Management Science Letters 2 (2012) 673–680

Contents lists available at GrowingScience

Management Science Letters

homepage: www.GrowingScience.com/msl

Optimization costs of the single-machine scheduling problem with maintenance activities by using genetic algorithm

Mahin Esmaeili*

Department of Mathematics and Computer Science, Shahid Bahonar University , Kerman, Iran

ARTICLEINFO

Article history:
Received July 22, 2011
Received in Revised form
October, 12, 2011
Accepted 15 October 2011
Available online
25 October 2011

Keywords:
Scheduling problem
Maintenance
Genetic algorithm

ABSTRACT

This paper deals with a single-machine scheduling problem with maintenance activities. Our purpose is to provide a near optimal solution using metaheuristics approach. In this problem, there are n jobs and m machines $(m > n)$, each job must be assigned to one and only one machine, where the processing time of job (j) is (p_j)**.** Furthermore there are M_G groups where each group has a fix periodic interval T and for each group, the maximum number of jobs processed in the machines available time interval (T) is K, $\left(M_G = \frac{m}{K}\right)$. For finding the near optimal solution, we consider optimizing total cost scheduling problem. This problem has two types of costs, *group cost* and *gap cost*. In this study, first, proposed problem is formulated in a mathematical model. Next, a heuristic genetic algorithm is used to obtain the proposed problem and on example is presented to verify the efficiency of the algorithm.

1. Introduction

The problem of scheduling jobs with maintenance activities is one of the most vital issues in service systems and industrial companies. The primary objective is to obtain cost of periodic maintenance in process industries. In this problem, machines must be stopped for periodical maintenance (Pinedo, 2002, Chen, 2006). Sbihi and Varnier (2008) considered a single-machine scheduling problem with several maintenances periods under two different scenarios. In the first one, maintenance periods were periodically fixed while in the second one, the maintenance was not fixed but the maximum continuous working time of the machine allowed was determined. The objective was to minimize the maximum tardiness, which are known to be strongly NP-hard. They proposed some dominance properties and an efficient heuristic.

Chen (2009) considered a single-machine scheduling problem with periodic maintenance where a schedule consists of several maintenance periods and each maintenance period is scheduled after a periodic time interval. He determined a schedule, which minimizes the number of tardy jobs subject to periodic maintenance and nonresumable jobs and solved the resulted problem using a heuristic

* Corresponding author.
E-mail addresses: Mahin-Esmaeili <m_esmaeili90@yahoo.com>

doi: 10.5267/j.msl.2012.10.014

approach and compared his results with optimal solution achieved from a branch-and-bound algorithm. Change et al. (2009) investigated a new meta-heuristic, which implements a methodology to solve the single machine scheduling problem by using the random-key concept combining with genetic operators in the hybrid algorithm to determine the best schedule for the single machine problems. The approach attempts to achieve the convergence and diversity effects when it is iteratively applied to solve the problem.

They implemented their proposed hybrid algorithm on a set of standard test problems available and reported promising results compared with the standard genetic algorithm. Hsu et al. (2010) introduced a single-machine scheduling problem with periodic maintenance activity under two maintenance stratagems. They introduced a single-machine scheduling problem where the machine ought to be interrupted for maintenance after a fixed periodic interval or after a fixed number of jobs. The paper minimizes the makespan using a two-stage binary integer programming for driving the optimal solution up to 350-job instances.

Low et al. (2010) presented a particle swarm optimization (PSO) algorithm to provide solution strategy for the single-machine scheduling problem with periodic maintenance activities. They discussed that the most important problem for PSO implementation is the procedure on developing an effective 'problem mapping' and 'construction of a particle sequence' mechanism. For the problem mapping aspect, they introduced the ''job-to-position" representation for the particles. The objective was to determine a schedule, which minimizes the makespan. The addressed problem is demonstrated to be NP-hard in the strong sense by transforming to the 3-partition problem.

The organization of this paper first introduces the concepts of the single–machine scheduling problem with maintenance activities (SMMA). Next, it introduces the mathematical models of the proposed problem in section 2 and introduces a heuristic genetic algorithm to solve the proposed programming model in Section 3. The implementation of the proposed model is given for an example in section 4 and the remarking conclusion is given in Section 5.

2. Preliminaries

2.1 The single – machine scheduling problem with maintenance activities

Consider a set of n independent jobs $\{J_1, J_2, \ldots; J_n\}$, m machines $\{M_1, M_2, \ldots; M_m\}$, which has to be scheduled without preemptions on a single machine, which could handle at most one job at a time. The machine is assumed to be continuously available from time zero onwards and unforced machine idle time is not allowed. Let p_j be the processing time of job j and t be the amount of time to perform each maintenance activity. Let T be the length of the time interval between two consecutive maintenance periods, ga_i be the gap

(i.e. the idle time) between the total processing time group g_i and T. Let K be the maximum number of jobs processed in the machine's available time interval T and M_G be the number of the groups. We think of each interval between two consecutive maintenance activities as a group (G) with a capacity of T. The minimum number of groups (L) required for processing n jobs and the minimum total gap within the first is $L-1$ groups (Hsu et al., 2010). The group cost (c_G) is related to the maintenance activities for each group. The gap cost (c_{Ga}) is related to the gaps the single – machine scheduling problem with maintenance activities. Consider decision variable x_{ij} as follows,

$$x_{ij} = \begin{cases} 1 & \text{if job } j \text{ is scheduled to machine } i, i = 1,\ldots,m \\ 0 & \text{otherwise} \end{cases}$$

For ease of convenience, any scheduled can also be denoted by such a vector x and costs variables c_{Ga}, c_G. Therefore, the cost function of assignment x can be expressed as

$$z(x, c_{\mathrm{Ga}}, c_G) = \sum_{s=1}^{M_G} c_G \sum_{i \in G_S} \sum_{j=1}^{n} p_j x_{ij} + c_{\mathrm{Ga}} (L-1)t$$

Therefore, the proposed scheduled problem has the following form

Stage 1:

$$\min \sum_{s=1}^{M_G} C_G \sum_{i \in G_S} \sum_{j=1}^{n} p_j x_{ij} + C_{Ga}(L-1)t \tag{1}$$

subject to

$$\sum_{i=1}^{m} x_{ij} = 1, j = 1, \cdots, n \tag{2}$$

$$\sum_{i=1}^{n} p_j x_{ij} \le T, i \in G_s, s = 1, \cdots, L \tag{3}$$

$$\sum_{j=1}^{n} x_{ij} \le K, i \in G_s, i = 1, \cdots, L \tag{4}$$

$$x_{ij} = 0,1, i = 1, \cdots, m, j = 1, \cdots, n \tag{5}$$

From now on, we assume that c_{Ga} and c_G are costs variables. The objective given in Eq. (1) is the minimization of the cost schedule where *n* jobs are allocated to *kth* groups. Constraints (2) ensure that each job must be assigned just to one group. Constraints (3) restrict the processing time for each group. Constraints (4) restrict the number of jobs assignment problem in each group. Moreover, Constraint (5) set up the binary restrictions for x_{ij}.

Definition A assignment x^* is called the optimal solution (optimal scheduling) problem Stage 1, if

$$z(x^*, c_{\mathrm{Ga}}, c_G) \le z(x, c_{\mathrm{Ga}}, c_G)$$

for any assignment x.

3. Heuristic genetic algorithm

In this section, a heuristic genetic algorithm is considered for solving the single – machine scheduling problem with maintenance activities.

3.1 Representation

Representation is one of the most important stages for the genetic algorithm. There are many ways to represent a solution of optimization problem. In this research, a chromosome is a set of integer value and the length of the chromosome can be exactly defined as a number m, which denotes the maximum number of the available processing jobs. A chromosome is represented as an array $S = \{s_1, s_2, \ldots, s_m\}$, where the value of s_i is equal to the index of the job to which the machine i is received, where i belongs *kth* group ($i \in G_k$) with ($m > n$). Here, *N* denotes the population size and the number of chromosomes is equal *N*. Therefore, we have,

$$s_i = \begin{cases} j \in J & if \ \ the \ job \ j \ \ assigned \ \ \ machine \ i \ \ that \ (i \in G_k) \\ 0 & if \ \ the \ machine \ i, i \in G_k \ do \ not \ receive \ any \ job \end{cases}$$

3.1.1. Initialization process

The initialization process of this problem can be described as follows: Let $j = 1$, randomly select machine i_j from interval $[1, m]$. Assume that $i_j \in G_k$, if $i_j \neq i_1, \ldots, i_{j-1}$ and the total of processing time kth group's jobs are less than T, in the chromosome S, then job j assigns machine i and let $j = j + 1$, otherwise select another machine, repeat this process until all jobs are assigned. We consider *N* as the population size. Therefore, the number of chromosomes is equal to *N*. We initialize chromosomes $S_1, S_2, \ldots, S_N$ by repeating the following algorithm *N* times.

Step 1. For $i = 1$ to *N*, repeat Steps 2 *N* times,

Step 2. Let $S_i[i'] = 0,\ i' = 1, \ldots, m,$

Step 3. Let $j = 1$, repeat Step 4 to 8 until $j = n$,

Step 4. Randomly generate a positive integer i_j from the interval $[1, m]$,

Step 5. Let machine i_j belong kth groups (i.e $i_j \in G_k$),

Step 6. For $i' = 1$ to m, repeat Step 7, m times,

Step 7. If $i' \in G_k$ then let $P_{G_k} = p_{S_i[i']} + P_{G_k}$,

Step 8. If $P_{G_k} + p_{i_j} \leq T$ and $i_j \neq i_1, \ldots, i_{j-1}$ then assign job j to machine i_j: $S_i[i_j] = j$, let $j = j + 1$, otherwise go to Step 4.

Obviously, all the chromosomes generated by above algorithm are feasible.

3.1.2 Crossover operation

Let $p_{cross} \in (0,1)$ be the crossover probability. In order to determine the parents for crossover operation, we repeat the following process from $i = 1$ to N: randomly generating a real number r from the interval $(0,1)$, the chromosome S_i is selected as a parent if $r < p_{cross}$. Let chromosomes (S_1', S_2') is selected from the chromosomes $S_1, S_2, \ldots, S_N$ for the crossover process. We use uniform crossover with a random mask chromosome P_{01}.

For $i = 1$ to m, randomly select a integer p from the set $\{0,1\}$ then let $S_{01}[i] = p$. For $i = 1$ to m, Let $S_{01}[i]$ is equal 0, and $S_1'[i] = j_i$, assume that $i \in G_k$ if $j_i \neq j_1, \ldots, j_{i-1}$ and the total of processing time kth group's jobs are lesser than T, in the chromosome S_1'', then $S_1''[i] = S_1'[i] = j_i$, otherwise randomly generate a positive integer j_i from the interval $[1, n]$ and consider all constrain problem then let $S_1''[i] = j_i$, otherwise randomly select another job. But, if $S_{01}[j] = 1$, similarly repeat the upper method, $S_1''[i] = S_2'[i] = j_i$, see, e.g., the researches see, e.g., the researches of Shasavari Pour et al. (2010). The crossover operation can be described as the following algorithm:

- Crossover algorithm:

Step 1. For $i = 1$ to m, repeat Step 2, n times,

Step 2. Randomly generate a positive integer p from the set$\{0,1\}$. Let $S_{01}[j] = p$,

Step 3. Let $i = 1$, repeat Step 4-11 until n,

Step 4. If $S_{01}[j] = 0$ then go to Step 6, otherwise go to Step 7,

Step 5. Let $S_1'[i] = j_i$,

Step 6. Let $S_2'[i] = j_i$,

Step 7. Let machine i belong kth group (i.e $i \in G_k$),

Step 8. For $i' = 1$ to m, repeat Step 9, m times,

Step 9. If $i' \in G_k$ then let $P_{G_k} = p_{S_1''[i']} + P_{G_k}$,

Step 10. If $P_{G_k} + p_{j_i} \leq T$ and $j_i \neq j_1, \ldots, j_{i-1}$ then assign job j_i to machine i: $S_1''[i] = j_i$, let $i = i + 1$, Otherwise go to Step 7,

Step 11. Randomly generate a positive integer j_i from the interval $[1, n]$ go to Step10.

3.2 Mutation operation

Let $P_{mut} \in (0,1)$ be the mutation probability. We use the following operator to select the chromosome to be mutated: for $i = 1$ to N, randomly generate a real number r from interval (0,1); if $r \leq P_{mut}$, then the chromosome S_i is selected to be mutated.
Let S be the chromosomes $S_1, S_2, \ldots, S_N$ for the mutation process. Randomly select two jobs j_1, j_2 assigned to the machines i_1, i_2 in the chromosome S, respectively, so $S[i_1] = j_1$, $S[i_2] = j_2$, then exchange the jobs j_1 and j_2. In the Fig. 2, the workers 2 and 7 are selected.

- Mutation algorithm

Step 1. For $i = 1$ to N, repeat Steps 2–8 N times,

Step 2. randomly generate a real number r from interval (0,1); if $r \leq P_{mut}$, then go to Step 3, Otherwise, go to Step 1,

Step 3. Randomly select two machines i_1, i_2 from the interval $[1, m]$, $(i_1 \in G_{k1}, i_2 \in G_{k2})$, assume $S[i_1] = j_1$, $S[i_2] = j_2$,

Step 4. For $i' = 1$ to m, repeat Step 5, m times,

Step 5. If $i' \in G_{k1}$ then let $P_{G_{k1}} = p_{S[i']} + P_{G_{k1}}$ and $i' \in G_{k2}$ then let $P_{G_{k2}} = p_{S[i']} + P_{G_{k2}}$,

Step 6. If $P_{G_{k1}} + p_{j_2} - p_{j_1} \leq T$ and $P_{G_{k2}} + p_{j_1} - p_{j_2} \leq T$ then go to Step 7,

Step 7. Exchange the jobs j_1 and j_2 in the chromosome S, respectively, by the operation $j = S[i_1]$, $S[i_1] = S[i_2], S[i_2] = j$.

3.3 Selection process

For selection process, we determine the fitness function $z_i^{'}$ to evaluate the i^{th} chromosome $i = 1,2,\ldots,N$. Let z_i be the value of the objective function in the Stage 1. Therefore, we have:

$$E(P_i) = \frac{(z_i^{'})^{10}}{\sum_{k=1}^{N}(z_k^{'})^{10}} \times 100, p_i = \sum_{k=1}^{i} E(P_i)$$

where

$$z_i^{'} = \frac{\sum_{i=1}^{N} z_i(x, c_{Ga}, c_G)}{z_i(x, c_{Ga}, c_G)}, \qquad z_i(x, c_{Ga}, c_G,) = \sum_{s=1}^{M_G} c_G \sum_{i \in G_s} \sum_{j=1}^{n} p_j x_{ij} + c_{Ga}\,(L-1)t$$

Then we use the spanning roulette wheel to prefer the chromosomes: randomly generate a number $p \in (0,100)$, if $p \in [p_{i-1}, p_i)$, then the chromosome P_i is selected. see, e.g., the researches N.Shahsavari pour, M.Esmaeili and R.Esmaeili (2011).This process can be described as the following algorithm:

- Selection algorithm:

Step 1. Let $j = 1$, repeat Step 2 until N.

Step 2. Randomly generate a number $p \in (0,1)$; if $p \in [P_{i-1}, P_i)$, then chromosome S_i is selected and let $j = j + 1$, see, e.g., the researches see, e.g., the researches of Shasavari Pour et al. (2010).

Genetic algorithm

Step 1. Randomly initialize N chromosomes,

Step 1. Let $k = 1$, repeat Step 2 to Step 7 until $k = TC$ (until a given number times (TC)),

Step 2. Calculate the fitness of each chromosome according to the objective values,

Step 3. Select the chromosomes by spanning the roulette wheel,

Step 4. Perform crossover process and mutation process on the chromosomes,

Step 5. If $k = T$ report the best chromosome as the optimal solution,

Step 6. Arrange the chromosomes in decreasing order of processing times to form a sequencing priority list,

Step 7. Select 50% from the best chromosome and another 50% Randomly select from the remain chromosomes and let $k = k + 1$.

4. The numerical example

In this section, the efficiency of the proposed heuristic algorithm is showed by solving an example. In the example, let $n = 60, m = 100, k = 5$ and $T = 50, t = 9$, and $M_G = \frac{100}{5} = 20$, furthermore the problem's data, processing time of jobs and costs and are given in the Table 1 and Table 2. We take the stage 1, as an example to solve the numerical example.

Table 1
The processing time of jobs of the numerical example, p_j (job(j), processing time(p_j))

(1,2)	(2,17)	(3,21)	(4,9)	(5,4)	(6,7)	(7,13)	(8,17)	(9,16)	(10,22)
(11,3)	(12,10)	(13,12)	(14,9)	(15,14)	(16,24)	(17,11)	(18,2)	(19,6)	(20,8)
(21,15)	(22,28)	(23,11)	(24,13)	(25,4)	(26,33)	(27,27)	(28,29)	(29,30)	(30,22)
(31,3)	(32,8)	(33,36)	(34,1)	(35,2)	(36,18)	(37,11)	(38,6)	(39,7)	(40,26)
(41,2)	(42,5)	(43,3)	(44,12)	(45,9)	(46,14)	(47,12)	(48,1)	(49,31)	(50,27)
(51,7)	(52,5)	(53,9)	(54,11)	(55,3)	(56,25)	(57,2)	(58,5)	(59,3)	(60,7)

In this example, C_G and C_{Ga} are 10 and 5, respectively.

Table 2
One sample optimal solution

Group	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Jobs	1	14	9	8	12	22	26	6	2	5	23	13	3	24	10	33	-	-	-	-
	11	25	18	17	19	30	38	39	4	15	36	44	40	28	16	-	-	-	-	-
	21	32	27	20	29	-	42	49	7	37	47	46	-	-	-	-	-	-	-	-
	31	35	34	41	48	-	43	52	58	54	53	51	-	-	-	-	-	-	-	-
	50	56	59	45	55	-	57	-	-	60	-	-	-	-	-	-	-	-	-	-

Let the crossover probability is $p_{cross} = 0.93$ and the mutation probability is $p_{mut} = 0.35$. All the evolution parameters are obtained by the statistic and analyze of the experiment results of a numerical example with 100 machines and 60 jobs. This example has multi optimal schedule and the optimal value of the objective function is equal to 2600. It is well known that the evolution process and the absolute errors or the relative errors can mainly characterize the efficiency of the genetic algorithm. For the given example, first, we considered 200 generations, with the given evolution parameters $p_{cross} = 0.93$ and $p_{mut} = 0.35$ then the optimal solution is obtained at the 1000th generation. If we consider 300 generations then the optimal solution obtain the 500th generation.

5. Conclusion

In this paper, one important scheduling problem is studied. There have been many algorithms for this problem and its extending problems, see, e.g., the studies Chang, Chen and Fan, (2009) and Low, Hsu and Su (2010). For solving the given problem SMMA, we designed a heuristic genetic algorithm. And using this algorithm the optimal solution (optimal scheduling) the proposed problem is obtained.

By considering the number of jobs, machines, groups and type assignment each job to each group and machine, this problem can be extended to the scheduling problem which for solving it should be used a different genetic algorithm or another algorithm. Furthermore, in the real world possible all of the single-machine cost, (SMMA) aren't crisp. Some costs of problem are characterized by uncertain information such as fuzzy variables. So, for solving those problems we need new studies and researches.

References

Chen, W. J. (2006). Minimizing total flow time in the single-machine scheduling problem with periodic maintenance. *Journal of the Operational Research Society*. 57: 410–415.

Chen, W. J. (2009). Minimizing number of tardy jobs on a single machine subject to periodic maintenance, *Omega*, 37, 591–599.

Chang, P. C., Chen, S. H., & Fan, C. Y. (2009). A hybrid electromagnetism-like algorithm for single machine scheduling problem. *Expert Systems with Applications* 36, 1259–1267.

Hsu, C. J., Low., C. & Su, C. T. (2010). A single-machine scheduling problem with maintenance activities to minimize makespan. *Applied Mathematics and Computation,* 215, 3929-3935.

Low. C., Hsu, C. J., & Su, C. T. (2010). A modified particle swarm optimization algorithm for a single-machine scheduling problem with periodic maintenance. *Expert Systems with Applications* 37, 6429-6434.

Pinedo. M. (2002). *Scheduling, Theory, Algorithms, and Systems, Prentice-Hall*, New Jersey.

Sbihi, M., & Varnier, C. (2008). Single-machine scheduling with periodic and flexible periodic maintenance to minimize maximum tardiness. *Computers and industrial Engineering*, 55, 830–840.

Shasavari Pour, N., Esmaeili., M., & Esmaeili, R. (2011). Optimization of fuzzy multi-company workers assignment problem with penalty using genetic algorithm. *Journal on Computer Science and Engineering,* 3, 3148-3160.

Shasavari Pour, N., Modarres, M., Tavakkoli-Moghaddam, R., & Najafi, E. (2010). Optimizing a multi-objective time-cost-quality trade-off problem by a new hybrid genetic algorithm. *Word Applied Journal* 10(3), 335-363.