Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

An approximation solution for scheduling problem in single machine under unavailability constraints

Omar Selt [1,*], Hachemi Benouadah [2]

[1]Laboratory of Pure and Applied Mathematics, Department of Mathematics, University of M'sila, Algeria
[2]Laboratory of Economic Policies and Strategies in Algeria, Department of Financials Sciences and Accounting, University of M'sila, Algeria

ARTICLE INFO

Article history:
Received 15 September 2016
Received in revised form
10 November 2016
Accepted 17 November 2016

Keywords:
Scheduling
Single machine
Heuristic
Metaheuristic
Tabu search

ABSTRACT

In this paper, an approximation solution for scheduling problem of *n* tasks on single machine with unavailability zones is proposed. This problem is strongly NP-complete which makes finding an optimal solution looks impossible task. In this frame, we suggested a heuristic (H_1) in which availability periods of machine are filled with the highest weighted tasks. To improve the performance of this heuristic, we used, on one hand, different diversification strategies $E_1$and E_2 with the aim of exploring unvisited regions of the solution space, and on the other hand, two neighborhoods (neighborhood by swapping and neighborhood by insertion). The computational experiment was carried out on single machine with different unavailability zones. It must be noted that tasks movement can be within one period or between different periods.

1. Introduction

A scheduling problem consists in organizing tasks realization time with consideration of time constraints (time limits, tasks series character) and constraints related to using and availability of required resources. The scheduling constitutes a solution to the considered problem, describes the tasks execution and resources location during time and aims to satisfy one or many objectives (Zribi et al., 2005) have studied the problem $1/N-C/\sum_{j=1}^{n} w_j C_j$ and have compared two exact methods: one is the Branch and Bound, the other is the integer programming. They have concluded that Branch and Bound method have better performance and it allowed resolving instances of more than 1000 tasks.

Chang et al. (2011) proposed a genetic algorithm (GA) enhanced by dominance properties for single machine scheduling problems to minimize the sum of the job's setups and the cost of tardy or early jobs related to the common due date.

Zitouni and Selt (2016) have studied the problem: $\frac{P_m}{N}-\frac{C}{\sum_{j=1}^{n} w_j C_j}$; they carried out a comparative study of heuristic and metaheuristic for three identical parallel machines.

In this, paper, the results of selt and zitouni (2014) research works are exploited to develop a different new heuristic to solve the tasks scheduling problem on single machine under different constraints

2. Tabu search

Tabu search is metaheuristic that keeps track of the regions of the solution space that have already been searched in order to avoid repeating the search near these areas. It starts from a random initial solution and successively moves to one of the neighbors of the current solution.

The difference of tabu search from other Meta-heuristic approaches is based on the notion of tabu list, which is a special short term memory. That is composed of previously visited solution s that includes prohibited moves. In fact, short term memory stores only some of the attributes of solutions instead of whole solution. So it gives no permission to revisited solutions and then avoids cycling and being stuck in local optima.

During the local search only those moves that are not tabu will be examined if the tabu move does not satisfy the predefined aspiration criteria. These aspiration criteria are used because the attributes in

* Corresponding Author.
Email Address: selt.omar@yahoo.fr (O. Selt)
https://doi.org/10.21833/ijaas.2016.12.001

the tabu list may also be shared by unvisited good quality solutions. The process of TS can be represented as follows.

Algorithm

Step 1: Generate initial solution x.
Step 2: Initialize the Tabu List.
Step 3: While set of candidate solutions X" is not complete.
Step 3.1: Generate candidate solution x" from current solution x
Step 3.2: Add x" to X" only if x" is not tabu or if at least one Aspiration Criterion is satisfied.
Step 4: Select the best candidate solution x* in X".
Step 5: If fitness(x*) > fitness(x) then x = x*.
Step 6: Update Tabu List and Aspiration Criteria
Step 7: If termination condition met finish, otherwise go to Step 3.

2.1. Neighborhood structure

Neighborhood determination constitutes the most important stage in metaheuristic methods elaboration. In the following part, we use two Neighborhoods, (neighborhood by swapping) and (neighborhood by insertion).

2.1.1. Neighborhood by swapping

Definition

Consider a sequence σ composed of n tasks. A neighborhood σ' is obtained by permuting two tasks. j and j' of respectively k and k'positions σ with $k' = k+1, k+2, \dots, n$. The set,

$$N_1(\sigma) = \{\sigma, \sigma \text{ is obtained by permutation of two tasks}\}$$

is called neighborhood of σ This set is consequently obtained by permutation of all tasks of σ two by two.

Formal statement 1

Consider a sequence σ, the set's cardinal of $N_1(\sigma)$ is $\frac{n(n-1)}{2}$.

Proof: The permutation of all tasks; two by two consists in permuting each task of the sequence with all remained tasks; without identical ones. The number of possible permutations in a sequence σ composed of n tasks is:

$$(n-1) + (n-2) + \cdots + 2 + 1 = \frac{n(n-1)}{2}$$

2.1.2. Neighborhood by insertion

Definition

Consider a sequence σ composed of n tasks; a neighborhood σ' is obtained by inserting one task j of a position k in a new position k' in the sequence σ. The set $N_2(\sigma) = \{\sigma', \sigma' \text{ is obtained by inserting a task of position } k \text{ in } k'\}$ is a neighborhood of σ. This set is consequently obtained by realizing all possible insertions of all tasks of σ.

Formal statement 2

Consider a sequence σ, the set's cardinal of $N_2(\sigma)$ is $(n-1)^2$.

Proof: Inserting a task j of position k in another position k' in the sequence σ allows getting n-1 possible insertions. Hence, for n tasks, there is n(n-1) insertions to be done. To avoid getting identical sequences, adjacent tasks insertions are counted once. Consequently n-1 insertions will be deleted. Finally, the number of obtained insertions is:

$$n(n-1) - (n-1) = (n-1)^2.$$

Formal statement 3

It must be noted that tasks movement can be within one period or between different periods.

2.2. Tabu list structure

The tabu method is based on the principle that consists in maintaining in memory the last visited solutions and in forbidding the return to them for a certain number of iterations. The aim is to provide sufficient time to the algorithm so it can leave the local optimum. In other words, the tabu method conserves in each stage a list L of solutions (Tabu's) which it is forbidden to pass-by temporarily. The necessary space for saving a set of solutions tabus in the memory is indispensable.

The list, that we propose, contains the found solutions sequences. After many tests, a dynamic size list, which varies according to the search amelioration state, is conceived. The initial size of this list is considered to be $\frac{3\sqrt{n}}{2}$ where n is the tasks number. After that, during the search, when 5 successive iterations pass without amelioration of solution, the list is reduced to a number inferior or equal to$\sqrt{n}$. On the other hand, when 5 successive iterations pass and the solution is ameliorated, the list is increased to a number superior or equal to $2\sqrt{n}$. The Tabu list is consequently dynamic and its size varies within the interval $[\sqrt{n}, 2\sqrt{n}]$. The decrease or the increase of list size must always be done at the end of the list.

2.3. Heuristic

An initial solution is always necessary. For this reason, we suggest in this part the following heuristic: assigne the (best) task h where $(\frac{P_k}{W_k} = \min_{j \in J}\{\frac{P_j}{W_j}\})$ to the best machine based on two principles justified by the two following propositions:

Formal statement 4

In an optimal scheduling, it is necessary to schedule the tasks; in each availability period of the machine according to the order SWPT.

Proof: It results directly by adjacent task exchange like used by Smith (1956) for the corresponding periods.

Formal statement 5

It is not useful to let the machine (idle) if a task can be assigned to this machine.

Notations:
We denote by:
$J = \{1, 2, \dots, n\}$: *The set of tasks.*
P_h: *Execution time of the task h.*
I: Single machine
k: Number of availability zones.
Z={1, 2, …, k}: Availability zones.
E_z: *Period of unavailability zones.*
σ: Sequence assigned to machine I
W_h: *Weight of the task h*
C_h: *Execution time of the task h by the machine I*
$C_z(z \epsilon Z)$: *Execution time of the task* $j \epsilon J_z$, *allocated to the zone z.*
f(σ): Objective function cost.
f_{swapp}: *Swapping algorithm cost.*
f_{insert}: *Insertion algorithm cost.*
Initialization
j={1, 2, …, n}; $E_1 = 0$; σ = ∅, $f(\emptyset) = 0$; $P_j =$ random (1.99); $W_j = random$ (1.10); $z = 1$ Begin Sort tasks $h \in j$ in increasing order according to the criterion $\frac{p_j}{w_j}$ in a list U_1.

Sort tasks $h \in J$ in decreasing order according to the criterion p_j in a list U_2

While ($U \neq \emptyset$ *and* $z_k \geq P_h$ *do*
Begin
Set $p_{h_1} = \frac{P_h}{w_h}$ *from the top list of* U_1
Set $p_{h_2} = max\ p_h$ *from the top list of* U_2
End if
Assigned the task h to the machine I;
Delete the task h from the two list U_1 *and* U_2;
$C_z = \sum p_j + E_z$;
Compute
Determine $\sigma = \sigma \cup \{h\}$ *and* $f_\sigma = f_\sigma + w_h C_z$;
and
End
Else
Begin
Set Z=Z+1;
End
End if
End

Algorithm
Step 1: Get an initial solution σ and T [1]=0 ;
Step 2: Do permutation by swapping
Step 3: Do permutation by insertion
Step 4: Compute: $f_1 = f_{swapp}$; $f_2 = f_{insert}$
Step 5: Consider L(Tabu list size)
Step 6: for k=1 to 3 Do
If $f_{init} < f_k$ *Do*
$T[1] = f_{init}$;
else $T[1] = f_k$;
End if
$T_k = T[1]$;
End
Step 6.1: $f_{best} = min\ (T_1, T_2, T_3)$
End if
Step 6.2: Display $\sigma(f_{best})$

2.4. Diversification strategies

The final time to execute this problem is chosen as T2700. It is divided according to diversification strategy to two times E_1 and E_2. After many experiments, these periods are chosen as follows:

E_1 =1700s Initial starting time: uses long term memory to store the frequency of the moves executed through of the search.

E_2=1000s restarting time makes use of influential moves.

3. Numerical example

Consider the problem P 1 with the following data:

Table 1: 6 Tasks scheduling results

j	1	2	3	4	5	6
Pj	11	36	88	10	91	31
Wj	3	6	8	7	4	1
Pj/Wj	3.66667	6	11	1.42	22.75	31
Pj(MAX)	91	88	36	31	11	10

6 Tasks scheduling results for example:
Results of heuristic (H_2) *are: f = 2548;*
Execution time = 0,650 s;
Results of tabu (swapping) are: f = 1986;
Execution time = 0,991 s;
Results of tabu (insertion) are: f = 2367;
Execution time = 1,542 s;
Execution time = 0,945

The best results are obtained by using by tabu (swapping) for f = 1986**.**

4. Computational analysis

Data generation

The heuristic were tested on problems generated with 500 tasks similar to that used in previous studies (Ho and Chang, 1995; M'Hallah and Bulfin, 2005) for each task j an integer processing time p_j was randomly generated in the interval (1.99) with a weight randomly w_j chosen in interval (1.10). The Table 2 presents:

1- The initial mean values of objective function corresponding to initial sequence.

2- The initial mean values of objective function obtained by using on one hand, the neighborhood by

swapping and on the other hand, the neighborhood by insertion.

3- The average times corresponding to the two neighborhoods.
4- The best costs.

Table 2: Heuristic cost amelioration

Initial Solution (average of 3 instances)		Tabu search by swap		Tabu search by insertion		Best costs
AC	AT (second)	AC	AT (second)	AC	AT (second)	
49635	0,897	44957	1,83	44845	2,76	44845
41544	1,01	41235	1,72	41094	2,64	41094
34220	0,99	33020	1,97	34008	2,38	33020
202482	4,43	200848	9,25	201830	14,40	200848
220786	6,65	220364	10,34	210600	16,03	210600
230501	3,98	197233	9,16	226582	14,19	197233
903625	7,65	834570	14,25	856713	28,09	834570
863040	9,86	791284	15,18	786173	30,53	786173
875476	10,23	774283	16,03	855364	27,98	774283
989476	20,14	970528	45,78	985476	35,80	970528
1098437	28,76	925376	39,73	906718	42 ,71	906718
1287142	27,97	1001583	43,91	973027	40,93	973027
22206778	40,12	19765183	69,03	21165329	71,63	19765183
15016104	45,67	13489345	72,87	13987543	79,51	13489345
21646539	42,56	21135631	61,42	21598743	67,91	21135631

5. Conclusion

In this paper, scheduling problem with single machine and availability zones is solved using a novel metaheuristic polynomial approach (Tabu search) with complexity O (ln n). The tabu list of this problem is dynamic and its size varies according to amelioration state of the solution. The developed approach is based on diversification strategy using solution search algorithm that restarts from the point of the solution that was chosen among the earlier best unmaintained found solutions. According to the curried out tests, it is concluded that the proposed approach ensure acceptable results.

It must be noted that the neighborhood by swapping presents the best costs with an acceptable execution time.

References

Chang PC, Chen SH, Lie T and Liu JYC (2011). A genetic algorithm enhanced by dominance properties for single machine scheduling problems with setup costs. International Journal of Innovative Computing, Information and Control, 7(5): 1-21.

Ho JC and Chang YL (1995). Minimizing the number of tardy jobs for m parallel machines. European Journal of Operational Research, 84(2): 343-355.

M'Hallah R and Bulfin RL (2005). Minimizing the weighted number of tardy jobs on parallel processors. European Journal of Operational Research, 160(2): 471-484.

Selt O and Zitouni R (2014). A comparative study of heuristic and metaheuristic for three identical parallel machines. Canadian Journal of Pure and Applied Science, 8(3): 3147-3153.

Smith WE (1956). Various optimizers for single-stage production. Naval Research Logistics Quarterly, 3(1-2): 59-66.

Zitouni R and Selt O (2015). Metaheuristics to solve a tasks scheduling problem in parallel identical machines with unavailability periods. RAIRO-Operations Research, 50(1): 83-90.

Zribi N, Kacem I, El-Kamel A and Borne P (2005). Minimisation de la somme des retards dans un jobshop flexible. Revue e-STA (SEE), 2(2): 22-27.