An Evolutionary Solution to a Multi-objective Scheduling Problem

Sumeyye Samur , Serol Bulkan

***Abstract*— Multi-objective problems have been attractive for most researchers because of its diversity in different areas, reality coming from real life applications and insolvability in polynomial time. Therefore, many algorithms including heuristics and/or evolutionary ones were developed to solve such problems.**

In this research, we propose a genetic algorithm approach to solve a bicriteria scheduling problem in identical parallel machines. Based on different lambda values, we try to minimize the combination of makespan (C_{max}) and tardiness (T_{max}). The problems with those objective functions are proven to be NP-hard in the literature and this combination of the problem is not studied before for parallel machines, to the best of our knowledge. The proposed solution is fairly broad to adapt to other scheduling problems.

***Index Terms*— genetic algorithm, makespan, parallel machine scheduling, tardiness**

I. Introduction

Scheduling is one of the essential research areas which deal with the optimization of some objective functions. Those objective functions are mostly single criteria functions in which the problem aims to optimize only one dimension of the problem such as makespan (C_{max}), lateness (Lmax), tardiness (T_{max}), earliness (E_{max}) and etc. Those problems may also consider some constraints like due dates, release times, precedence and etc. However, several researchers have recognized the importance of considering multiple measures of performance as a better representation of today's decision making [1], since the real world problems do not always have only one single criterion. Instead, they have two or more criteria to be optimized. In this sense, decision-makers face more complicated problems having multi-objective functions to be optimized. Therefore, a compromise solution must be found in accordance with the preferences of the decision-maker.

Multi-objective scheduling was studied by many researchers. Torres, Enscore, and Barton [2] proposed a simulated annealing heuristic for the average flow time and the number of tardy jobs in parallel machines. Koksalan and Keha [3] considered two bicriteria scheduling problems on a single machine. Stein and Wein [4] considered the problem of minimizing the makespan and total weighted completion time at the same time. Gupta and Ho [5] studied on the problem of scheduling jobs on two parallel identical machines where an optimal schedule is defined as one that gives the smallest makespan among the set of schedules with optimal total flowtime is studied. Gupta and Ruiz-Torres [6] considered the problem of generating a set of efficient (non-dominated) schedules on identical parallel machines involving total flow-time and total number of tardy jobs. In addition, there is also a survey on multi-objective meta-heuristics [7] and two other surveys on multi-objective scheduling as well [8, 9].

In this paper, we consider parallel machine scheduling and try to minimize makespan and maximum tardiness simultaneously. One of the criteria can be viewed as the manufacturer's objective and the other one as the customer's; resulting with the problem of a real life scenario. For such simultaneous optimizations, there are two possible methods as Gupta defines in [6]. First, a single objective function can be constructed, a linear combination of the various criteria is formed then it is optimized. Second, all efficient, which is also called Pareto Optimal, schedules can be generated where an efficient schedule is one in which any improvement to the performance with respect to one of the criteria causes a disintegration with respect to one of the other criteria.

In this research, we preferred to apply the first way which is an optimization of the linear combination of two measurements which are makespan and tardiness. To do so; we implemented a genetic algorithm. In Section 2; we discuss the problem and propose our solution. In Section 3; we test our algorithm and comment on the obtained results and finally in Section 4; we make the conclusion and present the future work.

II. The Problem and the Proposed Solution

A. The problem

Makespan (maximum completion time) minimization is one of the most dealt objective functions in scheduling area. Tardiness is also another important criterion for most problems. Both problems are proven to be NP-hard. In this research, we try to optimize those two criteria simultaneously. In general; there are m identical machines to process n jobs. Each job j has a deterministic processing time p_j and due date d_j. All jobs are ready at time zero and there is no precedence relation between jobs. Additionally, preemption or cancellation of jobs is not allowed.

The objective is to generate the best possible schedule which minimizes the makespan and the tardiness. If we use

Manuscript received February 24, 2010.

S. Samur is a PhD student in Department of Industrial Engineering, Marmara University, Istanbul, Turkey. She received her BS degree from Computer Engineering in Fatih University and MS degree from Computer Engineering in Marmara University. Her research interests include Operations Research, Scheduling Theory and Game Theory. Her email address is <sksamur@gmail.com>; phone: +90-506-268-42-05; fax: +90-216-348-02-93.

S. Bulkan is an Assistant Professor in Department of Industrial Engineering, Marmara University, Istanbul, Turkey. He received his BS and MS degree from Management Engineering in Istanbul Technical University. He also received another MS from Operations Research in Florida Institute of Technology and PhD from industrial engineering in Cleveland State University. His research interests include Operations Research, Production planning and scheduling in manufacturing systems. His email address is <sbulkan@marmara.edu.tr>

ISBN: 978-988-18210-8-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

the 3-field notation introduced by Graham [10], we can denote this problem as $P_m|| \lambda C_{max}+(1- \lambda)T_{max}$. We use lambda ($\lambda$) to express the varying objectives of the decision-maker. We use the following notations in the paper,

- m: machine size;
- n: job size;
- C_j: the completion time for job j;
- d_j: the due date for job j;
- $T_j = \max(0; C_j-d_j)$: the tardiness of job j;
- $C_{max} = \max_j C_j$: the maximum completion time (makespan),
- $T_{max} = \max_j T_j$: the maximum tardiness.

B. The proposed solution

We developed a genetic algorithm (GA) to solve the problem mentioned above. Before execution, we determined the values of some important variables in GA as in the following:

- Population size: There are 100 members in each population. (Parent, Child, New Populations)
- Generation size: There are at most 1000 generations to be performed.
- Stopping criteria: The algorithms stop either after 1000 (generation size) generations or if the best (fittest) member in the population does not improve for the last 200 generations. Either condition can make the algorithm stop.
- Mutation probability: It is 0.10.
- The job size and machine size: We run the GA for different number of machines and jobs. For job size; we tried 50, 100, 150, and 200 and for machine size; we tried 5, 10, 15, and 20.

```
Genetic Algorithm ()
{
    Parent population, child population and new population are empty at the beginning
    Generate initial (parent) population
    While (termination criteria is not met)
    {
        While (child population size < predetermined population_size)
        {
            Perform Crossover on parent population
            Optionally perform Mutation on the new obtained child individuals
            Add these children to the child population if not exists already
        }
        Generate new population from parent and child populations.
        Equalize parent population to new population
    }
}
```

1) Parent Population Generation (Initial population)

The initial parent population includes randomly created job sequences. Each member of this population shows a sequence, in other words; the order of the jobs to be scheduled. There are n! possible sequences. However, it is impossible to test all possible sequences; therefore we only take 100 randomly generated members (job sequences).

And based on the sequence, we schedule the jobs onto the machines one by one. To do so, we select the earliest ready machine which has the minimum completion time at that time.

Therefore, after this scheduling process, each individual in the population has its own C_{max} and T_{max} values and the objective function is calculated based on the predetermined lambda value.

2) Crossover

Based on the Roulette Wheel Selection two parents are chosen and crossover is applied. In this way, the fitter the individual is, the more chance it has to be selected. In our GA, we used two-point crossover.

Each job in a sequence is indexed beginning from 1 to job size. Two points are selected randomly on a parent chromosome. To create child1, the jobs corresponding to the ones between the selected points are taken from parent1 without any change. And beginning after the second point, the jobs in the parent2 chromosome are checked one by one, and if not already exists, it is put on the next available position in child1. Otherwise, next job in parent2 chromosome is checked and this process continuous until child1 is fully obtained. A similar procedure applies for child2.

A crossover example can be found in Fig.1 (job size is assumed to be 10 to make the example clear)

3) Mutation

After performing crossover, 2 child sequences are obtained. Based on a predetermined mutation probability (which is 0.10 in this research) each child either mutates or not. To do so, first a random double value is selected between 0 and 1, if the value is less than the mutation probability, the mutation is done, otherwise it is not.

To perform the mutation again two random points in the sequence are selected and the jobs in those points change their positions with each other.

A mutation example can be found in Fig. 2 (mutation is applied to child2 from Fig.1)

4) Child Population Generation

After crossover and mutation operations, child population is updated. In other words, if the population does not include the newly created child then the child sequence is added to the child population. Otherwise, crossover and mutation steps continue to repeat until the size of the child population is equal to the predetermined population size (which is already determined as 100).

5) New Population Generation

In this step, to get the fittest members of the child and parent population and also not to leave the randomness, we applied the following procedure to generate the new population:

ISBN: 978-988-18210-8-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

25% of new population are taken from the best (fittest) members of parent population and added to the new population

25% of new population are taken from the best (fittest) members of children population and added to the new population (if not already exists)

25% of new population are randomly chosen from the worst 75% of parent population added to the new population (if not already exists)

25% of new population are randomly chosen from the worst 75% of children population added to the new population (if not already exists)

III. Experimental tests and the results

We implemented the genetic algorithm in Java 1.6. Datasets are randomly created for 50, 100, 150, 200 jobs and 5, 10, 15, 20 machines. Due dates are randomly selected integer values between the processing time of the job and an upper value as in the following:

d_j = Random (p_j, (Sum (p_j)/machine_size)*tightness).

In this way, we aim to make the due date of a job based on the processing time of the job itself and the total processing times of all jobs. We determined the due date tightness as 1.10 which means the due time is at most as bigger as the 10% of the average processing time. In this way, we guaranteed to make the due dates flexible since the dataset is created randomly.

We test the same dataset for different lambda values which are 0, 0.25, 0.50, 0.75, and 1 in order to define different combinations in bicriteria objective.

To test the effectiveness of the algorithm, we looked at the percentage improvement between the best objective value in the first generation and that of the last generation.

Since the dataset is created randomly, we run the algorithm 10 times (repetition size) on the same dataset and take the average of the improvements in order to eliminate the effects of randomness.

In summary; for each job-machine pair, we create a dataset and on this dataset we try 5 different lambdas, in other words we try to optimize 5 different bicriteria objectives. And for each lambda value we run the GA 10 times and calculate the improvement. At the end; we take the average of these 10 improvements. Finally, we obtain average improvement values for each job-machine size datasets and interpret those results and show in a graph to observe the improvements more properly.

The graphs showing the relation between the improvement and different lambda values for varying machine sizes when the job size is fixed can be analyzed in Fig.3.

The results in Fig.3 can be interpreted as follows:

Independent of job size, when lambda is equal to 0, i.e. the objective is purely T_{max}, improvement over initial population is very large. These improvements decrease when lambda value is increased, i.e. the objective is bicriterion.

The minimum improvement is obtained for lambda is equal to 1 that is the objective is purely C_{max}. The reason for this result comes from scheduling field. In scheduling, when the number of jobs is very large compared to the number of machines, the makespan value for a randomly generated schedule is not very far away than the optimum makespan.

The graphs showing the relation between the improvement and different lambda values for varying job sizes when the machine size is fixed can be analyzed in Fig.4.

The results in Fig. 4 can be interpreted as follows:

Independent of machine size, when lambda is equal to 0, i.e. the objective is purely T_{max}, improvement over initial population is very large. These improvements decrease when lambda value is increased, i.e. the objective is bicriterion.

The minimum improvement is obtained for lambda is equal to 1 that is the objective is purely C_{max}. The reason for this result comes from scheduling field. In scheduling, when the number of jobs is very large compared to the number of machines, the makespan value for a randomly generated schedule is not very far away than the optimum makespan.

In summary, the proposed GA improved initial objective values independent of machine and job sizes. The improvement is the largest for the objective function with a lambda value of 0 and the smallest with a lambda value of 1.

IV. Conclusion and Future Work

In this research, we studied parallel machine scheduling with a bicriteria objective function which we tried to minimize makespan and maximum tardiness simultaneously. We also added lambda to the objective function to express decision maker's priorities between single objectives. As a solution; we proposed a genetic algorithm and made experimental studies to test the effectiveness of the method. Since there is no similar studies or appropriate lower bound calculations to test the effectiveness, we analyzed the improvement obtained by GA. The proposed GA gave quite good results and the improvement was at least 50% and even 100% for most cases. The method is fairly general to be applied to some other bicriteria objective functions.

In the future, we plan to compare proposed GA results to other heuristic solutions for bicriteria optimization. Moreover, we also want to improve the method by developing a better crossover operator in order to obtain superior results.

References

[1] Nagar, J. Haddock, and S. Heragu, European Journal of Operational Research 81, 88-104, 1995
[2] Ruiz-Torres, E. E. Enscore, R. R. Barton, Computers ind. Engng VoI. 33, Nos 1-2, pp. 257-260, 1997
[3] M. Koksalan, A. B. Keha, European Journal of Operational Research 145, 543–556, 2003
[4] Stein, J.Wein, Oper. Res. Lett. 21, 115–122, 1997
[5] J. N. D. Gupta, J. C. Ho, Computers &Operations Research 28, 705-717, 2001
[6] J. N. D. Gupta, A. J. Ruiz-Torres, European Journal of Operational Research 167, 679–695, 2005
[7] Jones, S. K. Mirrazavi, M. Tamiz, European Journal of Operational Research 137, 1-9, 2002
[8] V. T'kindt, J.C. Billaut, Springer, Berlin, 2002.
[9] Hoogeveen, European Journal of Operational Research 167, 592-623, 2005
[10] R.L. Graham, SIAM Journal of Applied Mathematics 17, 416–429, 1969

ISBN: 978-988-18210-8-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Parent 1	3	1	2	5		9	10	8	7	4	6
Parent2	9	10	6	3		2	8	4	7	5	1
Crossover points					*					*	
Child 1	6	3	2	4		9	10	8	7	5	1
Child 2	1	5	9	10		2	8	4	7	6	3

Fig 1. Crossover sample between randomly selected parents based on roulette wheel selection

Child 2 before mutation	1	5	9	10	2	8	4	7	6	3
Mutation points							*			*
Child2 after mutation	1	5	9	10	2	8	3	7	6	4

Fig 2. Mutation sample

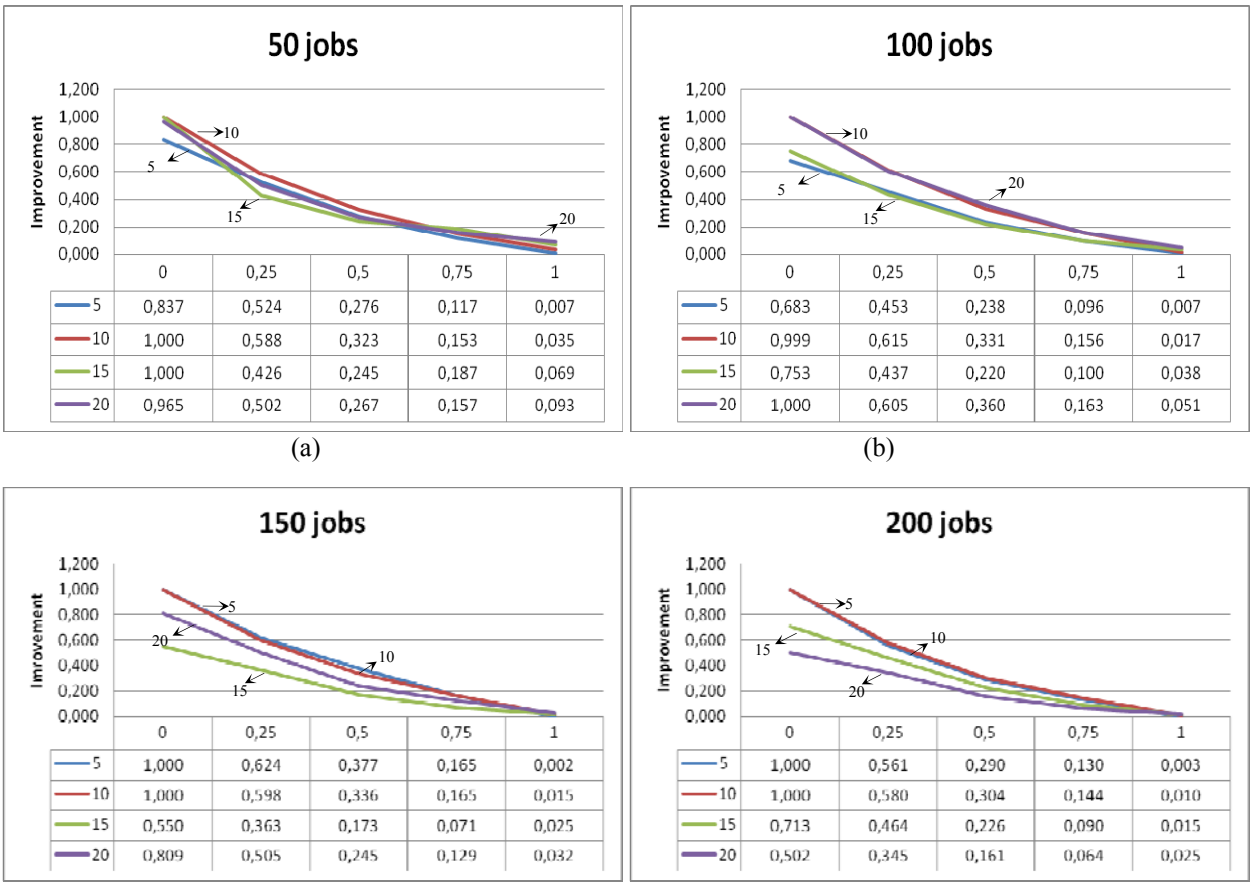
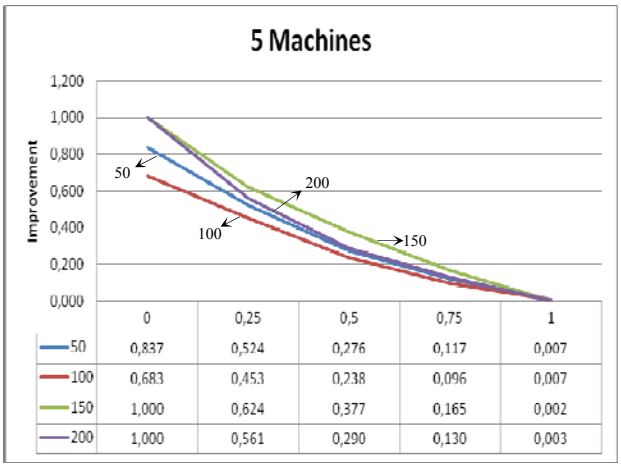


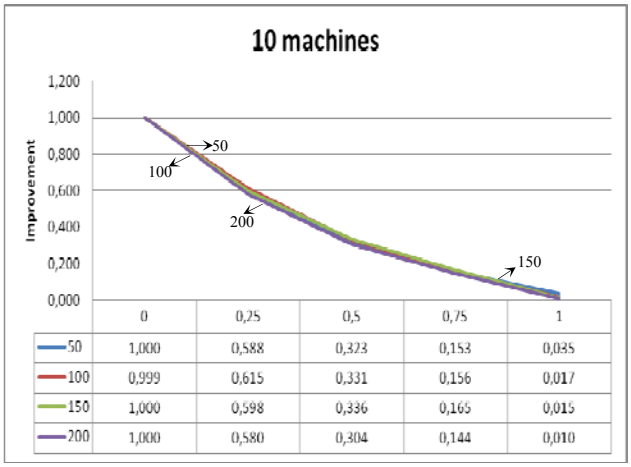
Fig 3. Improvement values based on number of jobs : (a) 50 jobs (b) 100 jobs (c) 150 jobs (d) 200 jobs

ISBN: 978-988-18210-8-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)



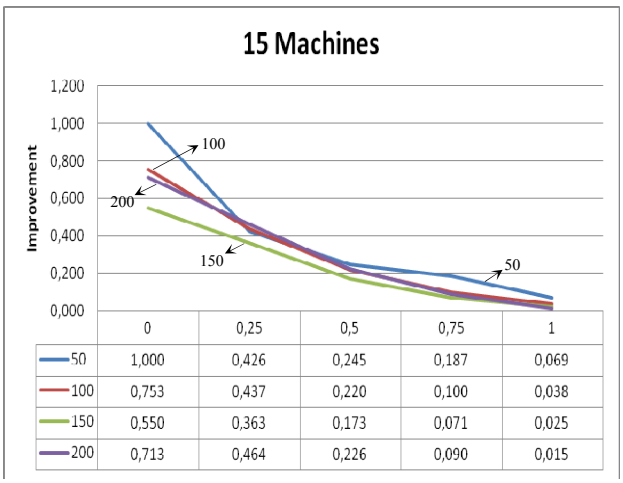
	0	0,25	0,5	0,75	1
50	0,837	0,524	0,276	0,117	0,007
100	0,683	0,453	0,238	0,096	0,007
150	1,000	0,624	0,377	0,165	0,002
200	1,000	0,561	0,290	0,130	0,003

(a)



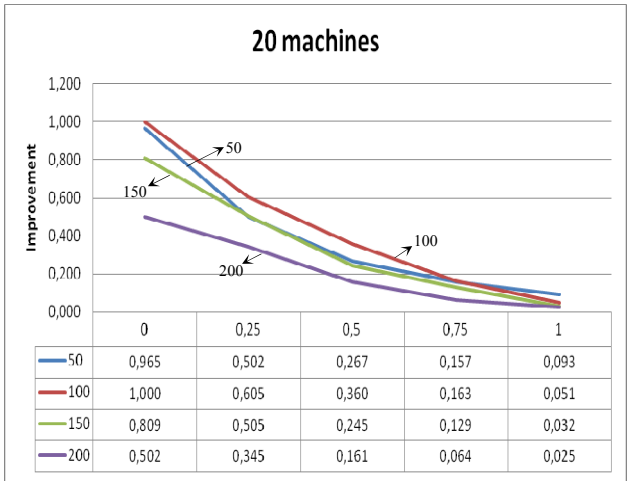
	0	0,25	0,5	0,75	1
50	1,000	0,588	0,323	0,153	0,035
100	0,999	0,615	0,331	0,156	0,017
150	1,000	0,598	0,336	0,165	0,015
200	1,000	0,580	0,304	0,144	0,010

(b)



	0	0,25	0,5	0,75	1
50	1,000	0,426	0,245	0,187	0,069
100	0,753	0,437	0,220	0,100	0,038
150	0,550	0,363	0,173	0,071	0,025
200	0,713	0,464	0,226	0,090	0,015

(c)



	0	0,25	0,5	0,75	1
50	0,965	0,502	0,267	0,157	0,093
100	1,000	0,605	0,360	0,163	0,051
150	0,809	0,505	0,245	0,129	0,032
200	0,502	0,345	0,161	0,064	0,025

(d)

Fig 4. Improvement values based on number of machines : (a) 5 machines (b) 10 machines (c) 15 machines (d) 20 machines

ISBN: 978-988-18210-8-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)