International Journal of Performability Engineering, Vol. 12, No. 2, March 2016, pp. 183-195.

Memetic Algorithm to Optimize Preventive Maintenance Schedule for a Multi-Component Machine

KARTIKEYA UPASANI[1]*, MIROOJIN BAKSHI[1], VIBHOR PANDHARE[2], and BHUPESH KUMAR LAD[2]

[1]*Discipline of Computer Science & Engineering, Indian Institute of Technology Indore, INDIA.*
[2]*Discipline of Mechanical Engineering, Indian Institute of Technology Indore, INDIA.*

(Received on November 26, 2015, Revised on February 08, 2016)

Abstract: Failure of a machine in a manufacturing industry leads to disruption of operations that severely impacts the enterprise in terms of revenue and customer satisfaction. The most prevalent strategy to mitigate the chance of machine failure is to perform preventive maintenance. Optimizing the preventive maintenance schedule is important to minimize operations cost and machine downtime. Further, for a multi-component machine, the knowledge of which components to perform preventive maintenance on can be crucial for such optimization. A stochastic simulation model can be used to evaluate all possible candidates and find the optimum preventive maintenance schedule. However, this involves infeasible computational time owing to the combinatorial nature of the problem. A Memetic Algorithm is proposed as a heuristic in the present work to address this challenge. Accuracy of obtained solutions and run-time is compared with brute-force search method and genetic algorithm for the same system. The Memetic Algorithm is found to yield better results as it can explore the search space more efficiently.

Keywords: *Maintenance scheduling, Multi-component machine, Memetic Algorithm, Genetic Algorithm.*

1. Introduction

A typical manufacturing industry consists of several machines that operate according to a decided job schedule for a shift. With usage, the condition of machine components degrades and if no Preventive Maintenance (PM) action is taken, the degradation process will ultimately result in a failure. This leads to a disruption of the pre-planned operations causing losses associated with machine repair or Corrective Maintenance (CM), like the lost production, delays in jobs, etc. Performing too less preventive maintenance can be ineffective in mitigating machine breakdowns. On the other hand, though excessive maintenance can eliminate the chances of a machine breakdown, the cost incurred and production lost for such maintenance schedules will be high, thereby affecting the throughput. According to [1], even a slight improvement in throughput could result in significant economic impact. Therefore proper planning of maintenance operations is important for an enterprise to operate in a cost-effective manner.

Although research on maintenance optimization was established years ago [2], the area of simulation-based optimization is becoming an emerging area [3]. The entities in a manufacturing system have complex interdependencies and interactions. [4] developed both analytical and simulation models for this problem and found that the analytical

*Corresponding author's email:kartikeya1994@gmail.com

model was complex and made unrealistic assumptions, whereas the simulation-based model was more flexible in handling stochastic nature of maintenance operations of an enterprise. [5] describes a method for scheduling maintenance operations by assessment and prediction of the level of degradation of machines and the impact of maintenance operations on production process. This involves evaluating effects of all possible maintenance schedules through discrete-event simulation that utilizes predicted probabilities of machine failures in the manufacturing system. [6] did probabilistic prediction of failure from Weibull analysis of failure data, and used Simulated Annealing as heuristic for finding the lowest cost preventive maintenance schedule. However, scarce amount of literature is available that considers a multi-component machine, which can be vital in improving the effectiveness of the maintenance scheduling. In the present work, a stochastic simulation model is used for the multi-component machine.

The multi-component machine model considered in this work allows specifying the number of components, failure distribution characteristics, repair distribution characteristics and labour requirements of each component, along with job processing cost, delay costs, and labour costs.

The objective function for such a stochastic simulation model is usually taken as the total cost or total downtime in a shift, and cannot be analytically defined. Moreover, the solution space for the optimal maintenance plan is vast due to the combinatorial nature of the problem. Hence a heuristic search method is needed to find a feasible solution in short time. While genetic algorithms have been successfully applied to many optimization problems, premature convergence is an inherent characteristic of such classical genetic algorithms [7]. This makes them incapable of searching numerous solutions of the problem domain. This paper proposes a memetic algorithm to search for solution to the problem of maintenance planning for a multi-component machine. Memetic algorithms have been used in several areas such as location optimization of health facility for diabetics [8], multi-compartment vehicle routing problem [9], and maintenance of railway infrastructure [10]. The performance of the memetic algorithm is found to be better as compared to other techniques like brute force search and Genetic Algorithm (GA). The heuristic function employed for local search makes premature (suboptimal) convergence less likely and enables more efficient searching of solution space, leading to better solutions in terms of cost and runtime. This is particularly useful for scenarios with large number of components and preventive maintenance opportunities due to which the solution space is vast.

The remainder of this paper is organized as follows: Section 2 describes the problem statement in depth. Section 3 proposes a solution to the problem and introduces different methodologies to attain this solution. Results of tests and comparisons of the different solution methodologies are presented in section 4. Section 5 concludes the paper and talks about future scope.

2. Problem Description

Consider a machine consisting of multiple assemblies, with each assembly comprising of multiple components. Let the total number of components in the machine be represented by *'n'*. Initially, a component may not be new and could have an accumulated age. Time

.

to failures of any component are considered to follow a two-parameter Weibull distribution with a given shape parameter (β) and scale parameter (η). A machine is said to breakdown when one or more of its constituent components fail. It is assumed that the components have only one failure mode, which brings the machine to breakdown state instantly and can be detected immediately. It is also assumed that the failure of a component is independent of other components.

Upon failure, Corrective Maintenance (CM) activity is performed on the machine. The total time for a CM activity for a component is given by the sum of actual time taken to repair (maintainability) and the time required to arrange for resources to perform repair (supportability), for that component. The uncertainties related to repair actions make it difficult to precisely predict the time required for CM activity. Hence, the time to repair (TTR) for CM is considered to be normally distributed with the mean (μ) and standard deviation (σ). μ and σ for a component can be easily obtained from its past data of TTRs. After a maintenance activity has been performed, age of the component is reduced by a factor known as Restoration Factor (RF) for that component. RF is used as a measure to model the degree of repair of a maintenance activity. A restoration factor of 1 signifies maximal or perfect repair. It means that the age of the machine component is restored by 100% to zero — 'as good as new condition' — after the maintenance activity. On the contrary, a restoration factor of 0 signifies minimal repair. This means that there is no effect of the maintenance activity on the machine component's age and thus the component stays in an 'as bad as old' condition. A restoration factor between 0 and 1 signify an imperfect repair with the value of RF signifying the extent to which the age will be repaired after maintenance activity. For example, for a component of age 1000 hours, a maintenance activity with the restoration factor of 0.8, will reduce the age to 200 hours. The RF of a CM activity for a component is taken as 1, which is the case of replacement of failed component with a new one. The parameters for CM for four components have been shown in Table 1, as considered in the assumed model.

Every component has specific labour requirements in terms of quantity and skill of the labour. Skilled, semi-skilled and unskilled labour are available for maintenance action, each incurring different costs, to carry out a maintenance activity. It is assumed that labour is always available for maintenance when it is required. Table 2 describes labour requirement for maintenance of each component. Table 3 describes the total available maintenance labour and the cost of maintenance labour per hour, as considered in the assumed model.

Preventive maintenance (PM) is a scheduled maintenance activity done as a proactive measure to mitigate machine failure and downtime losses. The total TTR for PM of a component, follows a normal distribution, similar to TTR for CM described above. As a PM activity represents imperfect repair, the RF for a PM activity is taken between 0 and 1. Further, if a maintenance activity involves more than one component, then it is performed as a series of maintenance activities (done one after the other) for each involved component. The cost incurred to carry out a maintenance activity consists of fixed costs (cost of spare parts, and other fixed costs) and cost of maintenance labour. Table 1 also shows the parameters for PM for four components, as considered in the assumed model.

Table 1: Model Parameters for Components

		Corrective Maintenance (CM)									Preventive Maintenance (PM)						
		Reliability		Maintainability		Supportability			Fixed Costs		Maintainability		Supportability			Fixed Costs	
COMP	AGE (hr.)	Time to Failure (hr.)		Time to Repair (hr.)		Waiting time for resources (hr.)		RF	Spare Parts (Rs.)	Other (Rs.)	Time to Repair (hr.)		Waiting time for resources (hr.)		RF	Spare Parts (Rs.)	Other (Rs.)
	X	η	β	μ	σ	μ	σ	α			μ	σ	μ	σ	α		
C1	420	2000	2.5	4	1	1	0.5	1	40000	0	4	1	0	0	0.4	0	8000
C2	545	2000	3	6	1	3	0.5	1	36000	0	6	1	0	0	0.5	0	12000
C3	630	3000	2.5	5	1	2	0.5	1	38000	0	5	1	0	0	0.6	0	12000
C4	817	3000	3	4	1	1	0.5	1	40000	0	4	1	0	0	0.7	0	8000

Table 2: Labour Requirement for Maintenance of Components as Considered in Model

COMP	Skilled		Semi-skilled		Unskilled	
	CM	PM	CM	PM	CM	PM
C1	0	0	1	1	1	1
C2	0	0	1	1	2	2
C3	0	0	2	2	1	1
C4	0	0	3	3	2	2

Table 3: Maintenance Labour Parameters as Considered in Model

	Skilled	Semi-Skilled	Unskilled
Cost (Rs./Hr.)	800	500	300
Available	2	4	8

The machine receives a job schedule for each shift, which is the description of jobs that it must complete in that shift. Each job has a due date and a penalty cost per hour, for every hour of delay after the due date. A delay in the job can be caused by machine breakdown or preventive maintenance activity. A PM activity can be scheduled after a job has ended and before the next job has begun, i.e. in between two jobs, but cannot interrupt a job. The instances in a schedule when a PM activity can be scheduled are referred to as *PM opportunities*, and the number of PM opportunities in a schedule is denoted by *'m'*. *m* is equal to the number of jobs in the schedule. PM activity must be incorporated into the schedule such that the total cost incurred in the shift is minimum. This involves answering two questions (i.e. decision variables) - when (at which opportunities) to perform PM and on what components to perform PM for an opportunity?

$$Maintenance\ cost\ for\ a\ component = (Total\ Fixed\ Cost) + (Time\ to\ repair) * (Total\ Labour\ Cost/\ Hour) \quad (1)$$

$$Penalty\ cost\ for\ a\ job = Penalty\ cost\ per\ hour * Delay\ in\ hours \quad (2)$$

$$Total\ Cost\ Incurred\ in\ a\ shift = \sum_{i=1}^{n}\left(\sum_{j=1}^{m} C_{ij} * PM\ Cost\ for\ i^{th} component + CM\ Cost\ for\ i^{th} component\right) + Total\ Penalty\ Cost\ for\ all\ jobs \quad (3)$$

$$where\ C_{ij} = \begin{cases} 1\ if\ PM\ is\ performed\ on\ i^{th} component\ at\ j^{th} opportunity \\ 0\ if\ no\ PM\ is\ performed\ on\ i^{th} component\ at\ j^{th} opportunity \end{cases}$$

The cost models for all the various costs involved are represented by equations (1)-(3). The models have been adopted from [11]. The total cost for a shift is given by the sum of delay costs of all jobs, costs incurred due to all PM activities and costs incurred due to all CM activities upon simulating the schedule, is given by equation (3). *'i'* denotes the i^{th} component, *'j'* denotes the j^{th} PM opportunity and C_{ij} denotes if PM is performed on the i^{th} component in the j^{th} opportunity, it is either '1' (Yes) or '0' (No).

To perform cost based optimization, a method is needed to evaluate the expected cost of operation of the machine for any given shift schedule. Discrete Event Simulation is used for this purpose, and the total expected cost incurred for a schedule is obtained by averaging the costs obtained by simulating that schedule multiple times. Probability distributions of each component (as defined in Table 1) are used for predicting events during simulation. For example, the Weibull distribution is used to obtain the conditional probability of failure for a component with some initial age, at a time instant of the shift.

The number of times a schedule is simulated to obtain the average cost incurred is referred to as simulation count, denoted by *'k'*. For *m* PM opportunities in the shift, the combination of components out of total *n* components for which PM activity has to be performed needs to be decided. For each opportunity, there is a choice of either performing or not performing PM activity for a component. Hence, there are 2^{m*n} possible ways of incorporating PM in a job schedule for a shift, including the case when no PM activity is scheduled for any component at any opportunity at all. The aim is to identify the PM schedule that minimizes the total cost incurred in that shift. But it becomes computationally infeasible to evaluate all possible PM schedules. For example, a schedule with 6 PM opportunities and a machine with 10 components, the number of total possible PM incorporated schedules is of the order of 10^{18}. The complexity further increases with increase in number of jobs and the number of machine components. Thus an effective method is needed that can produce optimal solutions in feasible time. In this work, a Memetic Algorithm (MA) is presented for the same.

3. Problem Interpretation and Solution

Out of 2^{m*n} possible solutions in the solution domain, the solution S_i must be found such that it has the least average cost. Hence the objective function can be defined as:

$$\min_{\forall i \in [1, 2^{m*n}]} S_i$$

Where S_i is the average of total costs incurred when the i^{th} possible PM incorporated schedule is simulated *k* times.

The solution (PM incorporated schedule) is encoded in the form of a binary string of length *m*n*, as at every PM opportunity there is an option of either performing or not performing a PM for each component. For example, a possible solution for a machine with *n*=5 components and *m*=3 opportunities can be represented by a binary string as:

1 0 1 1 0 0 1 0 1 0 0 1 1 1 1

m=1 m=2 m=3

Each group of bits represents a PM opportunity in the schedule, with the rightmost group being the first opportunity. The value at the i^{th} place, from right hand side, in each PM opportunity tells whether to perform PM (C_{ij}=1) or not to perform PM (C_{ij}=0) for the i^{th} component at that opportunity. In this string, PM activity has to be performed for component no. 2, 3 and 5 at the third opportunity; for component no. 2 and 4 at the second opportunity and for component no 1, 2, 3 and 4 at the first opportunity. The approaches used to solve the problem are presented below.

3.1 Brute-force Search (BF)

All possible preventive maintenance schedules are enumerated and the average cost of implementing the schedule is evaluated for each by simulation. The solution with the least cost is the most optimal solution for that shift, and is the solution to the problem. This can be used as a benchmark to assess optimality of solutions produced by other algorithms.

3.2 Memetic Algorithm (MA)

A memetic algorithm is a hybrid version of conventional genetic algorithm, which involves an extra step of optimization by local search. This helps the memetic algorithm in escaping local minima and achieve faster convergence as compared conventional genetic algorithms. Figure 1 presents the flow of MA. The various operations involved are described below.

Chromosome: The component combinations undergoing PM across opportunities for a schedule in a shift are represented by a binary string as described above. This binary string forms the *chromosome* for the MA.

Initialize population: A fixed number (population size) of chromosomes are randomly taken from the solution domain as the *initial population*. Population size is taken as 2**m***n*.

Fitness evaluation: Evaluation of the fitness function of the chromosomes in the population is done. For the problem at hand, the fitness function is the average cost of simulating the schedule represented by the chromosome for the shift, *k* times.

Selection of parents: Roulette wheel selection is used to select individuals for reproduction. Number of individuals selected is *m*n*, i.e. half of the population size.

Crossover: Pairs from the pool of selected parents are picked randomly and offspring are generated using Single Point Crossover. For each pair, a crossover point is randomly selected and two offspring are generated by swapping the chromosome data beyond the crossover point.

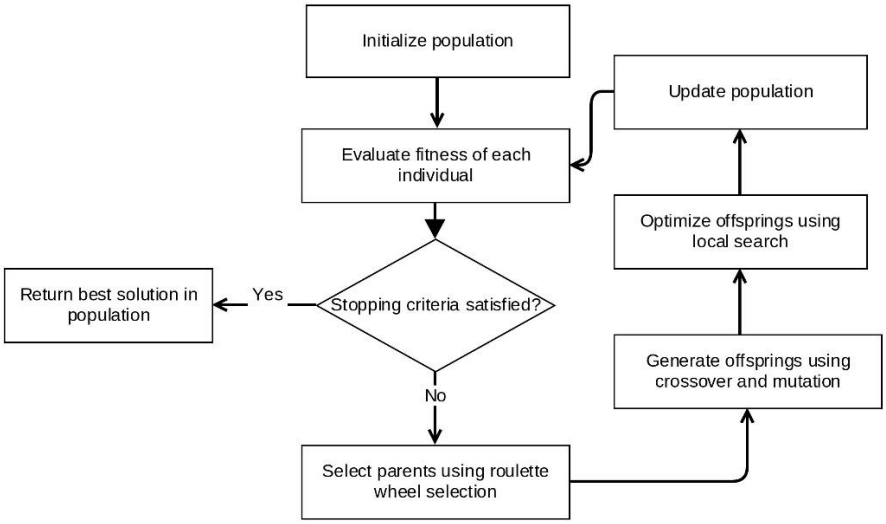

Figure 1: Flowchart of the Implemented Memetic Algorithm.

Mutation: Mutation is carried out on the off-springs using the bit flip mutation operator. Two bits in the chromosome are randomly chosen and flipped. For this experiment, the mutation probability is 0.4.

Local search optimization: Local search optimization is performed on each individual of the population. A neighbour is defined as any solution that can be reached from the current solution by flipping any two bits in its chromosome. Stochastic hill climbing approach is used to iterate through the neighbours of the current solution, and a neighbour is chosen to replace the current solution only if it results in an improvement. The heuristic function used to evaluate a solution for local search optimization is described in detail below.

Termination criterion: The search for optimal solution is terminated after 100 generations or if the fittest solution (one with the least cost) is the same for 25 consecutive generations. After meeting the termination criterion the algorithm returns the fittest solution.

Local Search Optimization and Heuristic Function: In local search, an attempt to improve the fitness of the population in each generation is made. The value of heuristic function of a chromosome is compared with the value of heuristic function of its neighbouring chromosomes. If a neighbour has a better (higher) heuristic function value, then the first chromosome is replaced with the neighbour.

Let *i* be the component number, *j* be the PM opportunity. P_i is the predicted probability of failure for the i^{th} component in a shift. Pi is estimated by computing the Time to Failure (TTF) multiple times for a component (that follows Weibull distribution characterized by η, β and age) and checking how many times the TTF lies in between shift duration. $C_{ij}$is the bit of the chromosome representing the decision of performing PM on the i^{th} component in the j^{th} opportunity, if C_{ij}=1 then PM is performed and C_{ij}=0 then PM is not performed. The heuristic function is modelled in such a way that a favourable decision regarding performing PM for a component for an opportunity is rewarded (the value of the function is increased by 1), while an unfavourable decision is penalized (the value of the heuristic function is decreased by 1). For example, if a component whose predicted probability of failure is high and is not scheduled for PM, then we subtract 1. The heuristic function *H( )*, for a chromosome is given by:

$$H\ (chromosome) = \sum_{i=1}^{n}\sum_{j=1}^{m} V_{ij}$$

Where,

$$V_{ij} = \begin{cases} 1, & P_i > 0.5\ and\ C_{ij} = 1 \\ -1, & P_i > 0.5\ and\ C_{ij} = 0 \\ 1, & P_i < 0.5\ and\ C_{ij} = 0 \\ -1, & P_i < 0.5\ and\ C_{ij} = 1 \end{cases}$$

Here, 0.5 is taken as threshold value for considering the probability of failure as 'high'. This can be set depending on the risk of machine failure one wishes to incorporate in the planning. The evaluation of *H( )* for chromosome starts from first opportunity (*j*=1) and if PM is performed on a component *i*, then the age of that component is updated.

Consider the chromosome, 100 101, as an example. In this, the machine has three components and the schedule has two opportunities. The probabilities of failure are calculated for each component for the first opportunity, say P_1=0.8, P_2=0.4 and P_3=0.6. Hence V_{11}=1, V_{21}=1, V_{31}=-1. As C_{11}=1, PM is set to be performed on the component 1, and its age is updated. This is repeated for the next opportunity. A higher value of *H( )* is indicative of a more favourable schedule.

3.3 Genetic Algorithm (GA)

The local search step of the memetic algorithm described above is turned off to get an equivalent genetic algorithm. Other genetic parameters and operators are kept untouched.

4. Results and Discussion

The following setup was used for execution and testing of all the three algorithms:
Intel(R) Core i7-4790 CPU @ 3.60GHz

4GB Memory

Windows 7 Professional 64-bit

Java v8.

The number of components (*n*) and the number of PM opportunities (*m*) in a shift schedule determine the size of the problem. The value of objective function and run-time

of the memetic algorithm is compared with brute-force search and conventional genetic algorithm for various *n* and *m* values.

Table 4: Results for Comparison of MA and BF for Small Problem Size

m	*n*	Brute Force Algorithm (BF)			Memetic Algorithm (MA)		
		Cost of solution	Time (sec)	Solution	Cost of solution	Time (sec)	Solution
3	4	36041	59.84	0000 0001 0110	36505.1	7.17	0000 0000 0110
3	5	52029.8	863.66	00000 00000 00110	52818.4	10.49	00000 00000 00110
4	4	37420	1336.51	0001 0000 0000 0110	35633.4	14.75	0000 0000 0000 0110

Table 4 presents the results of comparison of BF and MA for small problem sizes. As evident from the table, the optimal value of the objective function obtained by MA is nearly same as that of the BF. More importantly, the run-time of BF increases exponentially whereas that of MA increases linearly with increment in problem size. This validates the scalability of MA in terms of computation time, while maintaining accuracy of results.

In the third test case, the objective function value obtained from MA is less than that of BF. This may appear misleading, since the BF is expected to produce the most optimal solution. However, it is due to the stochastic nature of the objective function that such a variation exists. Increasing simulation count (k) will help in getting more accurate average costs, but will increase algorithm run-time.

Table 5 presents comparison of objective function and run-time of MA and GA. Here comparison with BF is not possible as its run-time is too large. It can be seen from Table 5 that the objective function value obtained from MA is lesser than that obtained from GA. Moreover, the run-time of MA increases gradually with problem size, which makes it an excellent choice for use with machines with larger number of components.

For a better understanding of relative performance of MA and GA, both algorithms are run for 100 generations without termination criteria, and with the same initial population. The objective function value obtained in each generation is plotted against the generation number in Figure 2. It can be observed from the graph that MA progresses faster towards the optimum than GA, and is able to reach a better solution in lesser number of generations. This can be attributed to the fact that GA gets stuck at local minima (as can be seen at generations 18, 41 and 57). On the other hand, the MA produces fitter offspring due to the additional step of local optimization and is able to escape such local minima.

Table 5: Results for Comparison of MA and GA for Larger Problem Sizes

Problem Size		Memetic Algorithm		Genetic Algorithm	
m	**n**	**Avg. Cost**	**Runtime (sec)**	**Avg. Cost**	**Runtime (sec)**
4	8	187259.1	75.240269	185521	58.615076
4	12	287711.6	128.307684	299929.6	99.990081
6	8	186558.8	166.37679	187806.5	141.205
6	12	291019.9	226.251081	307550.7	198.829984
10	20	515106.7	312.11912	535031.5	226.578425
10	30	755727.2	593.140146	787057.6	485.571496
20	20	507489.4	350.969291	539720	223.705157
20	30	762359.8	713.553458	808806.3	509.91421

To test the performance of algorithms with variation in operation parameters, experiments were conducted by varying the penalty (delay) cost per hour for jobs. The results have been shown in Table 6. With increasing penalty cost per hour, the algorithm makes an effort to reduce total cost by minimizing the downtime. This is done by reducing PM activity (i.e. lesser 1 bits in the solution schedule as observed in Table 6).

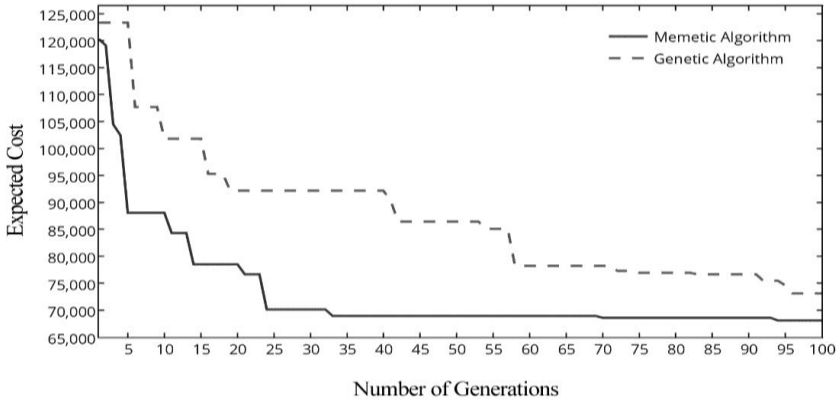

Figure 2: Comparison of Generational Performance of MA and GA.

The reduction in PM activity with reduction in penalty cost is a result of the corrective repair being considered to be "as good as new" (restoration factor for CM activity is 1), whereas the restoration factor of PM activity is taken to be 0.4 to 0.7. This localizes the solutions towards one particular area of the search space (i.e. zero), hence GA is able to produce solutions as good as that of the MA. Similar behavior is observed

when total downtime in a shift is used as the objective function to be minimized. Hence for problems with trivial solutions, the MA performs just as good (or bad) as the GA.

The Memetic Algorithm is advantageous for problems whose optimal solutions are spread across the solution space. In such cases, the algorithm's local search step enables it to not only converge to an optimum, but also explore the entire solution space efficiently. Hence for scenarios with older machines, the MA would be more useful. This can be evident from observations made in Table 7. In Table 7, the results of optimization with corrective maintenance considered to be "as bad as old" (restoration factor for CM activity is taken as 0) are presented. This ensures that the solutions obtained to minimize downtime involve scheduling of more PM activity, and hence are not trivial. The performance of the MA and GA is compared for both objective functions viz. cost and downtime. The MA is observed to produce better solutions than the GA.

Table 6: Results of Variation in Penalty Cost per Hour.

Penalty Cost per hour	MA Cost	GA Cost	MA Solution	GA Solution
0	272498.2	287735.1	000000000001 000100000011 001110001001 111000000000	001110000001 000000000100 000100001001 101000000000
200	287711.6	299929.6	000000000000 000100000001 001111001011 010000000000	000100000000 000100000011 001000000000 000000000000
500	318148.2	315938.2	000100000000 000000000001 000110000010 001000000000	000000000000 000110000010 011000000000 000000000000
1000	352527.5	351723.0	000000000000 001100000001 000000000000 000100000000	001000000000 000100000000 001000000000 000000000000
2000	405345.1	400718.9	000000000000 000100000000 001000000000 000000000000	000000000000 000100000000 001000000000 000000000000

Table 7: Results of Experiments with Corrective Maintenance Considered to be "as Bad as Old".

		Total Downtime Optimization		Cost Optimization	
m	**n**	**MA Downtime**	**GA Downtime**	**MA Total Cost**	**GA Total Cost**
4	6	375.57	409.87	1257331.45	1370628.30
4	8	449.73	476.28	1502867.65	1500885.10
4	12	821.97	847.53	2948262.10	3178298.30
6	8	427.12	484.63	1397716.55	1597201.45
6	12	834.45	867.82	2914139.00	2948005.25
6	20	925.41	936.31	2958533.40	3151284.05
10	20	904.73	922.20	2848462.00	3035134.00
20	8	562.69	567.94	1709998.00	1722076.90
30	12	1077.37	1142.69	3113586.05	3450808.60

5. Conclusions

A Memetic Algorithm is devised in this paper to obtain a solution to the problem of preventive maintenance planning for a multi-component machine. The problem is combinatorial in nature and the devised algorithm makes it possible to obtain the solution in feasible time. On comparison with brute-force search, the Memetic Algorithm is found to yield nearly the same solution in considerably lesser time. When compared with a conventional genetic algorithm, it is found that the Memetic Algorithm converges faster to an optimum solution, and is able to explore the solution space more efficiently. Moreover, the run-time of the Memetic Algorithm increases gradually with problem size, which makes it an excellent choice for use with models of machines with large number of components. The use of Memetic Algorithm for optimizing PM scheduling for a multi-machine scenario provides the immediate future scope of the current work.

References

[1]. Yao, X., E. Fernandez - Gaucherand, M. C. Fu, and S. I. Marcus. *Optimal Preventive Maintenance Scheduling in Semiconductor Manufacturing*. IEEE Transactions on Semiconductor Manufacturing, 2004; 17(3): 345–356.

[2]. Dekker, R. *Applications of maintenance optimization models: A Review and Analysis.* Reliability Engineering and System Safety, 1996; 52(3): 229–240.

[3]. Garg, A. and S. Deshmukh. *Maintenance Management: Literature Review and Directions.* Journal of Quality in Maintenance Engineering, 2006; 12(3): 205–238.

[4]. Rezg, N., A. Chelbi and X. Xie. *Modeling and Optimizing a Joint Inventory Control and Preventive Maintenance Strategy for a Randomly Failing Production Unit: Analytical and Simulation Approaches*. International Journal of Computer Integrated Manufacturing, 2005; 18(2-3): 225–235.

[5]. Yang, Z., D. Djurdjanovic and J. Ni. *Maintenance Scheduling In Manufacturing Systems*

Based on Predicted Machine Degradation. Journal of Intelligent Manufacturing, 2008; 19(1): 87–98.

[6]. Dhawalikar, M. N., P. K. Srividhya, V. Mariappan, A. Surlakar, M. J. Sakhardande and N. P. Bhale. *Hybrid Tool for Optimal Preventive Maintenance Schedule for Deteriorating Systems.* International Journal of Performability Engineering, 2015; 11(3): 283–291.

[7]. Garg, P. *A Comparison between Memetic Algorithm and Genetic Algorithm for the Cryptanalysis of Simplified Data Encryption Standard Algorithm*. International Journal of Network Security & Its Applications, 2009; 1(1): 34–42.

[8]. Alegre, J. F., Álvarez, A., Casado, S. and Pacheco, J. A. *Use of Memetic Algorithms to Solve a Stochastic Location Model: Health resources for diabetics in some provinces of Castilla-Leon*. XIII Jornadas de ASEPUMA. 2005.

[9]. Mendoza, J. E., B. Castanier, C. Gueret, A. L Medaglia and N. Velasco. *A Memetic Algorithm for the Multi-Compartment Vehicle Routing Problem with Stochastic Demands*. Computers and Operations Research, 2010; 37: 1886–1898.

[10]. Budai, G., R. Dekkerand and U. Kaymak. *Genetic and Memetic Algorithms for Scheduling Railway Maintenance Activities*. Econometric Institute Report EI, 2009; 30.

[11]. Lad, B.K. and M. S. Kulkarni. *Reliability and Maintenance Based Design of Machine Tools*. International Journal of Performability Engineering, 2013; 9(3): 321–332.

[12]. Alrabghi, A. and A. Tiwari. *State of the Art in Simulation-Based Optimisation for Maintenance Systems*. Computers & Industrial Engineering, 2015; 82: 167–182.

[13]. Tambe, P.P., Mohite, S. and Kulkarni, M.S. *Optimisation of Opportunistic Maintenance of a Multi-Comp Onent System Considering the Effect of Failures on Quality and Production Schedule: A case study*. International Journal of Advanced Manufacturing Technology, 2013; 69: 1743–1756.

Kartikeya Upasaniis currently pursuing his Bachelor's degree in Computer Science and Engineering at Indian Institute of Technology Indore, India. His research interests include Artificial Intelligence, Multi-agent Systems, and Internet of Things.

Miroojin Bakshiis currently pursuing his Bachelor's degree in Computer Science and Engineering at Indian Institute of Technology Indore, India. His research interests include Reinforcement Learning, Evolutionary Algorithms and Internet of Things.

Vibhor Pandhareis currently pursuing his Master's degree in Production and Industrial Engineering at Indian Institute of Technology Indore, India. His research interests include Industrial Internet of Things and Distributed Operations Planning.

Bhupesh Kumar Lad is an Assistant Professor in School of Engineering (Discipline of Mechanical Engineering) at the Indian Institute of Technology Indore, India. He received his Ph.D. from Indian Institute of Technology Delhi, India. His research interests include reliability, prognostics, computer aided manufacturing process planning.