International Journal of Academic Research in Computer Engineering
Volume 1, Number 1, Pages: 16-24, September 2016
www.ijarce.org

IJARCE

On line Tasks Scheduling in Real Time Multiprocessor Systems Based on Multi-Objective Genetic Algorithm

Habib Varmaziar[1*], Ali Ghaffari[1], and Mahmoud Lotfi Anhar[2]

1. Department of Computer Engineering, Tabriz Branch, Islamic Azad University, Tabriz, Iran.
2. Department of Computer Engineering, Isfahan University, Isfahan, Iran.

Receive Date: 28 May 2016; Accepted Date: 14 July 2016, Published Date: 15 September 2016

*Corresponding Author: varmaziar.habib1390@gmail.com (H. Varmaziar)

Abstract
In real time systems, the accuracy of the system's behavior depends on the results of calculations and generated time. One of the biggest challenges of these systems is tasks scheduling according to their deadline. Scheduling issues have complex and sometimes inconsistent conditions, as a result, the system must correctly respond the intended answer within the specific deadline. Industrial control systems, medical, controlled rockets, satellites and so on are of this category. An optimum allocation of tasks in real time multiprocessor systems, often referred to non-deterministic polynomial hard problem. Intelligent methods are used to solve such problems. Data collection in this paper consists of three categories and it is composed of 100 tasks that is implemented in Java language in Eclipse environment. In this paper, the tasks scheduling of real time multiprocessor systems in real time using multi-purpose genetic algorithms for doing tasks and deadlines was studied that this algorithm is optimized to the uniform rate scheduling algorithms.

Keywords: Genetic Algorithm, Scheduling Tasks, Real time System, Deadline

1. Introduction

Implementation of tasks on substrate of processing target are placed among each other, this togetherness is called Scheduling. Scheduling must ensure that any task receives its required processing at the scheduling output. Scheduling algorithms can be generally divided into two categories. The first category is offline and the second category is Online. In offline scheduling algorithms, all decisions regarding the scheduling are performed before the implementation of the system. These algorithms choose the tasks for implementation with reference to a table of pre-determined schedule, On the other hand, in online scheduling, all Scheduling decisions are done without having any concrete information of what is supposed to enter the system. This scheduling algorithm choose the tasks for implementation by examining the properties of active tasks. Online scheduling algorithm can be more flexible than offline ones because they are able to schedule tasks that their properties are not known in advance. Online scheduling algorithms also can be divided into two categories, fixed priority and dynamic priority. In the fixed priority in scheduling algorithms, all job produced by a task have the same priority. The Rate Monotonic (RM) scheduling algorithm is the optimum among single-processor algorithms with constant priority [1, 2]. A dynamic-priority algorithm may assign different priorities to things manufactured by a task. Earliest Deadline First (EDF) scheduling algorithm is a well-known example of the dynamic priority algorithms [3]. EDF scheduling algorithm is the optimum among all schedule single processor algorithms. One of the important responsibilities of real time systems is task scheduling according to

their deadline and also the purpose of scheduling in real time systems is to assign processors to process over time.

The system is called a real time system that the accuracy of process is not only dependent on the logical accuracy, but also it is dependent on the timing in which it is implemented. In real time systems, the accuracy of output is as important as a logical accuracy. A real time system can be divided into three categories including hard real time, Soft Real time and Firm Real time [1, 4]. In the hard real time systems, the completion of an operation after its deadline is considered useless. Ultimately, this may cause a critical failure of the complete system. Aerospace, nuclear, power infrastructure and automotive industry are examples of hard real time applications. Soft real time systems have reduced their imitation on time constraints in proportion to hard real time systems. In such systems, however, it is still preferred to complete the task before the deadlines, but some of these restrictions will not lead to a completely useless or dangerous output. Cellphones and multimedia are applications that can take advantage of such systems. Unlike the hard real time systems, firm real time systems are resistant to the delay arising in operation. This means that the loss of a deadline only reduces the quality of a service. Firm real time systems can be used in Systems such as plane tickets servers that Concurrency is needed but at the same time they are able to tolerate delay in the second. In multiprocessor systems, the problem of the tasks timing are usually scheduled by several ways depending on how much migration is allowed by the system at runtime. [5, 6, 7]. The task is a so-called migration if the latter's job is done on other processors. Based on legal migration. Three different migration strategies is possible based on the rate of allowed migration [8, 9]. They are typically divided into three classes, no-migration, full migration, restricted migration. Each is described as follows, in no-migration class, no migration is allowed each task before the start of its performance, at run-time, allocated to a specific processor and all its tasks is run on the same processor [8, 9].

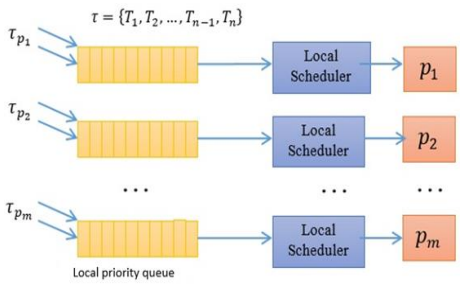

Figure 1. No-migration Schedule [9].

In Figure 1 shows a multi-part scheduling in which each processor maintains a specific priority areas related to their duties. In full migration schedule strategy, the job of a task can be migrated at any moment of their execution. All tasks are allowed to run on all processors in the system, but a task can run at any time on a single processor, meaning this means that task level parallelism is not possible.

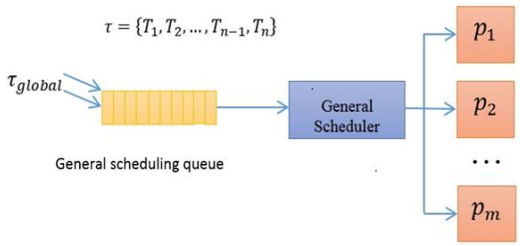

Figure 2. Scheduling with full migration [9].

In Figure 2 shows an example of schedule with full migration and at the strategies for restricted migration scheduling, tasks can only migrate within the confines of their migration. If the new job of a task is released, a high level Scheduler attributes it to a specific processor. After this assignment, the above-mentioned job should complete its implementation on the processor that is allocated to it.

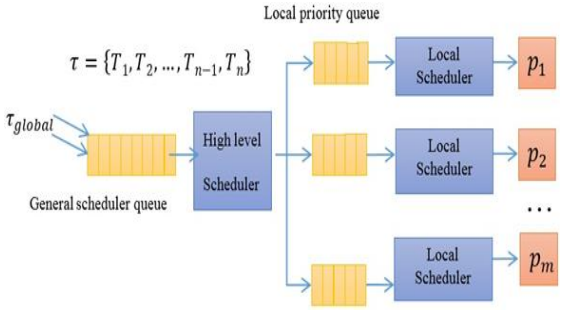

Figure 3. Scheduling with restricted migration [9].

Figure 3 shows a scheduling with restricted migration where there is a public priority queue with local priority queues to each processor.

In this paper, at the second part, we will discuss the scheduling algorithms and data sets for scheduling tasks in real time multiprocessor systems based on genetic algorithms. The third section examines the proposed method based on genetic algorithm. In the fourth section, simulation results and comparison of the proposed method with previous methods based on genetic algorithms and real time scheduling system will be discussed and finally in the fifth section, we will consider conclusions and future works.

2. Fundamental Research

2.1. EDF Scheduling Algorithm

Liu and Layland was presented EDF scheduling [10]. This dynamic priority scheduling algorithm is based on single-processor design and optimization across all dynamic priority scheduling algorithm. This algorithm schedules the task with the nearest deadline which includes four stages. The first step is to set the start time, end time, remaining time and the deadline for all tasks. Secondly, if the system is idle, the task will be added to the Scheduler and it is in stage four, otherwise goes to stage three. The third phase includes an update of the remaining task that is being processed. The fourth step if all tasks haven't been scheduled, it goes to stage two otherwise stops. In this algorithm, priority is assigned according to the deadline and tasks with minimal deadline have the highest priority [10, 11].

2.2. Dynamic- EDF (D-EDF) scheduling algorithm

D-EDF algorithm is a combination of the two algorithm with the ability to prioritize dynamic and static algorithm. The algorithm is simple and there isn't executive complexity. The implementation of this algorithm for high load of processor is suitable for real time multiprocessor model. Due to the availability of a number of processors, this algorithm is a truly dynamic performance. EDF algorithm is a combination of the two EDF scheduling algorithm and Deadline-monotonic (DM). DM algorithm is in the form of static schedule. In this case, each prioritized performance is relatively evaluated at the time of the stop. D-EDF algorithm in low loading manner uses EDF algorithm and the Performance of dynamic Prioritization with absolute time stoppage will be decided. In the top loading, DM algorithm will be used and the Static Prioritization performance with the stop of the relative time will be decided [12].

2.3. RM scheduling algorithm

In this algorithm, to every process in the operating system is attributed priority in proportion to the frequency of occurrence of the incident. For example, priority 50 is given the process is repeated every 20 milliseconds, and the priority 10 is given to the process is repeated every 100 milliseconds. This algorithm is a non-exclusive type. It can be proved that the algorithm is optimal. RM scheduling algorithms is of the algorithms with Static priority [10] and priority is assigned based on the time period. Tasks with the shortest period of time have highest priority.

2.4. Genetic Algorithm

Genetic algorithm was formed based on Darwin's theory of evolution. According to Darwinian evolution, only people with good genes can survive and produce new children and people with inferior genes are removed in the process. Multi-objective genetic algorithms including encoding, evaluate, synthesize, mutation, decoding [11], which is fully described in Section three. In general, GA is an oriented stochastic optimization technique that moves gradually toward the optimum point. About the properties on genetic algorithm compared to other optimization methods, it can be said that this algorithm is applicable to every problem without having any knowledge of the nature of the problem or any restrictions on the type of variables and its efficiency in finding the general optimum point has been proved. The capability of this method is in solving complex optimization problems in which either classical methods are not applicable or they are not reliable to find the general optimum. This algorithm is inspired by nature and is based on the principle of survival of the fittest [11, 13].

2.5. Dataset

In the beginning it is necessary to carry out the proposed method to introduce data set. To evaluate the proposed method, three different data sets created. Each of these data sets can be of much help to better assess their proposed methods. All of this data collection includes 100 tasks that each of the tasks can have randomly between 2 to 5 job, but jobs deadlines are different from each other, any job takes its own time. The first dataset, consisting of the tasks that the deadline is chosen with uniform probability from the interval [min, max] Min in this data set is equated with the minimum time required for sequential execution on a single-core processor and max is equated with the number of tasks multiplied by the maximum number of tasks that each task can have multiplied by the maximum duration of a job, multiplied by two. As It is known, although according to the time of execution of each job and the deadline for each task It may not be possible to implement all the tasks in a given time interval. However, it has been tried to select very generous deadline and it is expected to be evidence the lowest latency in this procedure and the maximum number of tasks to be carried on time. In the second data set, tasks have exactly the selected time period of time interval [min, max], the deadline has been selected with a uniform distribution, but the max is different from the first data set. In fact, in this data set the deadline is equal to the maximum number of tasks multiplied by the maximum number of job that each task can have

multiplied by the maximum duration of a job that is exactly half of the maximum time allowed for each task in the previous data set. The third data collection, method of choice deadline is as the two previous data sets, with the difference that the least possible amount for the deadline is also reduced. The amount in this series is the longest-sized jobs in the task.

3. Proposed Method

The aim of the proposed method is presenting a scheduling system to the jobs in multiprocessor systems. The proposed method is also considered a multiprocessor system features [14, 15] and offers a solution and it is also based on genetic algorithm. Problem-solving approach using genetic algorithm is shown in Figure 4.

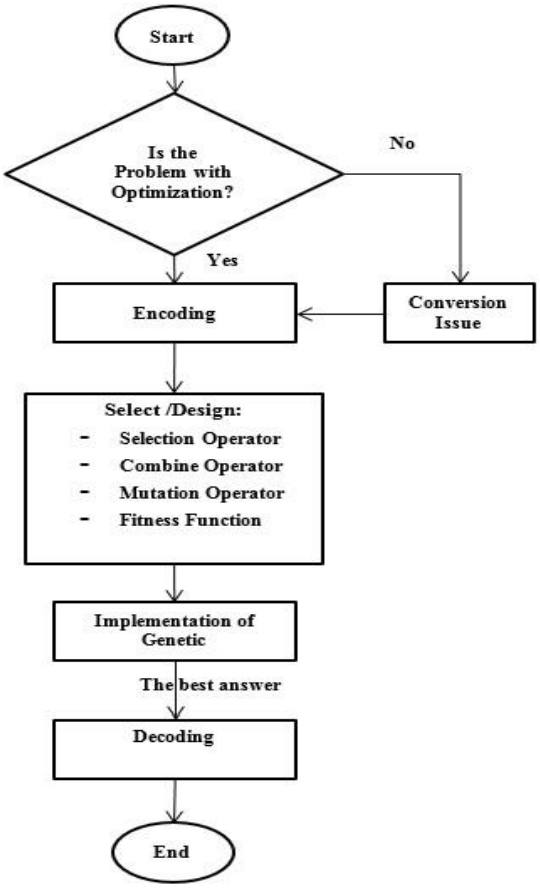

Figure 4. Problem solving using GA flowchart

Some definitions will be provided before scheduling algorithm description, j1 is the earliest start time, that is, the length of the longest path from an input to its job, it is displayed by erl(ji), and erl(ji) is formally defined as the formula (1) shows.

$$erl(j_i) = \begin{cases} 0 & if \neg\exists j_j |\ (j_j, j_i) \in E \\ \max\limits_{j_j \in p(j_i)} \{ erl(j_j) + e_{j_j} \} & 0.W \end{cases} \quad (1)$$

j_1 is the latest start time which is the longest time that it could start with Lst(j_i) displayed. Lst(j_i) is formally defined as it is shown in the formula (2).

$$Erl(j_i) = \begin{cases} d_i - e_{j_i} & if \neg\exists j_j |\ (j_j, j_i) \in E \\ \min\left\{ \min\limits_{j_j \in s(j_i)} \{ lst(j_j) - e_{j_j} \}, d_i - e_{j_i} \right\} & 0.W \end{cases} \quad (2)$$

Genetic Algorithm Based Scheduling [16, 17, 18, 19] is included the four phases of initialization, fitness assessment, review of the terminal condition and the end and which is shown in Table 1.

Table 1. Genetic Algorithm Based Scheduling.

1- Initialize the chromosomes population.

2- Assess the fitness of a population.

3- Check the condition of the terminal. If you reached the maximum number of repetitions do the following:

- [Parent selection]: A Chromosome is considered with minimum fitness twice and overcome on chromosome with maximum fitness to build Crossover.
- [Crossover]: Applying single point crossover with the possibility of combining possibility combination (PC) formed the new birth.
- [Mutation]: a new birth with the possibility of mutation (Pm).
- [Placement]: Embed new birth as new population.

4- End.

In this algorithm that it was explained in Table 1, each chromosome, indicating a possible schedule where each task is assigned to a processor. It should also be noted that in each chromosome tasks are allocated in an order from left to right to processors. Therefore, if two tasks are allocated to a processor, first, the task of left side runs on it then the right side. The proposed method is that depending on the length, each chromosome randomly selects a number of combinations and then moves the desired section to two parents alternatively. Of course, this shift moves each column entirely, it doesn't create any failures in the field. More precisely, if the length of the chromosome is equal to N and n is the number of chromosomes, then the number of N / n of combination is chosen randomly. An example of combination operator can be observed in Figure 5. It should be emphasized that in case of genes

transpositions, if their topology disrupted, parents are repeated as new child.

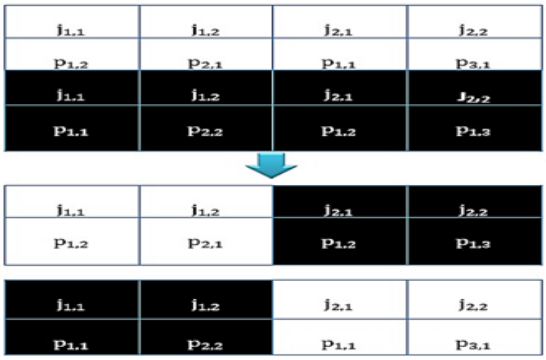

Figure 5. The Combination of the Two Chromosomes in Proposed Approach

According to the proposed approach provided in this section, the mutation operator acts on a gene but the chromosome information is taken into account. This operator acts in such a manner that a gene is chosen at random, then the section related to the processor of gene may be replaced by one of the processors in the chromosomes. An example of this operator can be seen in Figure 6. In this Figure, the third gene is mutated.

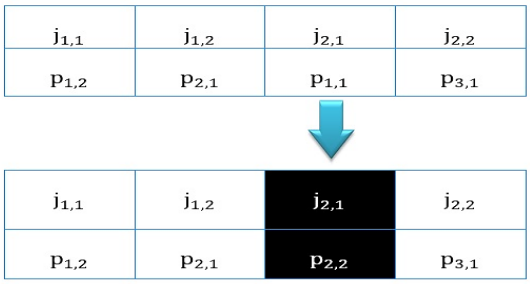

Figure 6. Proposed Mutation Operator

Competitive selection method is used for parental choice. Of course, this choice can be any of a variety of other methods for parental choice. The proposed approach doesn't create any restrictions on this operator. The initial population contains a certain number of chromosomes that are generated randomly and according to the tasks wait for the scheduling. As it was mentioned, each chromosome contains tasks related to scheduling awaiting for the tasks. But a processor is randomly assigned to each of these tasks. Algorithm related to the generation of the initial population is represented in Table 2.

Table 2. Initial Population Generation Algorithm.

1- Topological sorting : Applying topological sorting on the waiting tasks. 2- Production of the population: Average S times: Select M processors (cores) that will be associated with waiting tasks 3- End

The following will explain the evaluation function that somehow is the most important part of the proposed method. For more details, T1 and T2 tasks using their priority graphs is shown at Figure 7, assuming triple-core processors are now able to schedule, and two allowed chromosomes are shown.

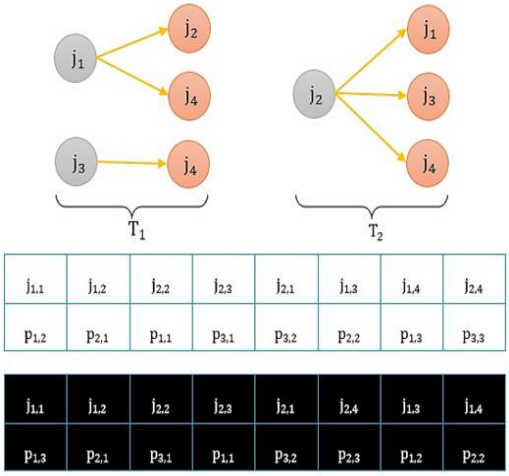

Figure 7. An Example of Two Chromosomes Based on Double Tasks.

The proposed fitness function F takes into account the multi-purpose at the same time, they have contradictory nature with each other and try to find the optimal point Pareto [20]. The fitness function is shown in equation (3).

$$F = \frac{A}{N_{PROC}} + \frac{B}{TARDINESS_{TOTAL}};\ A, B \in (0,1) \qquad (3)$$

In the formula (3), α, β are free parameters, they determine the impact of each component of fitness function. In this formula, first part ($\frac{\alpha}{N_{proc}}$) is trying to minimize the number of used CPU cores. Thereby energy consumption is also directly reduced. For example, two processors with four cores are self-employed. Compared to four processors used by each of its core they will lead to less energy consumption. The second part of the formula ($\frac{\beta}{Tardiness_{total}}$) tries to minimize the total delay in getting the tasks done, for example, if the start time of a task is after the allowed latest time for its implementation, the total running time as a delay is calculated in total.

4. Result and Experiment

In this section we discuss and compare the results between the proposed method and known methods of RM scheduling algorithm, EDF scheduling algorithm, D-EDF scheduling algorithm. In order to assess we will show them as charts. All three algorithms, despite their undeniable success,

simulate very simple concepts that of course makes extensive use of them in real time systems. These methods, along with the proposed method have been evaluated and tested so that the capability of the proposed method to be evaluated against the accepted algorithms, In Figure 9 we can see the number of tasks carried out for different population sizes. In this experiment, the number of processors is 4 and each processor has three cores.

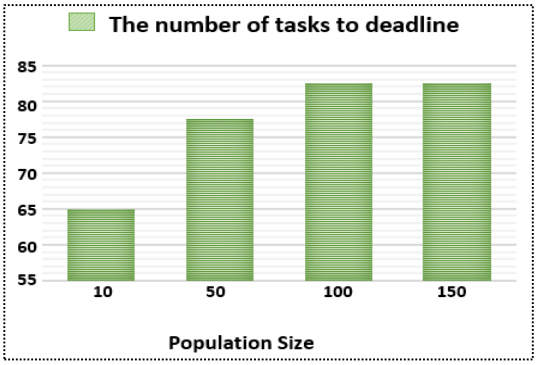

Figure 8. The Efficiency of the Proposed Approach for Different Population Sizes

Tests related to Figure 8, the Mutation probability and combination is considered 0.09 and 0.9 respectively. In Figure 10, the number of tasks carried out by RM, EDF and D-EDF methods for the first dataset is shown.

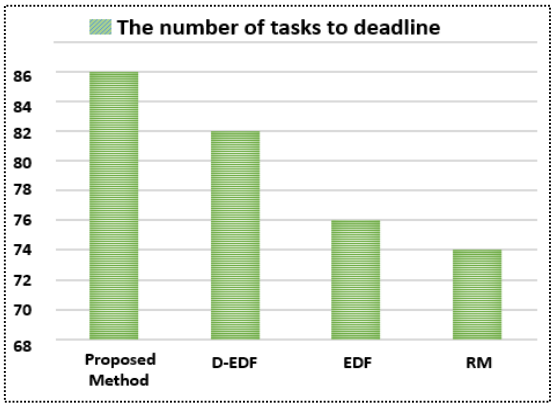

Figure 9. The Number of Tasks Carried out in the Deadline on the First Dataset

We found out from the comparison of Figures 8 and 9, if the deadline of performed task is relatively high the proposed method shows very high efficiency. It is important that the computational burden resulting from the implementation of a genetic algorithm with a population of 10 people is very low. Figure 10 shows the number of tasks to deadlines for different values of mutation probability. The results are for the first dataset; 150 Population size is considered.

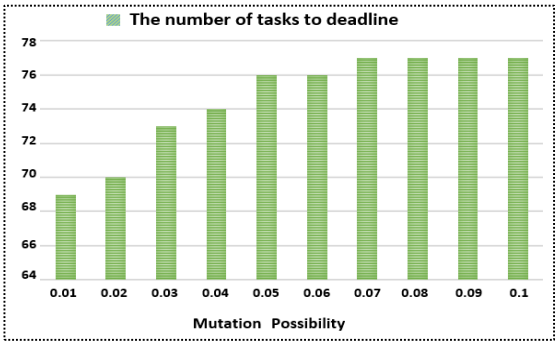

Figure 10. The effect of Mutation Probability Parameter on the Effectiveness of the Proposed Method

The values shown in Figure 10 is likely to 0.07 for the combined operator. In order to evaluate the effects of the probability values of crossover operator parameters, the number of completed tasks for different values of this parameter is shown in Figure 11.

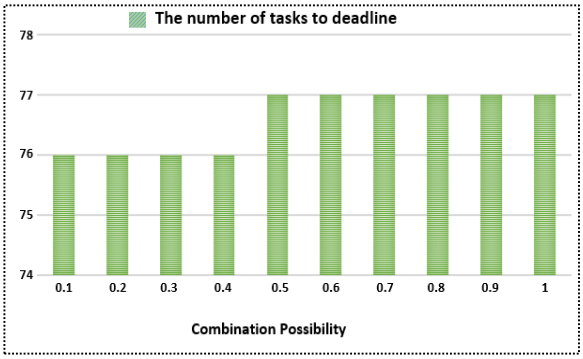

Figure 11. Effect of combination parameter probability on the effectiveness of the proposed method

In the experiment in Figure 11 that carried out on the first dataset, mutation probability is considered 0.1 The number of required generations to achieve the best results for different populations is shown in Figure 12 for the second dataset and for different population sizes.

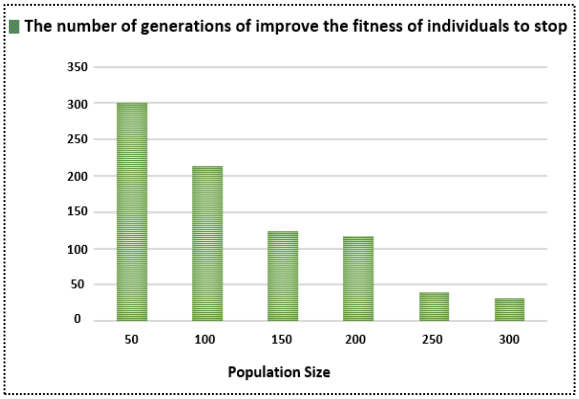

Figure 12. Evaluation of the Number of Required Generations to Stop the Genetic Algorithm

According to Figure 12, with the increase in population, the results haven't been performed. In

one experiment, for a population of 300, the number of completed tasks in a deadline was 47 cases. In Figure 13, the results of the implementation of the proposed method on the third data set is presented that strict deadlines are considered for the possibility of mutation operator for different values of parameters.

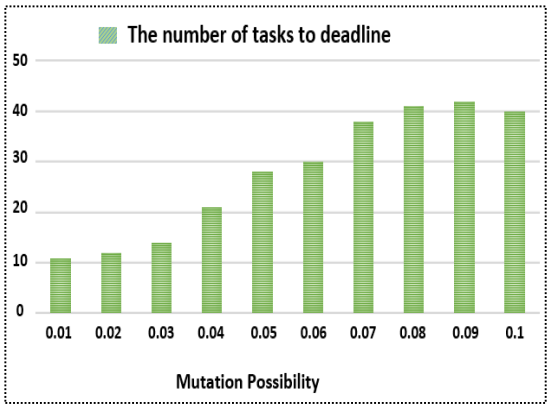

Figure 13. Effect of the Mutation Operator on the Proposed Approach for the Third Dataset

In Figure 13, the information for the value of possibility of combinations parameter is presented to be 0.6. The role of possibility of combinations on the functions of the proposed method when the deadline of jobs is strictly selected is shown in Figure 14. In the experiment, the probability of mutations was considered to be equivalent to 0.09.

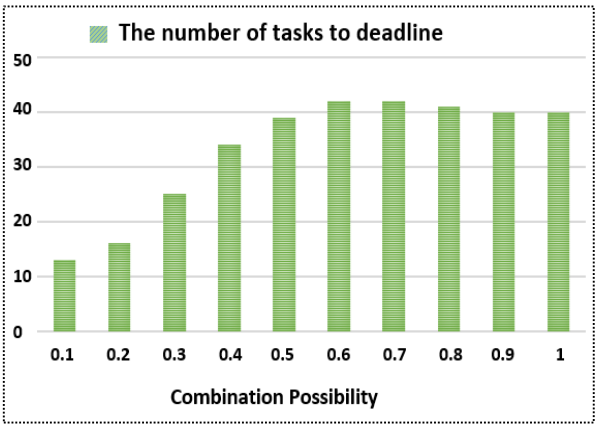

Figure 14. Effect of Combinations Operator on the Proposed Approach for the Third Dataset

According to mentioned figures, it is revealed that the proposed method affects under the effects of possibility of mutation-parameters and hybrid can have significant changes. Another experiments carried out in order to assess the performance of the proposed method was for optimal use of the processors. The results are presented in Figure 15. The third data set was used in this experiment. The number of processors is 16 and the number of cores per processor is considered to be 4.

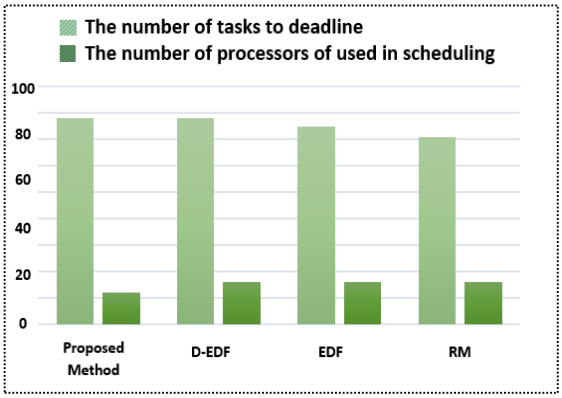

Figure 15. The Efficiency of the Proposed Method to Minimize the Number of Processors Used

In relation to Figure 15, the parameter α can be increased even all the job is completed within the stipulated time. Finally, considering some criteria, we will compare the proposed approach with three RM, EDF and D-EDF is shown in Table 3.

Table 3. Compare the proposed method with Scheduling Algorithms

Performance Criteria	Scheduling Algorithms			
	Proposed Method	D-EDF	EDF	RM
Implementation	Harder	Hard	Average	Simple
The number of switches	Very low	More than EDF	More than RM	Low
The operation of the processor	High	More than EDF	More than RM	Low
The number of job performed at the deadline	High	High	Average	Low
The chances of losing deadline	low	Average	Average	High

Finally, after many experiments, it can be firmly stated that for each three data sets, the proposed method is able to achieve better results from RM and EDF as very useful method that shows the significant efficacy of this approach. But, another point that makes the groundwork for future research is to determine the parameters of the proposed algorithm, especially, the three parameters of mutation probability, probability and size composition of the population.

5. Conclusions and Future Works

Real time systems are an integral part of human daily life. These systems can be found with different interpretations in mobile phones and PCs to power plants and military offices. Of course,

each of these applications have their own simulation applications, but basic concepts are the same in all of them. They have in common. The main features in real time systems is time scheduling based on processing activities. This means that the accuracy of real time tasks, in addition to the dependency to the logical results of processing is deeply dependent upon the time of result production. Since multiprocessor architectures are widely used, it would be clear and more obvious that the future of real time systems is based on multiprocessor architecture. Then, in today's real time systems, the issue of time deadline of jobs should be considered in combination with the features of multiprocessor systems. In this paper, a method for scheduling of jobs in real time systems considering two goals was presented. The purposes of this paper was the increase of the number of done jobs and time deadlines and the decrease of the number of processors have been used in timing, for this purpose we used a genetic algorithm and the reasons for using genetic algorithms are included three cases of proven performance in multi-objective optimization issues, modular structure that allow for the performance of different sections without negative effect on the other sections as well as the lack of need to the knowledge of items such as objective function that make use of it at a wide range of issues.

The proposed method in this paper has high generalizability and this paper can be starting point for more extensive research in the field of improving genetic algorithms in general or due to various issues. The works that can be done in the future include automatically determining the parameters of mutation and combination in order to achieve the best results without the need for error. We can also operate and implement the simulation of the proposed method on a system or software infrastructure, including operating system core.

References

[1] S. Pandit, R. Shedge, Survey of Real Time Scheduling Algorithms, IOSR Journal of Computer Engineering, Vo. 13, No. 2, pp. 44-51, 2013.

[2] S. Kato, A. Takeda, N. Yamasaki, Global Rate-Monotonic Scheduling with Priority Promotion, IPSJ Transactions on Advanced Computing System, Vol 2, No. 1, pp.64-74, 2008.

[3] R. Mohanty, H. S. Behera, K. Patwari, M. Dash, M. L. Prasanna, Priority Based Dynamic Round Robin (PBDRR) Algorithm with Intelligent Time Slice for Soft Real Time Systems, International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 2, No. 2, pp.46-50, 2011.

[4] M.B. Jones, J.S. Barrera III, A. Forin, P. J. Leach, D. Rosu, M. Rosu, An Overview of the Rialto Real time Architecture, 7th workshop on ACM SIGOPS European workshop: Systems support for worldwide applications, pp. 249–256, 1996.

[5] B. Srinivasan, S. Pather, R. Hill, F. Ansari, D. A. Niehaus, Firm Real time System Implementation Using Commercial Of f-The-Shelf Hardware and Free Software, In Proc. IEEE Real time Technology and Applications Symposium, pp.112 – 119, 1998,

[6] S. Kato, N. Yamasaki, Y. Ishikawa, Semi-partitioned scheduling of sporadic task systems on multiprocessors, In Proceedings of the 21st Euromicro Conference on Real time Systems, IEEE, Dublin, pp. 249–258, 2009.

[7] V. Nelis, J. Goossens, D. Milojevic, Energy-Aware Real time Scheduling in Embedded Multiprocessor Systems. PhD Thesis, Université Libre de Bruxelles, Belgium, 2010.

[8] B. Andersson, E. Tovar, Multiprocessor scheduling with few preemptions, In Proceedings of the 12th IEEE International Conference on Embedded and Real time Computing Systems and Applications, RTCSA '06, Sydney, Qld, pp. 322–334, 2006.

[9] J.H. Anderson, V. Bud, U.C. Devi, An EDF-based Restricted-Migration Scheduling Algorithm for Multiprocessor Soft Real time Systems, Real time Systems, Vol.38, No.2, pp.85-131, 2008.

[10] G.C. Buttazzo, Rate monotonic vs. EDF: judgment day. Real time Systems, Vol. 29, No.1, pp.5-26, 2005.

[11] M. Yoo, M. Gen, Multimedia Tasks Scheduling Using Genetic Algorithm, Asia Pacific Management Review, Vol.10, No.6, pp.373-380, 2005.

[12] D. Thakor, A. Shah, D_EDF: An efficient Scheduling algorithm for Real time Multiprocessor System, World Congress on Information and Communication Technologies (WICT), pp.1044 – 1049, 2011.

[13] F.S. Gharehchopogh, I. Maleki, B. Zebardast, A New Approach for Solving N-Queens Problem with Combination of PMX and OX Crossover Operators, Elixir International Journal Computer

Science and Engineering (Elixir Comp. Sci. & Engg.), Vol. 61, pp. 16650-16654, 2013.

[14] R. Nedunchelian, K. Koushik, N. Meiyappan, V. Raghu, Dynamic Task Scheduling Using Parallel Genetic Algorithms For Heterogeneous Distributed Computing, International Conference on Grid Computing & Applications(GCA), Las Vegas, Nevada, USA, 2006.

[15] K. Funaoka, S. Kato, N. Yamasaki, Energy-Efficient Optimal Real time Scheduling on Multiprocessors, The 11th IEEE Symposium on Object Oriented Real time Distributed Computing (ISORC), Orlando, Fl, pp.23-30, 2008.

[16] D. Petrovic, M. Morshed, S. Petrovic, Genetic Algorithm Based Scheduling of Radiotherapy Treatments for Cancer Patients, 12th Conference on Artificial Intelligence in Medicine(AIME 2009), Verona, Italy, Springer Berlin Heidelberg, pp.101-105, 2009.

[17] J. Carretero, F. Xhafa, A. Abraham, Genetic algorithm based schedulers for grid computing systems, International Journal of Innovative Computing, Information and Control, Vol.3, No.6, pp.1-19, 2007.

[18] S.R. Sakhare, M.S. Ali, Genetic Algorithm Based Adaptive Scheduling Algorithm for Real Time Operating Systems, International Journal of Embedded Systems and Applications (IJESA), Vol. 2, No.3, pp.91-97, 2012.

[19] Z. Zhang, W. Hard, A. Stante, Determination of Real time Network Con guration for Self-Adaptive Automotive Systems, Master Thesis, 81 pages, 2015.

[20] V. Guliashki, H. Toshevand, C. Korsemov, Survey of Evolutionary Algorithms Used in Multiobjective Optimization, Bulgarian Academy of Sciences, Bulgaria, 2009.