Task Scheduling and Idle-Time Balancing in Homogeneous Multi Processors: A Comparison between GA and SA

Mohammad Amin Pishdar
Department of Computer Engineering, Bandar Abbas Branch, Islamic Azad University, Bandar Abbas, Iran

Abbas Akkasi
Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran

ABSTRACT
Task scheduling problem has a special significance in multiprocessors due to efficient use of the processor and also spending less time. Tasks should be assigned to processors in such a way to minimizing makespan. In this paper, we use genetic algorithm and simulated annealing to solve task scheduling problem on multi homogenous processors with minimizing completion time. In addition we introduce another fitness function as processors idle-time balancing which should be less than a predetermined value. These algorithms are used to determine suitable priorities that lead to a sub-optimal solution. And finally to compare the performance of these algorithms, we design 9 test problem based on two fitness function.

Keywords
Genetic algorithm, multiprocessor task scheduling, parallel processing, simulated annealing.

1. INTRODUCTION
Computers can be used by several processors instead of a processor in which case they are called multiprocessing systems. In order to use these systems require a specific operating system that can process multiple applications or threads to execute in parallel on them. One of the main challenges in multi-processor systems which work is scheduled to be optimized. Task scheduling problem has a special significance in multiprocessors due to efficient use of the processors and also spending less time. Tasks should be assigned to processors in such a way to minimizing makespan. It is such a NP-hard problem, no algorithm is able to solve it definitively. This problem has been solved by many meta-heuristics algorithms.

As reviewed above, this article addresses task scheduling on multi-processors problems which solved by GA and SA.

This paper is organized as follows: Relevant previous researches are reviewed in section (2). In section (3) the problem is described. In section (4) the solving technique and characteristics of meta-heuristics are explained. Finally these algorithms are compared and the results are shown section (5).

2. LITERATURE REVIEW
Dahal et al [1] used genetic algorithm for dynamic scheduling of real-time tasks in a multi processor system to obtain a feasible solution. They used genetic algorithms combined with well-known heuristics, such as 'Earliest Deadline First' and 'Shortest Computation Time First. Miryani and Naghibzade [2] present study on Scheduling in Heterogeneous Systems. They use a multi objective approach for problem. They proposes a suboptimal scheduler for hard real time heterogeneous multiprocessor systems considering time constraints and cache reload time simultaneously. tainer and white [3] introduce a hybrid metaheuristic algorithm for solving the problem of multi-core deployment optimization (MCDO). they used Simulated Annealing and Ant Colony for hybrid algorithm. they showed performance of the algorithm by comparing with other algorithms. Hooshmand et al [4] perform a study on Scheduling of Task Graphs to Multiprocessors problem. for solving the problem they used a Combination of Modified Simulated Annealing and List based Scheduling. Gupta et al [5] used a genetic algorithm to solve the problem of multi-processor scheduling. To evaluate the performance, they compared genetic algorithm with HEFT. Their comparison is based on quality of answers, robustness and the effects of mutation on the function of the genetic algorithm. Omara and arafa [6] use two genetic algorithms for solving task scheduling in parallel processing. They consider some heuristic rules for better performance if algorithms. They present to fitness function which first of them minimize completion time and the second one consider load balancing. Wen et al [7], addresses a hybrid genetic and VNS algorithm for task scheduling in homogenous processors. They consider some good neighbor structure for minimizing completion time in task scheduling. Wu et al [8] developed task scheduling for multi-processors systems. They use genetic algorithms for solving their problem. Key features of their system was include a flexible, adaptive representation and an incremental fitness function. Roy et al [9] proposed a modification of heuristic approach of genetic algorithm method based on bottom-level by choosing the eligible processor for assigning the tasks which eventually decreases the computational time for finding the suboptimal schedule. Thanushkodi and Deeba [10] developed job scheduling in multi processors. They proposed Genetic algorithm, particle swarm algorithm and improved particle swarm algorithm to solve problem. Finally they compared performance of algorithms.

3. PROBLEM DESCRIPTION
The task scheduling problem is to assign a number of jobs onto the set of available processors in such a way that precedence constraints are maintained with the objective to minimize the completion time [11]. In addition, another objective function as balancing the processors idle time is used in this paper which should be less than a predetermined value (Beta). The second objective function value is calculated by difference between maximum idle-time and minimum idle-time.

In this paper as illustrated in Fig 1. a directed acyclic graph (DAG) consisting of 16 tasks is shown should be assigned to

the 3 Homogenous processors in such a way that precedence constraints are satisfied and to determine the start and finish times of each task with the objective to minimize the makespan and also to minimize the processors idle-time balancing.

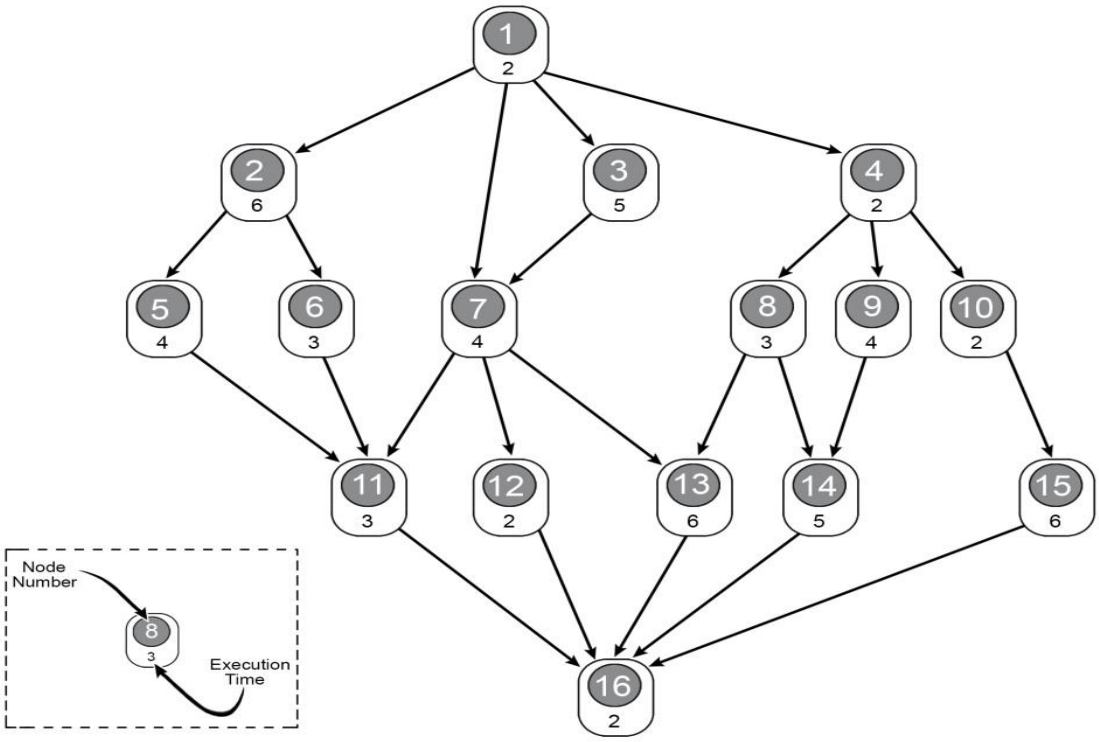

Fig 1: Directed Acyclic Graph (DAG)

Assumption:

- The processors are homogenous.
- According to graph node 1 can be ignored but it's time is considered in Gantt chart.
- Times follow uniform distribution between (2,6).
- Beta is equal to 5.

An example of task Gantt chart is shown in Fig 2. The completion time and each processors idle-time is visible. In this example, maximum idle-time and minimum idle-time belongs to P3 and P2 respectively. So the difference between them is equal to 8 which is not possible for us.

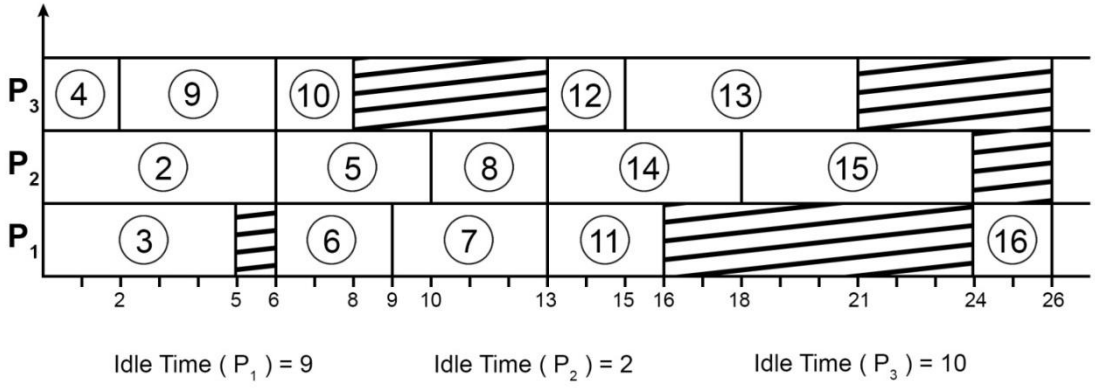

Fig 2: Task Gantt Chart

4. SOLUTION ALGORITHMS

4.1 Genetic algorithm

Steps of GA are organized below:

1. [Start] Generate random population of n chromosomes (suitable solutions for the problem).
2. [Fitness] Evaluate the fitness f(x) of each chromosome x in the population.
3. [New population] Create a new population by repeating following steps until the new population is complete.
    - [Selection] Select two parent chromosomes from a population according to their fitness (the better fitness, the bigger chance to be selected).
    - [Crossover] With a crossover probability cross over the parents to form a new offspring (children). If no crossover was performed, offspring is an exact copy of parents.
    - [Mutation] With a mutation probability mutate new offspring at each locus (position in chromosome).
    - [Accepting] Place new offspring in a new population.
4. [Replace] Use new generated population for a further run of algorithm.
5. [Test] If the end condition is satisfied, stop, and return the best solution in current population .
6. [Loop] Go to step 2 [12].

4.1.1 Initial population

Initial population, firstly constructed randomly as an array which is consisted all tasks expect task 1. Task 1 as initial node that has any precedence, assign randomly to one of processors and then all tasks will be scheduled with considering precedence. After array construction, tasks will be assigned to processors by equation (1).

P: Number of processors.

N: Number of tasks.

S: Number of tasks which are assigned to each processor.

$$S = \frac{N}{P} \quad (1)$$

As illustrated in Fig. 5. (Step 1), first five tasks{16,3,6,11,7} assign to processor P1. Processor P2 is consist of tasks{14,8,5,15,2} and tasks{4,13,9,10,12} is assigned to last processor.

4.1.2 Crossover

Each chromosome in population is subject to crossover with some probability. Crossover operator randomly selects two parent chromosomes by using roulette wheel selection procedure and tournament selection procedure. In this paper, two crossover operators are used which are chosen randomly.

- Single point crossover
  Each This operator is applied to the array of chromosomes. A point is selected randomly between 1 to number of tasks and divide parent 1 and parent 2 in two portion. The portions of chromosomes lying to the right of crossover point are exchanged to produce two offsprings. (Shown in Fig 3.).
- Double point crossover
  Two point is selected randomly between 1 to number of tasks and divide parent 1 and parent 2 in three portion. The portions of chromosomes lying to the between of two crossover points are exchanged to produce two offsprings. (Shown in Fig 4.).

After producing offsprings by crossover, it's may be possible to face some frequent tasks in array which should be solved. For this problem, first, all the frequent tasks has been specified in two producing array and exchange their position to modify the offsprings. (Shown in Fig 3. and Fig 4).

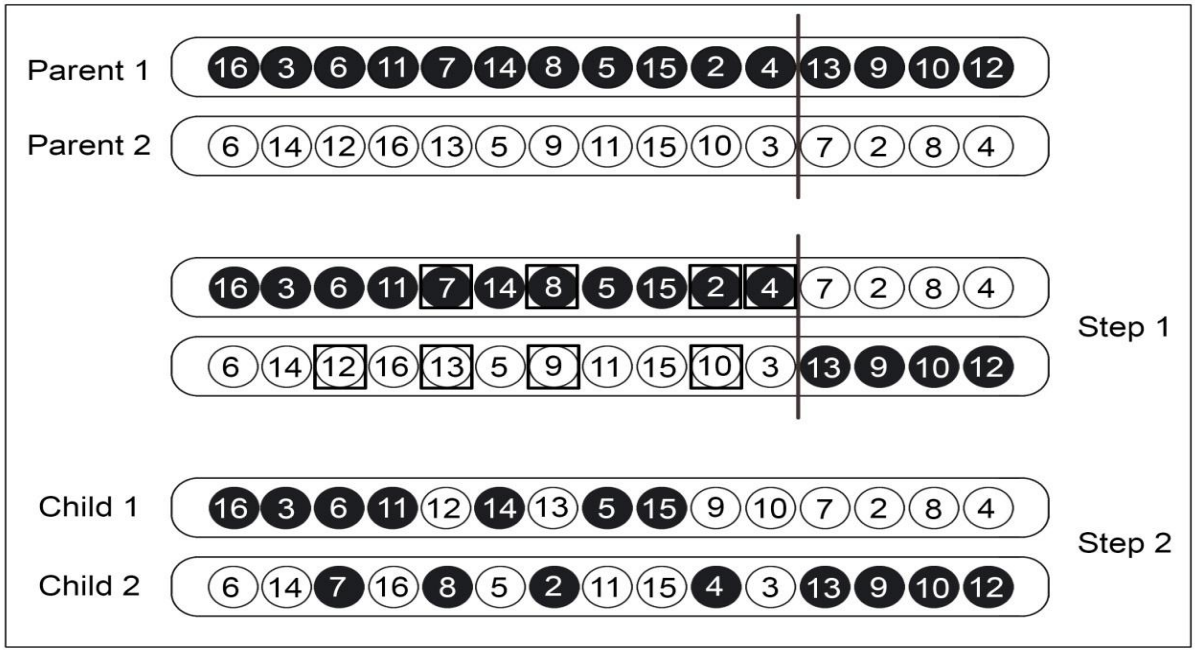

Fig 3: Single Point Crossover

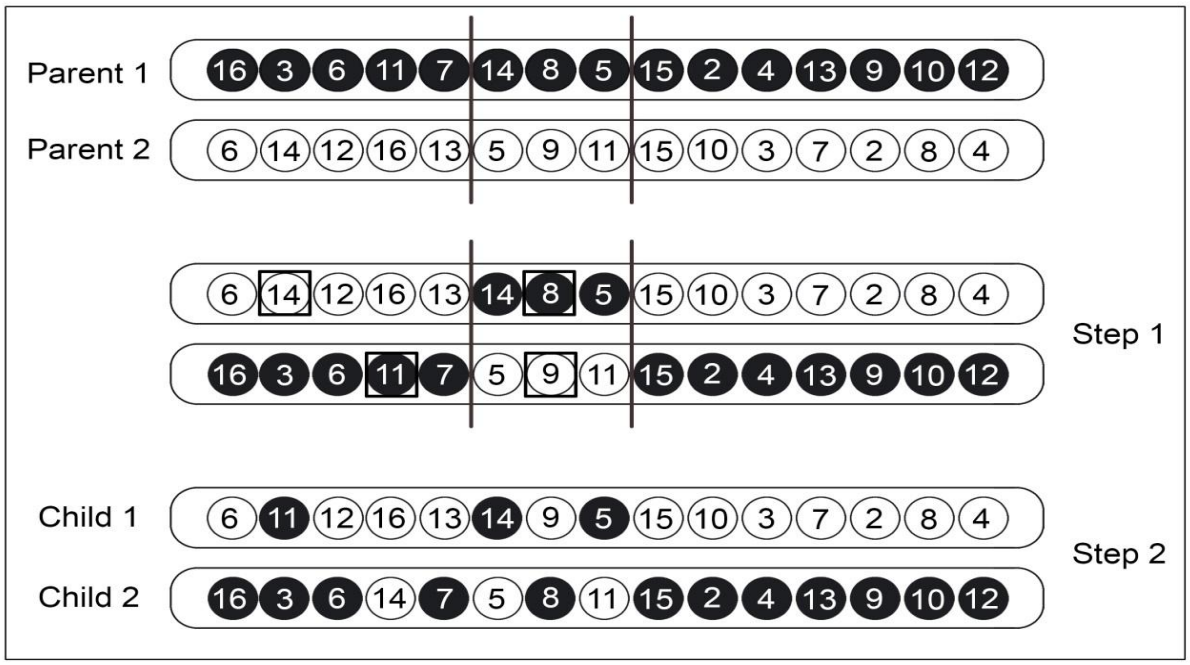

Fig 4: Double Point Crossover

4.1.3 Mutation

According to problem, mutation operator change the position of selected task. In this paper, two different methods are used for mutation operation which choose randomly.

- Matching and swapping Mutation

  As illustrated in Fig. 5. First, tasks of array are assigned to processors (Step 1). Then select on point between tasks randomly. For each portion it is assigned three random numbers between 1 and 3 separately (Step 2). After that for two portions with same matching number, right side of mutation point of one portion is transmitted to the left side of other one (Step 3). And finally change it to on array again (Step 4).

- Mutation based on completion time
  This kind of mutation performs mutation operation with attendance to completion time (CT) of processors. first it specifies two processors which have maximum and minimum CT and then, choose two tasks randomly among these processors and change their positions. (Fig. 6.)

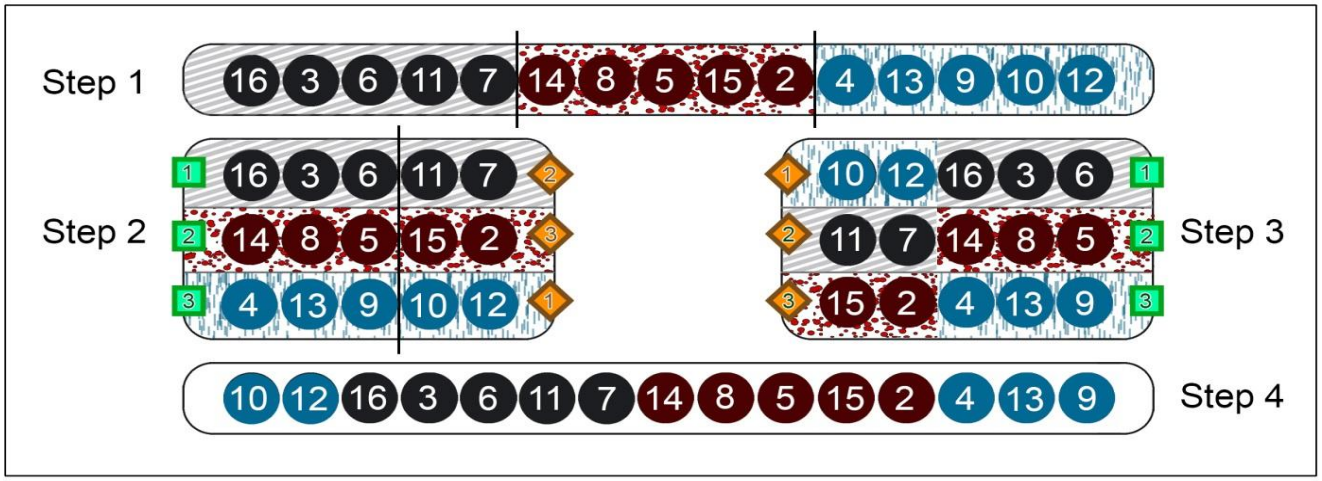

Fig 5: Matching and swapping Mutation

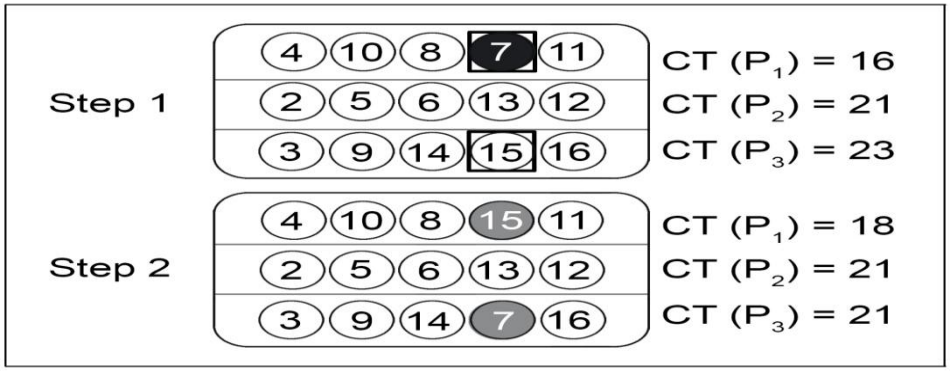

Fig 6: Mutation based on completion time

4.2 Simulated annealing algorithm

Steps of SA are organized below:

1. First, Generate a random solution and evaluation.
2. Consider the solution as the best answer.
3. Set the initial temperature. (T = T0)
4. Repeat steps below until an acceptable solution is found or you reach maximum number of iterations.
   - Generate a random solution in the neighborhood of the current response and evaluation.
   - New Reply in the event of better reception.
   - New Reply conditional acceptance if they not better.
   - The solution has been updated.
   - Reduce temperature.
5. If you need, return to Step 4.

4.2.1 Neighborhood structures

In simulated annealing algorithm, it is considered four neighborhood structures which are explained below (Fig.7.):

- Swap: two units of a solution are selected randomly and their positions are swapped.
- Reversion: in addition to swap, units located between swapped units are reversed, too.
- Insertion: the unit in the second position is located immediate after the unit in the first location and the other units are shifted to the right hand side accordingly.
- Swapping a reversed part of solution (SRPS): a subsequence of solutions is selected and then shifted to a new position, and also the subsequence selected part of solution is reversed

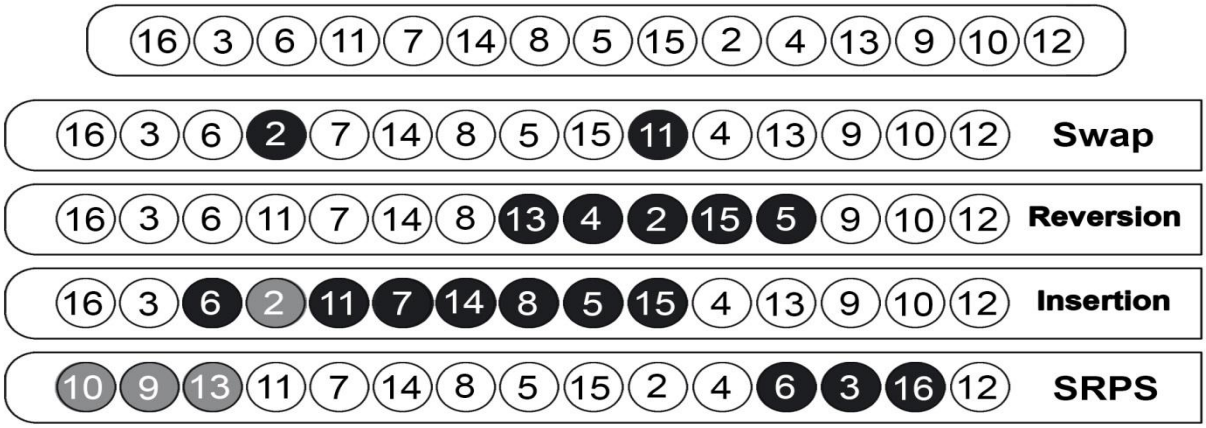

Fig 7: Neighborhood structures

4.2.2 Temperature reduction rule

In this paper, dynamic rule is used with some changes for temperature reduction rule. Based on this rule, temperature reduces with considering to the search region or solutions. In this rule if after some iteration, algorithm achieves almost same fitness function it should be used linear temperature reduction as shown in equation (2).

T_0: Initial temperature
i: Number of reduction temperature stage

β : Constant value between (0,1)

$$T_i = T_0 - i * \beta \quad (2)$$

Otherwise it used logarithmic rule which reduce temperature by less rate and has more convergence to global optimization. (Equation 3)

$$T_i = \frac{T_0}{\log(i)} \quad (3)$$

5. CONCLUSION

In this paper, genetic algorithm and simulated annealing algorithm is used for task scheduling problem based on directed acyclic graph (DAG) consisting of 16 tasks and 3 processors. 9 different type of testing problem is defined to evaluate performance of algorithms. The results are shown in Table 1. and Fig 8. and Fig 9. it should be noted that solutions are acceptable for us which have idle-time balancing less than or equal to 5.

Table 1. Comparison Between Ga And Sa

		GA			SA		
Max iteration	Initial population	Time completion	Idle-time balancing	Acceptable or not	Time completion	Idle-time balancing	Acceptable or not
200	100	37	8		36	7	
500	100	33	9		32	5	**
1000	100	29	5	**	29	4	**
200	200	30	9		32	7	
500	200	26	4	**	27	5	**
1000	200	21	5	**	22	6	
200	500	27	5	**	21	5	**
500	500	21	6		21	4	**
1000	500	21	4	**	21	4	**

The best solution is reached in Iteration=1000 and Npop=500 for Genetic algorithm and in Iteration=500,1000 and Npop=500 for simulated annealing algorithm. In 9 type testing problem Genetic algorithm give us just 5 acceptable solution in comparison to simulated annealing algorithm with 6 acceptable solution.

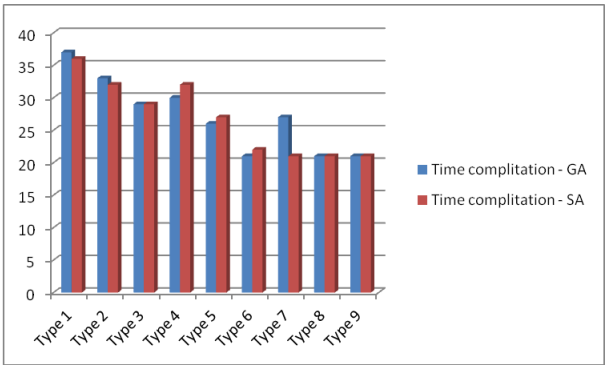

Fig 8: Completion time

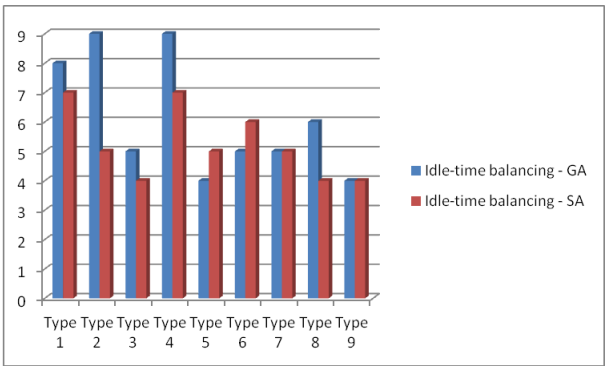

Fig 9: Idle time

6. REFERENCES

[1] Dahal, K. and Hossain, A. and Varghese, B. and Abraham, A. and Xhafa, F. and Daradoumis, A. (2008). Scheduling in Multiprocessor System Using Genetic Algorithms. 7th Computer Information Systems and Industrial Management Applications.

[2] Miryani, M. R. and Naghibzadeh, M. "Hard Real-Time Multiobjective Scheduling in Heterogeneous Systems Using Genetic Algorithms," Proceedings of the 14th International CSI Computer Conference (CSICC'09)., 2009, pp. 437-445.

[3] Turner, H. and White, J. (2013). Multi-core Deployment Optimization Using Simulated Annealing and Ant Colony Optimization. 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications.

[4] Houshmand, M. and Soleymanpour, E. and Salami, H. and Amerian, M. and Deldari, H. (2010). Efficient Scheduling of Task Graphs to Multiprocessors Using A Combination of Modified Simulated Annealing and List based Scheduling. Third International Symposium on Intelligent Information Technology and Security Informatics.

[5] Gupta, S. and Agarwal, G. and Kumar, V. (2010). Task Scheduling in Multiprocessor System Using Genetic Algorithm. Second International Conference on Machine Learning and Computing.

[6] Omara, F. A. and Arafa, M. M. (2010.). Genetic algorithms for task scheduling problem. J. Parallel Distrib. Comput. 70, pp. 13-22. Available: www.elsevier.com/locate/jpdc

[7] Wen, Y. and 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, pp. 567-581. Available: www.elsevier.com/locate/ins

[8] Wu, A. S. and Yu, H. and Jin, S. and Lin, K. and Schiavone, G. (2004, Sep.). An Incremental Genetic Algorithm Approach to Multiprocessor Scheduling. IEEE Transactions on Parallel and Distributed Systems. 15(9), pp. 824-834.

[9] Roy, P. and Alam, M. M. and Das, N. (2012, July.). Heuristic Based Task Scheduling In Multiprocessor Systems With Genetic Algorithm By Choosing The Eligible Processor. International Journal of Distributed and Parallel Systems (IJDPS). 3(4), pp. 111-121.

[10] Thanushkodi, K. and Deeba, K. (2011, May.). On Performance Comparisons of GA, PSO and proposed Improved PSO for Job Scheduling in Multiprocessor Architecture. International Journal of Computer Science and Network Security (IJCSNS). 11(5), pp. 27-34.

[11] Kaur, R. and Singh, G. 2012. Genetic Algorithm Solution for Scheduling Jobs in Multiprocessor Environment. India Conference (INDICON).

[12] http://cs.nyu.edu/courses/fall12/CSCI-GA.2965-001/geneticalg.