Indian Journal of Fundamental and Applied Life Sciences ISSN: 2231– 6345 (Online)
An Open Access, Online International Journal Available at www.cibtech.org/sp.ed/jls/2014/04/jls.htm
2014 Vol. 4 (S4), pp. 3089-3097/Vahedi and Mohammadi Baygi
Research Article

APPLICATION OF SOCIAL IMPROVEMENT IN THE GENETIC ALGORITHM IN A TASK SCHEDULING PROBLEM

Vahedi M.A.[1] and *Mohammadi Baygi S.J.[2]
[1]*Payame Noor University, Damghan center, Damghan, Iran*
[2]*Damghan Branch, Islamic Azad University, Damghan, Iran*
**Author for Correspondence*

ABSTRACT
The method of the genetic algorithm is inspired of the evolution of beings in nature, but it is demonstrated that there is a concept of a solitary evolution in the basic genetic algorithm method. In this paper, social improvement in the basic genetic algorithm, which is inspired of the social evolution in nature, is proposed. Results show that the proposed method has more benefits than the basic genetic algorithm and better efficiency and more convergence speed can be achieved, in solving the problems. This method is applied in a problem of multiprocessor task scheduling and the advantage of the proposed method is approved by the simulation results.

Keywords: *Genetic Algorithm, Social Evolution, Task Scheduling, Cross Over, Mutation*

INTRODUCTION
Evolution is the process by which organisms make adaptations to the surrounding conditions. It works through natural selection where individuals who have the best adaptations for reproductive success will be selected. This kind of evolution is inferred initially by this fact that reproduction and life expectancy increases and can probably stay alive and secondly that characters vary among individuals and this can lead to differential rates of survival and reproduction in a population. Of course all of these differences in traits are heritable. Thus, when individuals in a population die they are replaced by the offspring of parents that were better adapted to live and reproduce in the environment in which natural selection took place. This process creates and conserves characters that are apparently fitted for the roles they are carrying out (Wallace, 1855; Darwin, 1859).
Therefore, one may have a dual attitude on evolution in nature, the solitary and the social evolution. In the former, new generations might have better traits than their parents so they have a greater chance to survive and reproduce. However, there is no guarantee for this success. In other words, at the solitary evolution, there is no guarantee for child population to be better individuals than their parents or they may be as good as their parents.
But during the social evolution, in nature, a competition between all parents and children as new parents is performed and the best individuals will be selected for generating the new individuals. In this competition, only the best individuals are able to generate new populations and the weak individuals will be departed from the population. In other words, during social evolution, a competition among all of the first and filial generations is performed and the best ones will be selected for reproduction, For instance, a weak and slow rabbit will be eaten by a fox and there is no chance for it to survive and reproduce, while the strong and fast one can escape and save its generation. Therefore, during this natural selection the weak ones will be mislaid and the strong individuals, also parents, will survive and reproduce until stronger ones appear (Clark, 1984; Nisbet, 1969).
Recently, mankind has a huge role in changing the natural selection of different domestic animals, causing this process to be faster than normal natural conditions. In the genetic evolution of horses, for example, the best ones with the best desired characteristics, either child or parents, will be targeted and selected to reproduce in laboratorial conditions and generate the new population with minimum degree of failure; this procedure continue until the desire population is adopted. It can be mentioned that this process is also carried out for the genetic evaluation of other animals, especially those which are under expiration (Tavasoli, 2003; White, 1969).

Although natural selection causes fitness and adaptation, there are some non-adaptive causes of evolution including mutation and genetic drift. The genetic algorithm method is probably created based on an evolutionary algorithm which is inspired by the evolution of nature and was invented by John Holland.
Idea of evolutionary computing was introduced in 1960s by Rechenberg (1973) in his work "Evolution strategies" (Evolutions strategy in original). His idea was then developed by other researchers so that John Holland and his colleagues invented the Genetic Algorithms (GAs). This led to Holland's book "Adaption in Natural and Artificial Systems" published in 1975 (Holland, 1975).
GA has been theoretically and empirically proved to provide a robust search in complex search spaces. Having been established as a valid approach to the complex problems requiring effective search, GA is now finding more widespread application in business, scientific, and engineering circles. GAs has been widely and successfully used for many optimization problems.
This algorithm is computationally simple and also powerful in its search for the improvement of the solution. GAs are probabilistic techniques that start from an initial population of generated potential solutions to a problem, and regularly evolve towards better solutions through a repetitive application of genetic operators such as crossover, mutation, selection and reproduction (Haupt and Haupt, 2004).
GAs are also problem solving methods which have been successfully used for optimization problems.
Several modifications have been made on the genetic algorithm but the shortcoming of the genetic algorithms is that it has a solitary evolution in its concept.
It has a stochastic selection of parents and also uses genetic operators such as crossover and mutation. Parents generate children and it is significant that, after child generation, parents will be eliminated from the population and they will not be participated in the new generations and while there is no guarantee for the progeny to be better individuals than their parents, the mean objective value of the new generation may deviate from the optimum.
In this paper some modification as the social improvements on the basic genetic algorithm, which is inspired of the social evolution in the nature, is proposed to improve and speed up the performance of the method. To show this concept, the proposed method is applied on a problem of multiprocessor task scheduling and the advantage of the proposed method against the basic genetic algorithm (Hou *et al.*, 1994) is compared.
The rest of this paper is organized as follows; first it explains some details on the problem of multiprocessor task scheduling and deals with working of the basic genetic algorithm. Then, the proposed methodology is explained and the results and discussion are presented.

MATERIALS AND METHODS

Task Graph Model

A set of tasks could be represented by a Directed Acyclic Graph (DAG). The task graph G= (V, E) is a DAG which has a set of nodes V and a set of directed edges E which connect the nodes to each other. The V is a set of m tasks to be executed, so V= $\{T_1, T_2, \ldots T_m\}$.
The directed edges are represented by E=$\{e_{ij}\}$ which e_{ij} is an edge between two tasks T_i and T_j, which specifies that Ti must be completed before T_j starts; the notation $T_i \geq T_j$ is used for this purpose and T_i is a predecessor of T_j and T_j is a successor of T_i.
If there is a path of the directed edges from T_i to T_j then T_i is an ancestor of T_j and T_j is a successor of T_i. A set of predecessors of task T_i is denoted by PRED(T_i). A set of successors of task T_i is denoted by SUCC(T_i).
The height of a task is defined as Equation 1 (Hou *et al., 1994*):

$$height(T_i) = \begin{cases} 0 & if \quad PRED(T_i) = \phi \\ 1 + \max(height(T_j)) & |T_j \in PRED(T_i) \end{cases} \quad \text{Eq. (1)}$$

By definition, for any two tasks T_i and T_j, if T_j is a successor of T_i, then T_j has a larger height value than T_i. If there is no relationship between the two tasks, then they could execute in any arbitrary order. In Figure 1, a DAG is shown which has eight tasks according to their height and their execution time (the time needed for a task to execute).

If there is no relationship between two tasks, then the height-ordering condition will not be used. For example the tasks T_5 and T_6 could be executed in any arbitrary order.

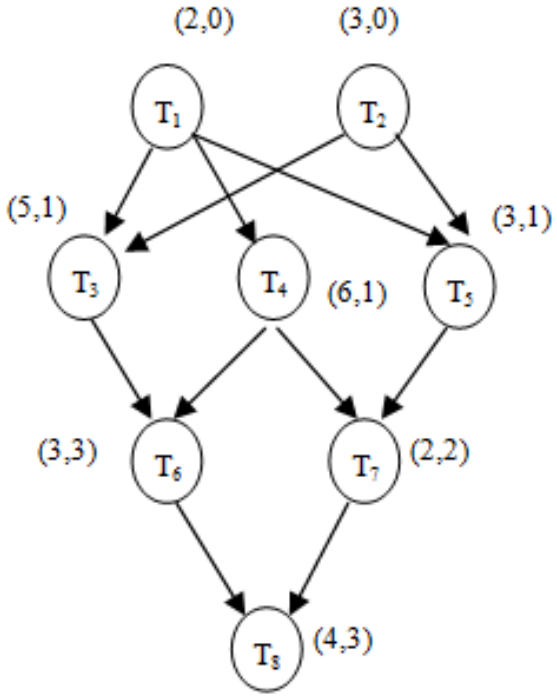

Figure 1: DAG with eight task (execution time, height)

Figure 2 shows a legal schedule on three processors for the DAG in Fig. 1, in which precedence constraints are considered. The length of schedule or the total finishing time is 19.

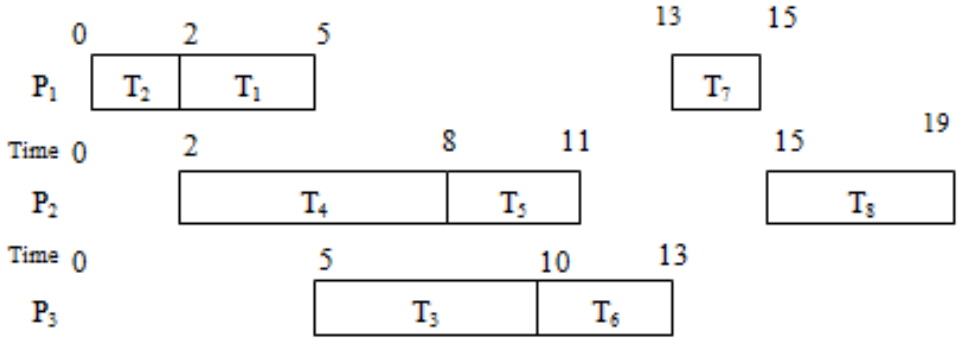

Figure 2: Legal schedule on three processors
Finishing time = 19

The complexity of scheduling problem is very dependent on the DAG (Yang *et al.*, 2011; Wen and Yang, 2011), the number of processors and also the execution time of the task. Many heuristic methods have been proposed to schedule tasks on a multiprocessor system. GAs have also been used for solving problem strategies (Dandass, 2003; Lee, 2003; Shenassa and Mahmoodi, 2006; Daoud and Kharma, 2011).

The problem of task scheduling on multiprocessor system can be stated as finding a schedule for a general task graph to be executed on a multiprocessor so that the schedule length or the finishing time can be minimized (Chitra *et al.*, 2011; Xu *et al.*, 2014). In 1992, John (1992) used the genetic algorithm to

evolve programs to perform certain tasks. He called his method "genetic programming". Hou *et al.* (1994) applied the genetic algorithm on the multiprocessor task scheduling problem. In this paper, the presented algorithm by Hou was chosen as the Basic Genetic Algorithm (BGA).

The Basic Genetic Algorithm

The following is an outline and some details of the basic genetic algorithm in solving a task scheduling problem, presented by Hou. A brief description of this method is presented in Ref 10:

A1- Generate schedule for the problem N times and store the string created in POP.

This step is used to produce an initial population of the generated potential solutions to the problem. Each string of the potential solution corresponds to a legal search node which satisfies the following conditions:

The precedence relations among the tasks are satisfied;

Every task is presented and appeared only once in the schedule; (completeness and uniqueness)

A2- Do step A3-A8 until the algorithm is converged;

A3-Compute the fitness value of each string in POP;

The fitness value is typically the objective function that should be optimized in the problem. It is used to evaluate the search nodes and to control the genetic operators.

A4- Do the selection process;

Good strings have better fitness value and therefore should be preserved into the next generation. A biased Roulette wheel is used to implement the selection process.

A5- Do step A6 NPOP/2 times, for which NPOP is the number of strings in the population;

A6- Pick two strings by the selection process and perform the crossover with a probability of *cr*. If the crossover is performed, put the new strings in TMP; otherwise, put the two strings picked in TMP.

The crossover starts with two parent chromosomes (Omara and Arafa, 2010) to exchange their subparts and create two new children as shown in Fig. 3 considering the following steps:

1- Select the crossover points where the list could be cut into two halves.

2- Exchange the bottom halves of strings A and B from the crossover point.

The crossover is applied with a certain crossover rate.

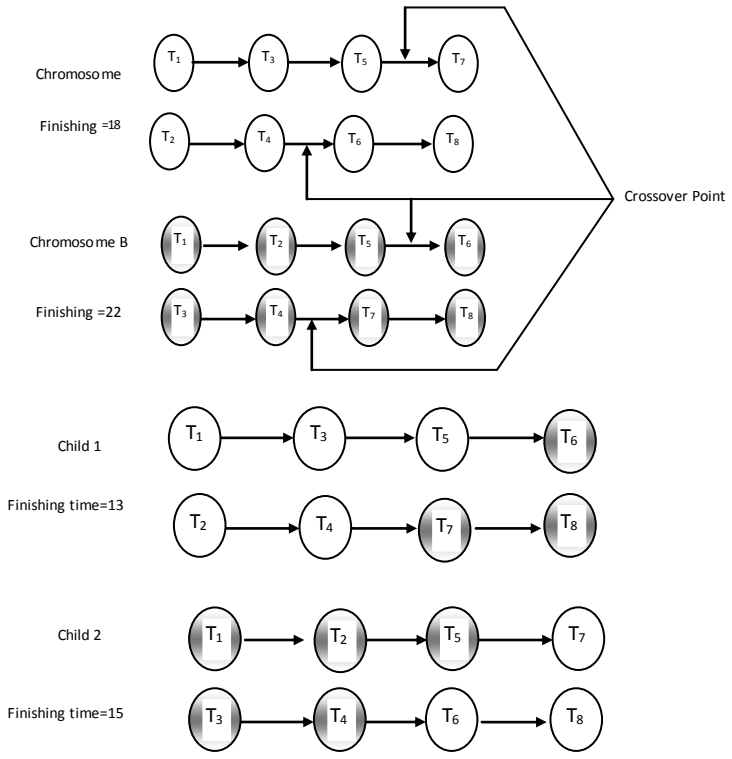

Figure 3: A scheme of the cross over process

A7- On the each of the strings in TMP perform mutation with a probability of *mr*. If it is done, replace the new string in TMP; otherwise, place the string picked in TMP;
The mutation selects a chromosome and then randomly exchanges the two tasks with the same height (fig. 4). The mutation is applied with a certain mutation rate (*mr*) which is used to prevent the search process from converging to local optimal prematurely.

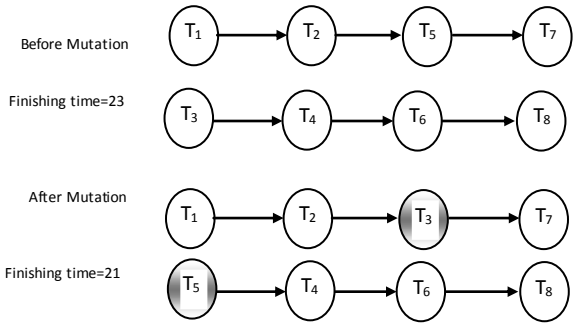

Figure 4: A scheme of the mutation process

A8- Replace the strings in POP with new strings in TMP;

Social Evolution in the Genetic Algorithm

When the children are generated from the parents, in the BGA, in which the mutation or crossover is performed, the parents will be omitted and the children will be selected to appear in the next population. Knowing that a progeny may not be a better individual than its parents, having a better fitness value, this may cause an inferior objective value in the next population. Also, it causes the oscillation of the mean objective value of the whole algorithm and because of this oscillation more time and more generation is required for the algorithm to converge to optimal solution.
But it is known that during the social evolution in nature and natural selection, the weak individuals will be mislaid and the strong ones, also parents, will survive and reproduce until stronger ones appear. Also during the social evolution in nature, a competition between all the parents and children as new parents is performed and the best ones will be selected for generating the new individuals. In this competition, only the best individuals are able to generate a new population and the weak parents will be eliminated from the population.
Using this concept, in this paper, a social improvement, based on the social evolution of nature, is defined in the BGA. For this purpose, the combined population is defined as the combination of all the parents and the progenies strings. In the social genetic algorithm, SGA, a competition between all the strings of the combined population is performed and the best individuals will be selected for generating the new population. In this competition, only the best individuals, who have a better fitness value, are able to generate a new population and the weak strings will be eliminated from the population. For this concept, the necessary condition for each string of the combined population is to have a better objective value than half of the combined population to be allowed to appear in the next population. Due to this selection, the mean objective value of the new populations gets better with time and as a result, the mean objective value of the whole algorithm gets better during the generations and it prevents the oscillation of the mean objective value or the divergence of the algorithm. Some results of this concept are depicted in Section 5.
During the social improvement in the BGA, the members of the combined population are evaluated and the first half members of the combined population will be selected as the new population. In this proposed algorithm, the child generation of BGA is modified as follows:
B1- Generate the schedule for the problem N times and store the string created in POP;
B2- Do step B3-B9 until the algorithm is converged;
B3- Compute the fitness value of the each string in POP;
B4- Sort all the strings in POP according to their fitness value;
B5- Do step B6 NPOP/2 times;

B6- Pick two strings using the selection process and perform the crossover with a probability of *cr* on the two strings. If the crossover is performed, put the new strings in TMP; otherwise, put the chosen strings in TMP.
B7- On the each of the strings in TMP performs mutation with the probability *mr*. If the mutation is performed, replace the new string in TMP; otherwise, place the chosen string in TMP.
B8-Sort all the strings in TMP according to their fitness value and merge them with POP to generate the combined sorted population according to their fitness value.
B9- Select the first NPOP strings of the combined population as new POP
The other shortcoming of the BGA is that a non-efficient or a slightly big value of the mutation rate (*mr*) might cause the algorithm to diverge. However, in the SGA, a big or non efficient value of the mutation probability rate may not cause the algorithm to diverge. In the SGA, the children should have a better objective value than half of the combined population to be allowed to appear in the next generation and as a result the method just converges to the optimal value without any oscillation.

RESULTS AND DISCUSSION

Comparing the proposed SGA and the BGA, a set of simulation is performed under a set of common assumptions by OCTAVE. An Intel Pentium Dual CPU 1.8 GHZ with 2GB of RAM is used to implement these algorithms. For this purpose, a set of 10 random graphs consisting of 100 to 150 tasks are generated with random execution times of 2 to 50sec. These tasks would be scheduled on multiprocessor with 10 to 13 processors as listed in table 1.
After generating the data, the task scheduling of the problem using the BGA and SGA is performed and the parameters of the both algorithms are determined. The number of generations and the stop condition is set as 200 generations. The crossover and the mutation rate are set as 0.85 and 0.06 respectively; however, in the SGA the mutation rate is set as 0.25(this mutation rate will diverge the BGA).

Table 1: Multiprocessor task scheduling data

Graph Number	Number Of Tasks	Number Of Processors
1	100	10
2	105	10
3	110	10
4	115	11
5	120	11
6	125	11
7	135	12
8	135	12
9	140	12
10	150	13

Each graph in table 1 is scheduled three times for the two algorithms. The average computation time and the average finishing time are calculated for each of the graphs, while finding the sub-optimal schedule for each algorithm.
Figure 5 represents the average computation time vs. the generation progress in the two Algorithms.

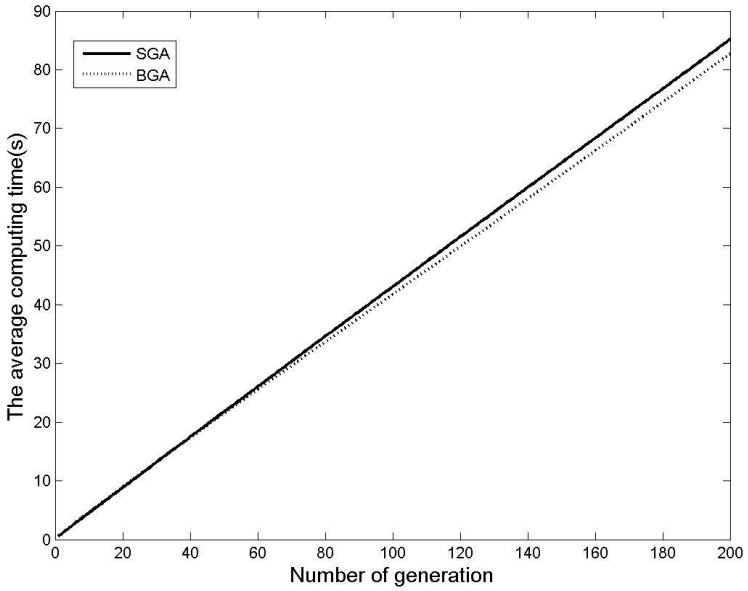

Figure 5: The average computation time vs. the generations

Figure 5 shows that the average computation time varies linearly according to the increase in the number of the generations in the BGA and SGA. It also shows that the SGA algorithm has a higher computation time in finding the suboptimal schedule. The speed of the SGA algorithm appears to be 0.97 times that of the BGA in 200 generations.
Figure 6 also comprise the finishing time of the two algorithms against the generations.

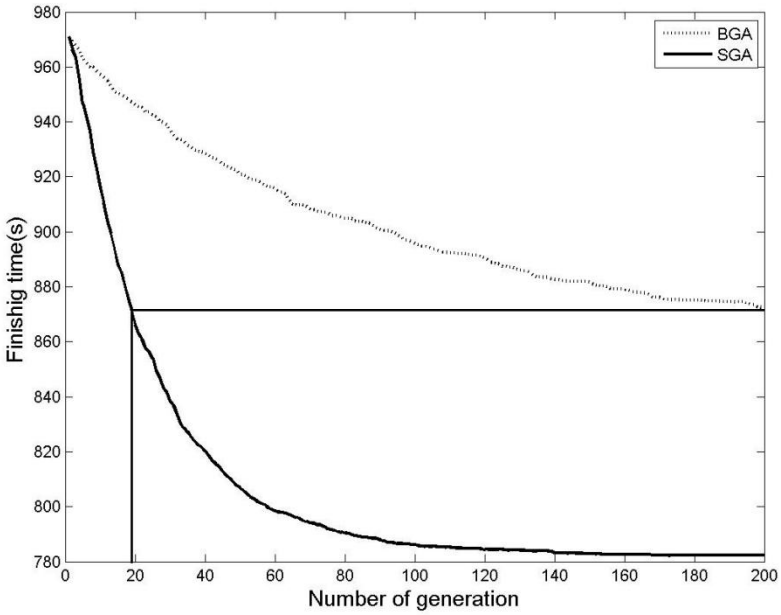

Figure 6: The idea of the SGA speed

Table 2 represents a comparison of the finishing time and the improvement rate of the proposed method to the BGA in 200 generations as follows:

Table 2: Finishing time and the percent of improvement

	BGA	SGA
Finishing time(s)	873.4	781.2
Improvement from initial point	99.4	191.6
Improvement acc. to BGA(%)	-	92.7

While it is desired to reduce both criteria simultaneously, the optimal solution is a conflict between the computation time and finishing time. Results in tables 2 demonstrate that the SGA algorithm has reduced the finishing time in which the optimal solution in the SGA algorithm is approved at about 92.7%.
To demonstrate and highlight the advantage of the SGA against the BGA, the solution result of the BGA after 200 generations is compared with the same result in the SGA algorithm. Figure 6 also shows that the final solution of the BGA is achieved after 200 generations and 82.9 sec (figure 5) but the same result is achieved at the SGA algorithm in 18 generations and 7.93 sec, which means that the SGA algorithm has about 10 times more speed than BGA to get its solution results.
To show the effect of SGA on the oscillation of mean objective value of the algorithms and to demonstrate the behavior of all chromosomes in the generations in the BGA and SGA, Fig 7shows the average finishing time vs. the generation's progress for the time scheduling of 30 tasks in 3 processors (for instance).

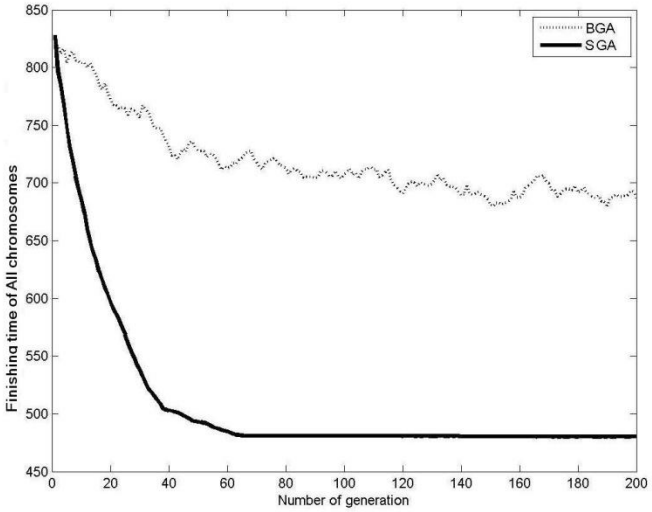

Figure 7: Average finishing time vs. the number of generations

As it shows, the mean objective value of BGA is oscillating. In the BGA method, there are many generations for which their mean objective values are not better than their parents and as a result, the oscillation of the BGA will occur. Because of these oscillations, more generations are required to improve the objective function of the algorithm and this would decrease the convergence speed of the algorithm to optimum. In contrast to the BGA, the SGA algorithm converges to a better local optimum with no oscillation. Figure 7 also comprises the finishing time vs. the generation progress for the two algorithms. However, the SGA algorithm demonstrates the better finishing time for the problem.

Conclusion

In this paper, a social evolution methodology in the basic genetic algorithm is proposed. In order to show the effectiveness of this method, an experimental simulation is performed by applying the two algorithms to different task graphs.
In the social evolution methodology, the necessary condition for each string is to have a better objective value than half of the combined population to be allowed to appear in the next population. This concept

causes the proposed algorithm to converge to optimal solution without any oscillation. Also, in contrast to the BGA, a big value of the mutation rate does not affect the convergence of the proposed algorithm.

REFERENCES

Chitra P, Rajaram R and Venkatesh P (2011). Application and comparison of hybrid evolutionary multiobjective optimization algorithms for solving task scheduling problem on heterogeneous systems, *Applied Soft Computing* **11**(2) 2725-2734.

Clark L (1984). *Social Darwinism in France* (University of Alabama Press) Tuscaloosa.

Dandass YS (2003). A Genetic Algorithm for Scheduling Acyclic Digraph in the presence of Communication Contention. *In Proceedings of the 17th Annual International Symposium on High Performance Computing Systems and Applications*, Quebec, Canada 223-230.

Daoud MI and Kharma N (2011). A hybrid heuristic–genetic algorithm for task scheduling in heterogeneous processor networks, *Journal of Parallel and Distributed Computing* **71**(11) 1518-1531.

Darwin C (1859). *On the Origin of Species* (John Murray publication) London, retrieved 2011-03-01, 502.

Haupt RL and Haupt SE (2004). *Practical Genetic Algorithms,* 2nd edition, with CD (John Willy & Sons) Hoboken, New Jersey, ISBN 0-471-45565-2.

Holland J (1975). Adaptation in natural and artificial systems (The University of Michigan Press) Ann Arbor, MI.

Hou ESH, Ansari N and Ren H (1994). A Genetic Algorithm for Multiprocessor Scheduling. *IEEE Transactions on Parallel and Distributed Systems* **5**(2)113-120.

Koza JR (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (MIT Press) ISBN 0-262-11170-5.

Lee YH and Chen C (2003). A Modified Genetic Algorithm for Task Scheduling in Multiprocessor Systems. *The 9th workshop on compiler techniques for high-performance computing*, Taipei, Taiwan.

Nisbet RA (1969). *Social Change and History: Aspects of Western Theory of Development.* New York (Oxford University press) 335.

Omara FA and Arafa M (2010). Genetic algorithms for task scheduling problem, *Journal of Parallel and Distributed Computing* **70**(1) 13-22.

Rechenberg I (1973). Evolutions strategie - Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. PhD thesis, 1971, Reprinted by Fromman-Holzboog.

Shenassa MH and Mahmoodi M (2006). A Novel Intelligent Method for Task Scheduling in Multiprocessor Systems Using Genetic Algorithm. *Journal of Franklin Institute* (Elsevier) **343**(4) 361-371.

Tavasoli GA (2003). *Sociological Theory* (Samt Publication) Tehran.

Wallace AR (1855). On the Law which has Regulated the Introduction of New Species. *Annals and Magazine of Natural History* 2nd Series.

Wen Y, Xu H and Yang J (2011). A heuristic-based hybrid genetic-variable neighborhood search algorithm for task scheduling in heterogeneous multiprocessor system, *Information Sciences* **181**(3) 567-581.

White L (1969). The Social Organization or Ethological Theory. *In Rice University Studies: Monographs in Cultural Anthropology* **52**(4) 1-66.

Xu Y, Li K, Hu J and Li K (2014). A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues, *Information Sciences* **270** 255-287.

Yang J, Xu H, Pan L, Jia P, Long F and Jie M (2011). Task scheduling using Bayesian optimization algorithm for heterogeneous computing environments, *Applied Soft Computing* **11**(4) 3297-3310.