Research Journal of Applied Sciences, Engineering and Technology 4(9): 1205-1210, 2012
ISSN: 2040-7467

Submitted: January 14, 2012 Accepted: February 02, 2012 Published: May 01, 2012

Minimizing Earliness and Tardiness Penalties in a Single Machine Scheduling Against Common Due Date using Genetic Algorithm

Milad Yousefi and Rosnah Mohd Yusuff
Department of Mechanical and Manufacturing Engineering, University Putra Malaysia, Serdang, Malaysia

Abstract: Scheduling problems involving both earliness and tardiness costs became more important in recent years. This kind of problems include both earliness and tardiness penalties and tries to reduce them. In this study a case study from real world is investigated. Since the machine is bottle neck in the production line and all jobs have a common restrictive due date and different earliness and tardiness penalties, the problem is NP-hard. In this study a discrete Genetic Algorithm (GA) is successfully implemented to find an optimum scheduling for the production line. The results show the effectiveness of the proposed algorithm in finding near optimal solution.

Key words: Earliness/tardiness, genetic algorithm, optimization, scheduling, single machine

INTRODUCTION

In the seventies, a new manufacturing philosophy named Just in Time (JIT) became world-wide, being the automobiles manufacturer Toyota one of its precursors (Womack, *et al*., 1990). JIT consists in the elimination of all waste and the continuous improvement of productivity. (Arnold, 1998). A scheduling activity has an important responsibility in the improvement of the system performance when executed according to the JIT philosophy. Based on the premise of elimination of waste, the scheduling must balance the activities' execution trying to complete them near their due date, not before neither later.

In a production system both earliness and tardiness have certain penalties for manufacturer. In both cases costs are incurred: Early jobs cause holding costs by tying up capital while the effects of tardy jobs are dissatisfied customers and, in the long run, the loss of goodwill and reputation. Note that quantifying the incurred costs exactly is difficult, as most of them are not directly linked to cash flows. However, taking these kinds of costs into account is important in view of opportunity cost arguments (Biskup and Feldmann, 2001; Feldmann and Biskup, 2003).

Single-machine scheduling is the process of allocating a group of jobs to a single machine or resource. The tasks are organized so that one or many performance measures may be optimized. There are many real-world situations involving a single machine used for completing various tasks. Environments involving multiple machines can sometimes be considered as a single-machine problem by considering the bottleneck machine. Studying the single-machine problem theoretically is the building block for understanding more complicated scheduling problems with multiple machines (Al-Turki *et al*., 2001). In single machine scheduling There are n jobs available at time zero, which have to be processed on a single machine. Each of these jobs needs exactly one operation (Kanet, 1981; Biskup and Feldmann, 2001).

In different industries different machine scheduling systems are applied. Such as single machine scheduling, parallel machine scheduling and flexible machine scheduling; this study is about single machine scheduling and optimization of it.

Focusing on single machine scheduling is significant because understanding single machine problems could help us to understand complex problems well. For example multiple-machine scheduling problems are as a rule computationally intractable and a better understanding of them and their inherent structure may be gained by analyzing single-machine problems. Furthermore, in many real life situations it is one machine that causes a bottleneck of the whole production environment, so that production planning should be orientated towards this single machine (Feldmann and Biskup, 2003).

In the 1970s some papers have been published on scheduling problems that include both earliness and tardiness penalties. The earliest of these was by Sidney (1977). It showed that the problem of minimizing maximum job penalty (early or tardy), where all jobs have

Corresponding Author: Milad Yousefi, Department of Mechanical and Manufacturing Engineering, University Putra Malaysia, Serdang, Malaysia

the same early and tardy cost functions and idle time could be inserted, was polynomially bounded. A comprehensive survey on the common due date assignment and scheduling problems can be found in Gordon, Proth *et al*., (2002). One of the pathfinders studying in common due date problems was done by Kanet (1981). A brilliant review on this kind of problems can be found in Baker and Scudder, (1990).

Generally speaking, there are two classes of common due-date problems which have proven to be NP-hard, namely if a restrictive common due date is given or if different jobs incur different Penalties Biskup and Feldmann, (2001). A common due date is called unrestrictive as long as the optimal schedule S* (with f (S*)) can be realized. Obviously, an (externally) given common due date, for which $d \geq \sum_{i=1}^{n} Pi$ holds, is unrestrictive. Furthermore, the due date is called unrestrictive if it is a decision variable. For $\alpha_i = \beta_i = 1$ and d being unrestrictive problem is polynomially solvable in O (n log n) time by a matching algorithm; see Kanet (1981). Even with $\alpha \neq \beta$ where $\alpha_i = \alpha$ and $\beta_i = \beta$ for all jobs i = 1, n the problem remains polynomially solvable, see (Panwalkar *et al*., 1982). The unrestrictive common due-date problem becomes NP-hard if different penalties for the jobs are considered. For the case that the earliness and tardiness penalties are equal, that is $\alpha_i = \beta_i$ for i = 1,2 …, n, it is possible to construct a dynamic optimization algorithm. It's running time O $(n\sum_{i=1}^{n} Pi)$ is pseudopolynomial as it depends on the sum of processing times; see (Hall and Posner, 1991). In past few decades several metaheuristic algorithms applied to tackle this problem. For example; see (Sidney, 1977; Sundararaghavan and Ahmed, 1984; Bagchi *et al*., 1986; Fry *et al*., 1987; Al-Turki *et al*., 2001; Celso *et al*., 2005; Khorani *et al*., 2011). The variety of usage algorithm is high. Some studies applied Beam Search; (Peng and Thomas, 1989; Valente, 2009; Valente, 2010; Valente, 2008; Rakrouki *et al*., 2012). Abdul-Razaq and Potts developed a branch and bound algorithm for weighted tardiness and earliness problem, but do not allow idle time (Abdul-Razaq and Potts, 1988). This study applies genetic algorithm to optimize a real world problem.

GENETIC ALGORITHM

The genetic algorithm (GA) is an optimization and search technique based on the principles of genetics and natural selection. A GA lets a population made up of many individuals to evolve under specified selection rules to a state that maximizes the "fitness" (i.e., maximizes the benefit function). This method was developed by John Holland (1975) during the course of the 1960s and 1970s and finally one of his students, David Goldberg made it popular. David Goldberg in 1980s could solve a difficult problem involving the control of gas-pipeline transmission for his thesis (Goldberg, 1989). Holland's original work was summarized in his book. He was the first to try to develop a theoretical basis for GAs through his schema theorem. The work of Jong (1975) showed the usefulness of the GA for function optimization and made the first concerted effort to find optimized GA parameters (Jong, 1975). Goldberg has probably gave the most important contributions to the GA with his successful applications and excellent book in 1989. Since then, many versions of evolutionary programming have been tested with different degrees of success and failure.

Some of the advantages of a GA are:

- Optimizes with continuous and discrete variables
- This study provides a discrete GA
- Doesn't require derivative information
- Searches from a wide sampling of the cost surface at the same time
- Deals with a large number of variables
- Is well suited for parallel computers
- Optimizes variables with extremely complicated cost surfaces (they skip a local optima)
- Gives a list of optimum solutions, not just a single one
- The optimization may done with the encoded variables
- GA can work with numerically generated data analytical functions or experimental data

These advantages are fascinating and produce excellent results when traditional optimization approaches fail miserably.

Figure 1 shows a basic model of a genetic algorithm, one of the main techniques in artificial evolution. From an initial population (population of parents), couples of parents are selected to reproduce. Cross-over and mutation are possible to give children. Until the evolution is stopped, children are selected to become parents and so on. This basic model can be modified to match the requirements of the problem to solve.

Proposed genetic algorithm: In this research, the chromosome or individual that stands for a solution has two main components: the sequence itself and the idle times inserted at the beginning of the schedule. For example: [6] [4, 1, 2, 3] indicates that the processing order is: jobs 4, 1, 2 and 3, with first job starting at time 6. The genetic operators used were:

- **Mutation:** Each individual has a probability P_{MP} to undergo sequence mutation. The new solution is equal to its parent concerning sequence and idle

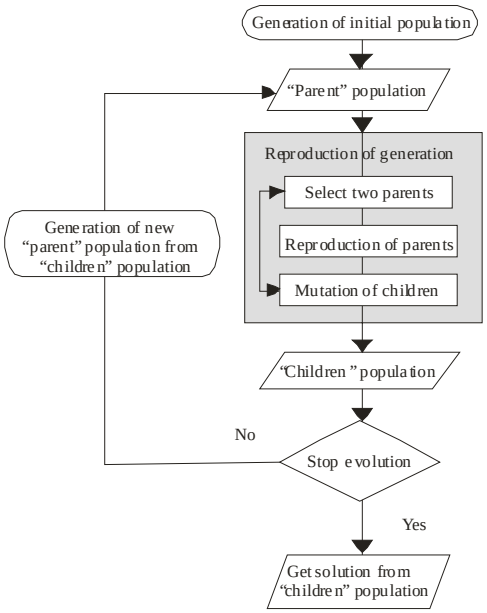

Fig. 1: Basic genetic algorithm

times, except for a pair of jobs randomly interchanged. One job necessarily belongs to the set of jobs completed on the due date or earlier, while the other job belongs to the set of jobs completed after due date.

- **Crossover:** Each individual (named "parent 1") has a probability PCR to undergo crossover choosing any other individual in the population ("parent 2"). There are different methods of crossover available in literature in this study we applied two points crossover. The new individual generated has chromosome parts of each parent and idle time equal to "parent 1". For example, take crossover points 4 and 7 and the individuals below:

  [3] [2, 1, 3, 7, 9, 6, 10, 4, 5, 8] Parent 1
  [4] [5, 2, 1, 7, 10, 8, 3, 4, 6, 9] Parent 2

  The procedure would generate the following sequence:

  [3] [2, 1, 3, 7, 10, 8, 3, 4, 5, 8]

  Since there are jobs that appear twice, it is necessary to replace them by other jobs. The procedure checks the first and third parts and looks for jobs that also appear in the second part; in this example, job 8. This job is replaced by other job that has never appeared (job 6), following the same order as in parent 1. In this example the method produces the following offspring:

  [3] [2, 1, 3, 7, 10, 6, 3, 4, 5, 8] Offspring

A similar procedure to generate a valid offspring is presented by Kozan and Preston (1999) and Celso *et al*., (2005).

- **Initialization:** In the first step of GA many individual solutions are randomly generated to form an initial population. The population initial generation depends on the nature of the problem, but typically contains several hundreds of possible solutions. Traditionally, the initial population is produced randomly, it allows the entire range of possible solutions (the search space). Sometimes, the solutions may be "seeded" in areas where optimal solutions are likely to be found.
- **Selection:** During each successive generation, a proportion of the existing population is selected to create a new generation. Individual solutions are selected through a fitness-based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as the latter process may be very time-consuming.
- **Reproduction:** The next step is to breed a second generation population of solutions from those selected through genetic

  For each new solution to be created, a pair of "parent" solutions is selected for generating from the pool selected previously. By breeding a "child" solution using the above methods of crossover and mutation, a new solution is produced which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new generation of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more similar to nature of biology, some research suggests more than two "parents" are better to be used to reproduce a good quality child. Eiben *et al*., 1994; Ting, (2005)

  These processes eventually result in the next generation population of offspring that is different from the initial generation. Generally speaking the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for generating, along with a small proportion of less fit solutions, for reasons already mentioned above.

  Crossover and Mutation are the most famous genetic operators but it is possible to use other operators such as regrouping, colonization-extinction, or migration in genetic algorithms.
- **Termination:** This generational process is repeated until a termination condition has been reached. Common terminating conditions are:
- A solution is found that satisfies minimum criteria

- Fixed number of generations reached
- Allocated budget (computation time/money) reached
- The highest level solution's fitness is reaching or has reached a plateau such that successive iterations no longer produce better results
- Manual inspection
- Combinations of the above

Simple generational genetic algorithm procedure:

- Choose the initial population of individuals
- Evaluate the fitness of each individual in that population
- Repeat on this generation until termination (time limit, sufficient fitness achieved, etc.):
- Select the best-fit individuals for reproduction
- Generating new individuals through crossover and mutation operations to give birth to offspring
- Evaluate the individual fitness of new individuals
- Replace least-fit population with new individuals

RESULTS

This study provides genetic algorithm for a real world case with 16 jobs on a single machine with total processing time of 83 units $\sum_{i=1}^{16} pi = 83$ and different earliness and tardiness penalties for each job. The assumptions of this study are:

- Preemption of jobs is not allowed
- It is not necessary the first job start at time zero. It could start later if it helps to minimize total costs
- This study considers no setup time for machine to do different jobs

Table 1 shows our collected data from case it is included number of jobs, processing time, earliness penalty and tardiness penalty for each job.

The objective is to jointly minimize the sum of earliness and tardiness penalties, where α and β stand for earliness cost and tardiness cost per unit of time respectively.

$$f(s) = \sum_{i=1}^{n} \alpha i \; Ei_{+} \sum_{i=1}^{n} \beta i \; Ti$$

To tackle this problem and find an optimum scheduling, genetic algorithm as a popular algorithm was used. To find an optimum sequence in first step the chromosomes were created. In this study chromosomes have two parts. First part depicts for the idle time in the chromosomes were created. In this study chromosomes have two parts. First part depicts for the idle time in the beginning of process. The first part of chromosome has only one number. The other part of chromosome has 16 gens each of which indicates one job.

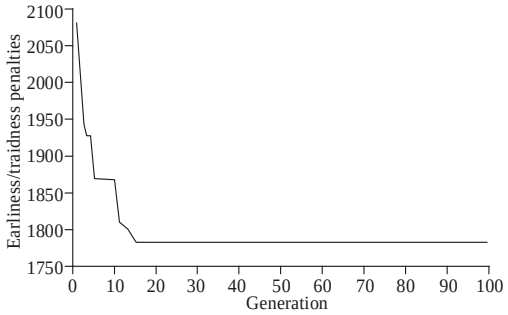

Fig. 2: The process of finding the best solution by genetic algorithm

Table 1: Collected data from real world case

Jobs	Processing time	Earliness penalty	Tardiness penalty
1	5	2	14
2	1	8	7
3	7	4	8
4	6	9	9
5	4	5	7
6	8	5	9
7	5	7	5
8	2	4	14
9	7	2	14
10	1	8	7
11	10	4	8
12	6	9	9
13	3	4	3
14	5	4	5
15	7	6	7
16	6	8	9
Total	83	89	135

To implement genetic algorithm in this case the initial population is 200 and the number of generation is 200. Both of them have been selected based on literature. The ratio of mutation in genetic algorithm of this study is 0.1.

In this study for programming of algorithm MATLAB software is used. Figure 2 shows process of the best solution proposed by genetic algorithm.

As it is demonstrated in Fig. 2, between initial generation and 18th generation sum of earliness/tardiness penalties drastically reduces and after that the earliness/tardiness penalties remains constant. After 50 executions of MATLAB program the optimum solution of genetic algorithm is:

[0] [9, 1, 6, 12, 10 , 8, 2, 5, 4, 16, 3, 7, 14, 13, 15, 11]

This chromosome is the best chromosome with lowest amount of total earliness and tardiness penalties. The first gen [0] means the first job should start at time zero where the next jobs after that are 9 and 1 respectively. In this case, job 2 will finish exactly on due date (30) and it doesn't have any earliness or tardiness penalties. The total amount of earliness and tardiness penalties for this optimum solution is 1780 units.

CONCLUSION

This research intended to contribute to the study of the single-machine scheduling problem with a common due date and earliness and tardiness penalties in a real world case. We presented heuristic methods that exploit the properties of this problem, specially the insertion of an initial idle time. A genetic algorithm applies to tackle this problem and the results show that applying heuristic algorithm reduces needed time for scheduling. It also decreases earliness and tardiness penalties.

The proposed GA presented in this study can be applied to the other types of machine scheduling problems such as parallel machine scheduling, flow shop scheduling and flexible assembly line as well. And also other algorithms can be applied for the first time in this field.

ACKNOWLEDGMENT

We are grateful to Kenseisha (M) Sdn Bhd that allows us to collect data from there. Their cooperation has greatly improved the presentation of the study.

REFERENCES

Al-Turki, U., C. Fedjki and A. Andijani., 2001. Tabu search for a class of single-machine scheduling problems. Comput. Oper. Res., 28(12): 1223-1230.

Arnold, J.R.T., 1998. Introduction to Materials Management. Upper Saddle River, Prentice Hall, N.J.

Bagchi, U., R. S. Sullivan and Y.L. Chang, 1986. Minimizing mean absolute deviation of completion times about a common due date. Nav. Res. Log. Quart., 22(3): 585-592.

Baker, K.R. and G.D. Scudder, 1990. Sequencing with earliness and tardiness penalties: A review. Operat. Res., 38: 22-36.

Biskup, D. and M. Feldmann, 2001. Benchmarks for scheduling on a single machine against restrictive and unrestrictive common due dates. Comput. Amp Operat. Res., 28(8): 787-801.

Celso, H.M., A. Hino, D.P. and A.B. Mendes, 2005. Minimizing earliness and tardiness penalties in a single-machine problem with a common due date. Europ. J. Operat. Res., 160: 190-201.

Eiben, A. E. P.E., Raue and Z. Ruttkay, 1994. Genetic Algorithms with Multi-parent Recombination. PPSN III: Proceedings of the International Conference on Evolutionary Computation. The 3th Conference on Parallel Problem Solving from Nature, pp: 78-–87. ISBN: 3-540-58484-6.

Feldmann, M. and D. Biskup, 2003. Single-machine scheduling for minimizing earliness and tardiness penalties by meta-heuristic approaches. Comput. Ind. Eng., 44(2): 307-323.

Fry, T. D., R. D. Armstrong, and J.H. Blackstone. 1987. Minimizing weighted absolute deviation in single machine scheduling. lIE Trans., 19(4): 445-450.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, MA, pp:

Hall, N., G.W. Kubiak and S.P. Sethi, 1991.Earliness-tardiness scheduling problems, II: Deviation of completion times about a restrictive common due date. Operat. Res., 39: 847-856.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Jong De, K.A., 1975. Analysis of the behavior of a class of genetic adaptive systems. Ph.D. Thesis, University of Michigan, Ann Arbor.

Khorani, V.N. Forouzideh, and A.M. Nasrabadi, 2011. Artificial neural network weights optimization using ICA, GA, ICA-GA and R-ICA-GA: Comparing performances.

Kozan, E. and P. Preston, 1999. Genetic algorithms to schedule containers transfers at multimodal terminals. Int. Transact. Operat. Res., 6: 311-329.

Kanet, J.J., 1981. Minimizing the average deviation of job completion time about a common due date. Naval Res. Logistics Quart., 28: 643-651.

Panwalkar, S. S., M. L. Smith. 1982. Common due date assignment to minimize total penalty for the one machine scheduling problem. Oper. Res., 30: 391-399.

Peng, S.O and T. Morton 1989. The Single Machine Early/Tardy Problem. Manage. Sci. 35(2).

Rakrouki, M.A., T. Ladhari, and T.V T′kidindt, 2012. Coupling Genetic Local Search and Recovering Beam Search algorithms for minimizing the total completion time in the single machine scheduling problem subject to release dates. Comput. Oper. Res. 39(6): 1257-1264.

Sidney, J., 1977. Optimal single-machine scheduling with earliness and tardiness penalties. Operat. Res., 25(1): 62-69.

Sundararaghavan, P.S. and M.U. Ahmed 1984. Minimizing the sum of absolute lateness in single-machine and multimachine scheduling. Naval Research Logistics Quarterly 31(2): 325-333.

Ting, Chuan-Kang 2005. On the Mean Convergence Time of Multi-parent Genetic Algorithms Without Selection. Advances in Artificial Life, pp: 403-412. ISBN 978-3-540-28848-0.

Valente, J.M.S., 2008 Beam search heuristics for the single machine early/tardy scheduling problem with no machine idle time. Comput. Indus. Eng. 55(3): 663-675.

Valente, J.M.S., 2009. Beam search heuristics for the single machine scheduling problem with linear earliness and quadratic tardiness costs. Asia-Pacific J. Operat. Res., 26(3): 319-339.

Valente, J.M.S. 2010. Beam search heuristics for quadratic earliness and tardiness scheduling. J. Operati. Res. Soci., 61(4): 620-631.

Womack, J.P., D.T. Jones, and D. Ross, 1990. The Machine that Changed the World. N.Y. Rawson Associat., 22(3): 72-86.