. A., "Using genetic and local search algorithms as a tool for providing optimality for job

scheduling", M.Sc. Thesis, College of Science. University of Al- Mustansiriyah 2005.

حلول تقريبية لمسائل جدولة متعددة المقاييس

م.عدوية علي محمود النعيمي
adawiyaali@yahoo.com
جامعة ديالى - كلية العلوم - قسم الرياضيات
أ.د. طارق صالح عبد الرزاق
dr.tariqsalih@yahoo.com
الجامعة المستنصرية - كلية العلوم - قسم الرياضيات

المستخلص

ان هذا البحث يقدم خوارزميات بحث محلية لإيجاد حلول تقريبية لمسائل جدولة متعددة المقاييس على ماكنة واحدة حيث المسألة الأولى هي المجموع لأعظم تأخير لاسالب واعظم تأخير لوحدات عمل متأخر والمسألة الثانية هي المجموع لوحدات عمل متأخر كلي واعظم تأخير لوحدات عمل متأخر.

مقياس العمل المتأخر يخمن كفاءة الجدول بالاعتماد على فترات زمنية للأجزاء المتأخرة للأعمال . اقترحت خوارزميات البحث المحلية وهي طريقة النزول ، طريقة تقوية المحاكاة والخوارزمية الجينية . بالاعتماد على نتائج التجارب الحسابية تم صياغة استنتاجات حول كفاءة خوارزميات البحث المحلية.

الكلمات الرئيسية: بحث محلي، خوارزمية جينية، مقياس عمل متأخر، جدولة متعددة المقاييس.