HYBRID ALGORITHM BASED ON ANT AND GENETIC ALGORITHMS FOR TASK ALLOCATION ON A NETWORK OF HOMOGENEOUS PROCESSORS

Sawsan Yousef Abu Shuqeir[1] and Tamara Amjad Al Qublan[2]

[1]Departement of Information Technology, AL Balqa Applied University, Al Huson, Jordan

[2]Departement of Information Technology, AL Balqa Applied University, Al Huson, Jordan

ABSTRACT

In the field of parallel computing, there is an essential problem which is called Task Allocation Problem(TAP). The task allocation problem (TAP) is a problem where many of tasks require to be allocated to a set of processors.The number of m tasks that is needed to be allocated with number of n processors where (m>n) so that the time needed to process all the tasks is minimized. This paper presents an efficient algorithm (TAP_ACO_GA) to solve the task allocation problem. The proposed algorithm is based on the idea of Ant Colony Optimization Algorithm(ACO) and the idea of Genetic Algorithm (GA) .The proposed algorithm is tested in different dataset and its results are compared with the results in[2] and the results show that TAP_ACO_GA is better than the other algorithm.

KEYWORDS

Ant Colony Optimization , Task Allocation , Parallel Programming, Genetic Algorithm .

1. INTRODUCTION

Parallel computing is the use of more than one processor at the same time to execute a program or many of computational threads. Given an instance of a computational problem and a parallel machine with a certain number of processors, we would like to solve the instance as quickly as possible with the available processors. If we say that the computation starts at time 0, this means that we want to minimize the overall finishing time or make span of the computation, i.e., the finishing time of the last task to complete[10]. The problem of task allocation can be defined in terms of the count of tasks and the count of processors that are available. When there are more than three processors to solve the problem, then TAP is known to be NP-hard . The tasks can be executed simultaneous or waiting to the completion of other tasks if there is dependency between tasks. Tasks can be executed in sequence or at the same time on one or more processors. A cluster of computer consists of more than one computer that are linked through a LAN. The networked computers act as one that is more powerful machine. A computer cluster gives speed in processing , better in data integrity, the ability to handle large computational load, and larger in storage capacity[2] .

Task allocation problem can be classified as one of the main research problems for parallel system in which tasks are assigned to different processors of the network, in such a way that will lead to minimization in the make span, Make span is the time that is needed to process all the tasks

t1+t2+t3...+tm on processors p1,p2,...pn where (m>n). These problems may be categorized as static and dynamic types of task allocation [2].

Many algorithms for task allocation have been presented. The purposes of these algorithms are to balance the load among processors, reduce the execution time of the parallel program, minimize the makespan and others[2]. In our proposed algorithm we have apply the idea of Ant Coloy Optimization Algorihim and Genetic Algorithm to find the optimal solution for allocating the task to the processors to complete the job of the problem in minimum makespan and load balance. The rest of this paper is structured as follows: Section 2 presents Ant Colony Optimazation. Section 3 presents Genetic Algorithm. Section 4 presents some related research background. Section 5 explains the problem, while Section 6 shows how the proposed algorithm solve the TAP problem . Section 7 shows the results after testing the program on the proposed algorithm. Finally, Section 8 shows the conclusion of the paper and the future works.

2. ANT COLONY OPTIMIZATION

Ant Algorithms [6] are a recently developed as a new approach that has been successfully use to solve many of NP-hard combinatorial optimization problems. The idea of Ant Colony Algorithms is suggested based on the behavior of real ants , real ants are a clever creatures that are capable to find the shortest path to the source of food from the nest without using visual cues ,And if the path to the food source is damaged by any ostacle ,then the ants are also capable of adapting to these changes in the environment,so that they also find a new shortest path[4-5][7]. During their walk ants put a small amount of natural material called pheromone on the ground this tell other ants that it follow this path if an ant reaches to the source food in a shorter path then it will return to the nest before the other, and its path will have more pheromone than the other pathes followed by the other ants, and this give a mark to the other ants that this path is followed by greater number of ants so that the other ants will follow this path because ants prefer to follow a path rich in pheromone and they will put another pheromone on the path these steps will continue until all or most of the ants follow the same path which is the shortest. Figure 1 shows real ant behavior [18]. This behavior of this clever creatures can be used to show how they can find the shortest path from source food to nest after the sudden appearance of an unexpected obstacle has damage the initial path.

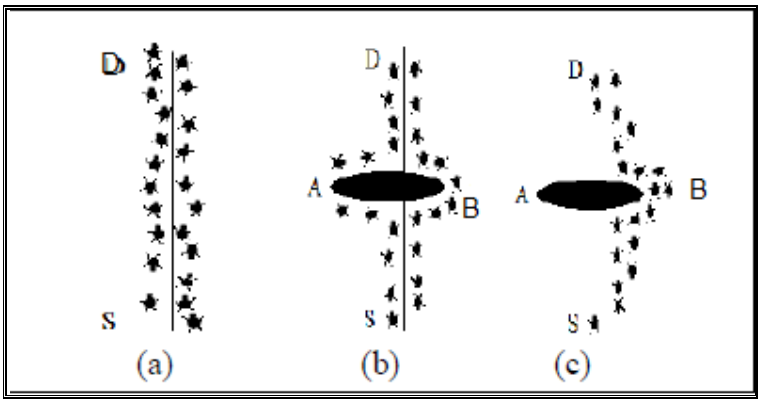

Figure 1. Ant behavior for finding shortest path.

In (Figure 1a) we see a set of ant that moves along a straight line from their nest S to a food source D.
In (Figure 1b). we see that the path is cut off by an obstacle into two sides (A) and (B) so the ants have to select between turning left or right and because no previous pheromone found in

both sides the selection is done with equal probability. The side (A) is longer than the side (B). Ants that follow the shorter path (SBD) will reach the food source before the ant that follow the longer path (SAD) so the amount of pheromone will increase in the shorter path .

Finally in (Figure 1c) All the ants have to choose the shorter path .
Figure 2 show the Pseudocode algorithm for ACO.

```
Begin
Initialize phoremone values
While (not termination condition ) do
    For k=1 to number of ants do
        construct a solution
        local pheromone update
    End For
    update pheromone for best solution
End While
End Begin
```

Figure 2. The Pseudocode algorithm for ACO

3. GENETIC ALGORITHM

Genetic Algorithm (GAs) has been used for many NP hard problems as a good search technique[9]. And also Genetic Algorithms are a part of evolutionary computing technique, which is speed the growing of the area of artificial intelligence.Genetic Algorithms are inspired by Darwin's theory about evolution, solution to a problem solved by genetic algorithms is devoloped[7-9]. The starting of the solutions using Genetic algorithm is begin with a set of initial solutions called population which represented by chromosomes . Solutions from one population are taken and used to give a new solutions (offspring) by apply crossover and mutation . This action is done by a hope, that the new population which is generated will be better than the old one. The way that is used to select the solutions that needed to form new solutions are selected according to their fitness, the good fitness are the more chances they have to reproduce.

The main advantage of GAs over the other methods is their parallelism. GAs travel in a search space that uses more individuals for the decision-making , so it has less chance to keep in a local extreme like the other available decision-making techniques. It is a population-based in which each of the individual population is developed in parallel and the optimal individual one is kept and come from the last set of population[17].

The summary of the Genetic Algorithm is show in figure 3[7].

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.
    1. **[Selection]** Select two parent chromosomes from a population according to their fitness (the better fitness , the bigger chance to be selected)
    2. **[Crossover]** With a crossover probability cross over the parents to form new offspring (children). If no crossover was performed, offspring is the exact copy of parents.
    3. **[Mutation]** With a mutation probability mutate new offspring at each locus (position in chromosome).
    4. **[Accepting]** Place new offspring in the new population
4. **[Replace]** Use new generated population for a further run of the algorithm
5. **[Test]** If the end condition is satisfied, **stop**, and return the best solution in current population
6. **[Loop]** Go to step **2**

Figure 3. Summary for Genetic Algorithm .

The most important steps for GA are Crossover and Mutation, but in our proposed algorithm we just use Crossover .

The figure 4 and figure 5 below show two types of Crossover:

- Two point Crossover
- Uniform Crossover

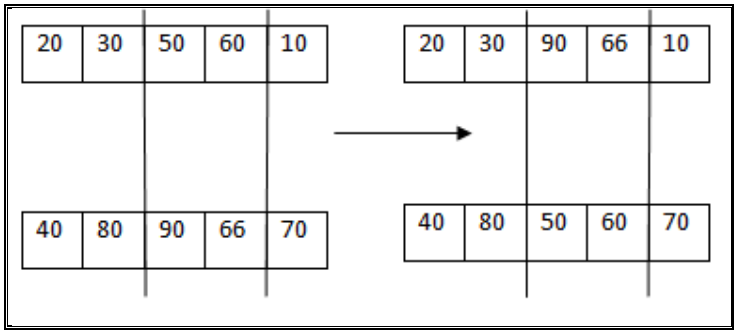

Figure 4. Two point Crossover

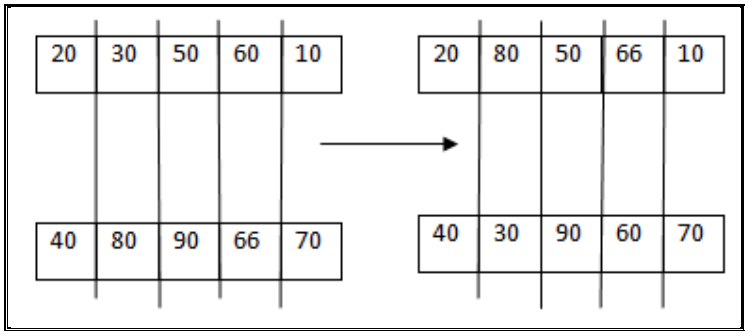

Figure 5. Uniform Crossover

4. RELATED WORK

There are number of systems developed for managing the execution of tasks on a network of machines. Examples include Condor, OSF DCE, and PBS [1]. Working with NP hard problems can often be developed by using local search methods to the solution found[14].

For solving task allocation problem in a heuristic techniques some algorithms were used[2], like Min-Min[11], the Fast Greedy[11], and Ant Colony Optimization [4].In Min-Min method, the task with minimum completion time is selected first and assigned to a processor. This algorithm has fast scheduling time, but poor load balancing [12][15]. Fast Greedy algorithm assigns each of the task to the processors in arbitrary order with the minimum completion time [12].

Meta heuristic Algorithm methods are fundamental part in methodical solutions which are have applicability in solving the problems of Hybrid Optimization .The development of the Mata-hubristic methods is in the early 1980's[3]. They have prove their success in solving a particular hybrid optimization problems. This group of methods consists of Ant Colony Optimization Algorithms , Genetic Algorithms, Memetic Algorithms , Problem-space search , Evolutionary Algorithms, Simulated Annealing, Tabu Search Algorithms and Hybrid Algorithms[3] .

5. PROBLEM STATEMENT

We have in our system a problem that consists of m tasks let we denote the set T={t1,t2,….tm} to be the set of these tasks and these tasks will be distributed on m processors to accomplish their jobs, let we say the set P={p1,p2,…pn} be the set of processors let m>n which mean that each processor will may have more than one task to execute depending on the execution time of the tasks and on the load already assigned to that processor.

All processors are homogeneous, which means that the execution time of a specific task is the same on any processor, one of these processors is selected to be the root and the jop of the root is to distribute the tasks in T to the processors in P after the processors complete their jobs (execute the tasks) the root will combine the executed tasks to give the final solution to the overall of the problem. By applying our algorithm we wish to minimize the makspane and the load balance.
To control the scheduling of tasks over the processors an upper bound value is used The value of the bound is calculated using formula 1 below. Where, m is the number of tasks to be scheduled, and n is the number of processors.

$$\text{Bound}=\sum_{j=1}^{m} tj/n \qquad (1)$$

In some cases there is a task that can't be delivered to any processor because the load of each processor in addition to the load of task exceeds the bound value, to deal with this case the bound can be increased during the scheduling processor with a constant value.

6. ANT COLONY OPTIMIZATION WITH GENETIC ALGORITHM FOR TASKS ALLOCATION

In our proposed solution for Task allocation problem we represent the problem as a complete graph G = (V, E) where the node (V) represents a task and the edge (E) represents a path between two tasks. Each edge in the graph is assigned a pheromone trial t at initial time the value of t will be 0.5 and this value is increased by a small amount depending on whether the ants follow this edge or not.

Our algorithm is perform as follow at first cycle each ant construct a candidate solution randomly and then the pheromone trials are updated with a value which depends on the solution constructed by the ants.

The solutions constructed by the ants in the prevoius step (parents)is used by genetic algorithm to produce new solutions (offspring) by applying cross over step, then the best solution is selected and recorded. The ants that has a better solution will have more chance to be selected.
This work will be continue until some conditions met like the maximum number of cycles. The Pseudocode of the complete algorithm TAP_ACO_GA is presented in Figure 6.

```
Algorithm: (TAP_ACO_GA)

Initializing data
While (k<=num_ant) do

    produce candidate solution randomly
    Update pheromone
    Save best solution
    Update pheromone for best solution
 End while

Repeat
 k=1
 While (k<=num_ant) do
        Construct solution as the following
          1. Choose the first task t randomly and assign it to a processor
          2. T=T-t
          3. Repeat
                Select next tasks depending on the property value p as
      given in
                 equation 2
              while (T !=θ)

          4. Update pheromone
          5. Save ants solutions (old solutions)
          6. Update pheromone for best solution
  End while
  Apply genetic technique
       1. Do crossover on the ants solutions to generate new solutions
```

2. Compare the old solutions with new solutions
3. Save best solutions
4. Update pheromone for best solution

Until (terminated condition is reached)

Figure 6. TAP_ACO_GA algorithm.

6.1 Pheromone Trials and Heuristic Information

At initilization , all the edges will have an equal amount of pheromone and during search for a soultion by the ants , the pheromone trial will be updated to a value depending on whether the constructed solution is good or not. Typically, the edges of the best solution will have the higher amount of pheromone update.

6.2 Constructing a Solution

In TAP_ACO_GA algorithm each ant starts from a randomly selected task, and then it selects the next task from those unselected tasks depending on the property value *p* as given in the next formula [13][5].

$$p_{a,c}^{k}(t) = \frac{t_{a,c}^{\alpha}(t) * h_{a,c}^{\beta}(t)}{\sum_{i \in u} t_{a,c}^{\alpha}(t) * h_{a,c}^{\beta}(t)} \tag{2}$$

where,

k denotes the ant number, $p_{a,c}^{k}$ is the probability with which ant *k* chooses to select task *a* with set of tasks in processor c, *t* denotes the iteration numbers, *u* denotes the set of task that are not visited yet, $t_{a,c}$ is the sum of pheromone value between task *a* and set of task in processor c, $h_{a,c}$ is the heuristic function which was chosen to be the inverse of the maximum difference of execution time between set of processor if we add task *a* to processor c at iteration *t*, α and β are parameters that control the pheromone trials and the heuristic information.The construction process is stopped when the maximum number of cycles is reached.

6.3 The Technique for Pheromone Update

The pheromone trails are updated on each edge after each ant has build a solution. The ant that builds the best solution (*bs*) will have the higher amount of pheromone update. In TAP_ACO_GA, pheromone is updated according to the following formula :

$$t_s(t+1) = \rho * t_s(t) + q/bv \tag{3}$$

Where,

t_s is the pheromone amount between tasks in schedule s at time *t*, t_s (*t*+1) is the amount of pheromone between tasks in schedule s at the next iteration, *bv* is the value assigned to the best solution, *q* and ρ is a given constant values.

The next formula shows the updated for pheromone trails for the other schedule :

$$t_j(t+1) = \rho * t_j(t) \quad (4)$$

Where,

t_j is the amount of pheromone between tasks in schedule j at time t, $t_j(t+1)$ is the pheromone amount between tasks in the schedule j at the next iteration, ρ is a given constant value.

The Pseudocode of the Pheromone Update is presented in figure 7.

```
If (solution<global solution) Then
  For every edge between two tasks in best solution do
        Apply equation ( 3)
Else
  For every edge between two tasks  do
        Apply equation ( 4)

End If
```

Figure 7. Pheromone Update.

Pheromone trials are updated according to *min_max* technique [16]. To avoid search stagnation situation if the relative differences of pheromone trails are too extreme.The Pseudocode of the Min_Max technique for Pheromone Update is presented in figure 8.

```
if pheromone(ti,tj)> Max pheremone then
    pheromone(ti,tj)= Max pheremone
if pheromone(ti,tj)< Min pheremone then
    pheromone(ti,tj)= Min pheremone
```

Figure 8. The Min_Max technique for Pheromone Update.

7. EXPERIMENTAL RESULTS

In order to validate the efficiency of the proposed method, several dataset for TAP problem is considered.The simulation program for the TAP_ACO_GA algorithm is performed on Intel Core 2 duo , 2.2 GHz machine with 2 GB RAM. The Table 1 below shows the computational workloads used in the simulation.

Table 1. Task work load parameters

Parameters	**Values**
Number of tasks	8-32 task
Execution time of tasks	10- 800 (seconds)
Number of Processors	4 homogeneous

In this study, we compared the performance of the proposed algorithm with the BTS_ACO algorithm. In all experiments, parameters are set to the values that are listed in Table 2.

Table 2. The Ant parameters values

α	**β**	ρ	**Max phoremone**	**Min phoremone**	**# of Ants**	**Q**
0.5	1	0.9	1	0.5	**4**	0.1

Some TAP dataset is used to compare the new algorithm with BTS_ACO.The comparison of Total Execution Time(sec) is shown in Table. 3 (the results of BTS_ACO are taken from the paper).

Table 3. A comparison between BTS_ACO and TAP_ACO_GA Total Execution Time(sec)

Number of tasks	**Total execution Time(sec)** BTS_ACO	**Average Execution time (sec)** BTS_ACO	**Total Execution Time(sec)** TAP_ACO_GA	**Average Execution time (sec)**TAP_ACO_GA
8	2165	541	1424	356
16	4565	1141	4520	1130
32	8144	2036	6535	1634

The figures (figure 9 and figure 10) below show the comparison of total and average execution time between the two algorithms as a chart.

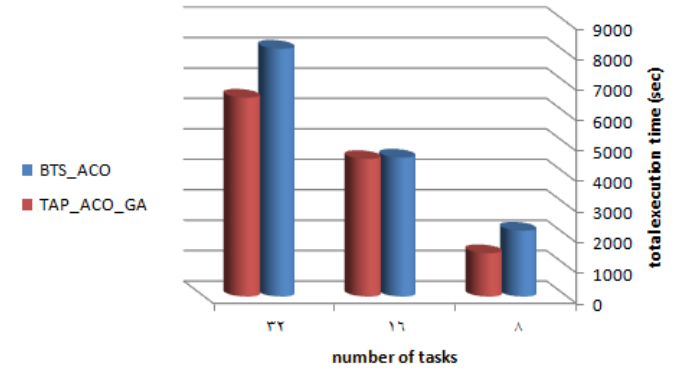

Figure 9. Total execution time for BTS_ACO & TAP_ACO_GA

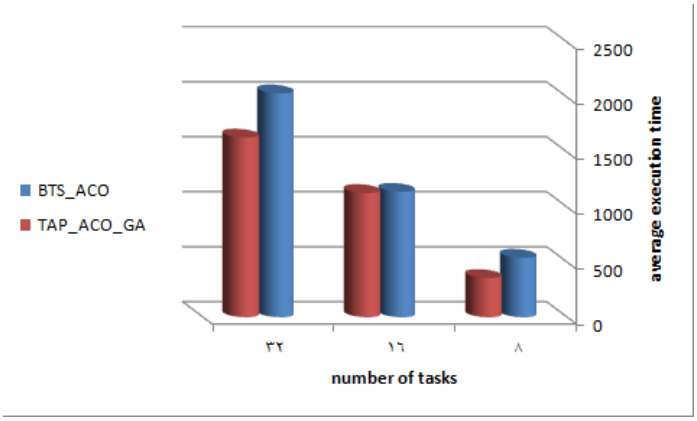

Figure 10. Average execution time for BTS_ACO & TAP_ACO_GA

Table 4 presents the comparison of better results obtained from solving the TAP problems acoording to makespane and load balance .

Table 4. A comparison of makespane and load balance

between BTS_ACO and TAP_ACO_GA

Number of tasks	**Makespan** BTS_ACO	**Makespan** TAP_ACO_GA	**Load balance** BTS_ACO	**Load balance** TAP_ACO_GA
8	800	600	470	190
16	1300	1220	445	145
32	2156	2290	776	514

The figures (figure 11 and figure 12) below show the comparison of makespan and load balance between the two approches as a chart .

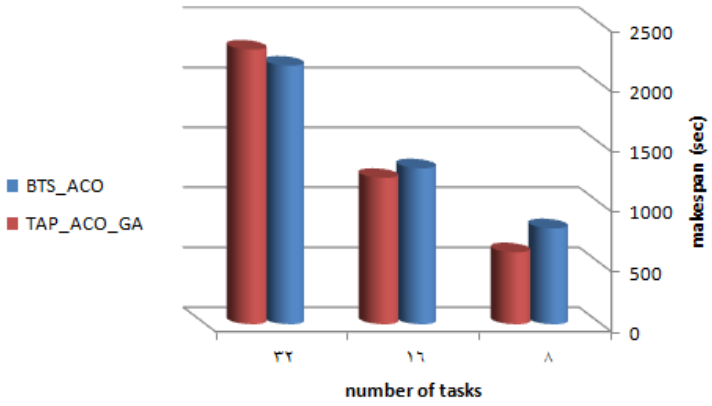

Figure 11. Makespan for BTS_ACO & TAP_ACO_GA

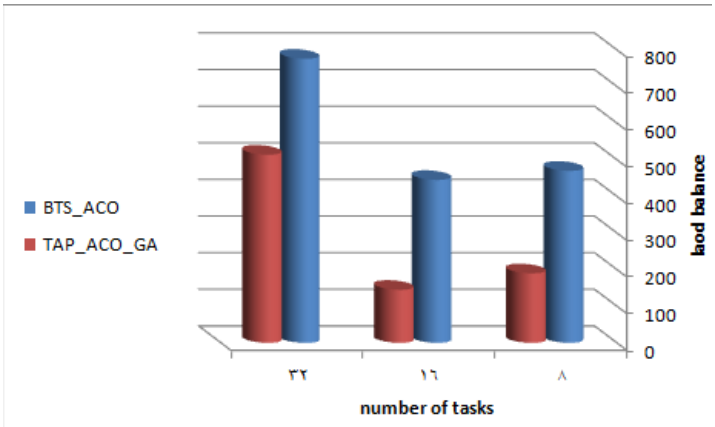

Figure 12. Load balance for BTS_ACO & TAP_ACO_GA

The results show that the proposed algorithm is better than the other BTS_ACO in terms of the ability to finding better solutions.

8. CONCLUSION AND FUTUREWORK

This paper applied ant colony optimization algorithm with genetic algorithm to solve task allocation problem. New algorithm TAP_ACO_GA is presented . Our proposed algorithm is based on the ideas of Ant Colony Optimization algorithm(ACO) and Genetic Algorithm (GA) , we test our algorithm in different dataset and compare our results with the results in[2] and the results show that TAP_ACO_GA algorithm is better than the other algorithm.

In the future we want to apply the idea of TAP_ACO_GA to solve task allocation on a network for Heterogeneous processors, and to apply the ACO with GA for solving other heuristic search problems .

ACKNOWLEDGEMENTS

The authors of this paper would like to thank the editor and the reviewer for their good comments and constructive suggestions which help to improve the level of the paper.

REFERENCES

[1] R. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M.Halderman, D. Hensgen, E. Keith, T. Kidd,
M. Kussow, J.Lima, F. Mirabile, L. Moore, B. Rust, and H. Siegel(1998) “Scheduling resources in multi-user, heterogeneous,computing environments with SmartNet,” *in 7th IEEEHeterogeneous Computing Workshop*, pp 184–199.

[2] B. Al-Sharaa and T. Al-Qublan,(2013) "Bounded ant colony algorithm for task allocation on anetwork of homogeneous processors using a primary site (BTS-ACO)", *International Journal of Computer Science & Information Technology (IJCSIT)* Vol. 5, No. 3, pp 165-173.

[3] S. Alhamdy, A. Noudehi,and M. Majdara,(2012) " Solving Traveling Salesman Problem (TSP) using Ants Colony (ACO) Algorithm and comparing with Tabu Search, Simulated Annealing and Genetic Algorithm ", *Journal of Applied Sciences Research*,Vol. 8,No. 1,pp 434-440.

[4] Z. Hlaing and, M. Khine,(2011) "An Ant Colony Optimization Algorithm for Solving Traveling Salesman Problem ", *International Conference on Information Communication and Management IPCSIT* vol. 16,pp 54-59.

[5] Z. Hlaing and M. Khine, (2011) "Solving Traveling Salesman Problem by Using Improved Ant Colony Optimization Algorithm ",*International Journal of Information and Education Technology*, Vol. 1, No. 5,pp 404-409.

[6] I. Jr,Z. Čičková,(2011) "Solving the Travelling Salesman Problem Using the Ant Colony Optimization", *Management Information Systems*,Vol. 6 , No. 4,pp 010-014

[7] M. Alhanjouri and B. Alfarra ,(2013) " Ant Colony versus Genetic Algorithm based on Travelling Salesman Problem ", *Int. J. Comp. Tech. Appl*., Vol. 2,No. 3,pp 570-578

[8] P. BAJPAI and DR. M. KUMAR,(2010)" Genetic Algorithm – an Approach to Solve Global Optimization Problems ", *Indian Journal of Computer Science and Engineering* ,Vol. 1, No. 3,pp 199-206

[9] F. Khan, N. Khan, S.Inayatulla,And Sh.Nizami,(2009) "Solving ISP Problem by Using Genetic Algorithm " , *International Journal of Basic &Applied Sciences IJBAS-IJENS* Vol.09 No.10 , pp55-60.

[10] An Experimental Study1 and T. Hagerup2,(1997) "Allocating Independent Tasks to Parallel Processors ", *JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING* ,Vol. 47,No.2 ,pp 185–197.

[11] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I.Reuther, J. P. Robertson, M.D. Theys, B. Yao, D. Hensgen and R. F.Freund, (2001) "A Comparison of Eleven Static Heuristicsfor mappinga Class of Independent Tasks onto Heterogeneous DistributedComputing Systems",*Journal of Parallel and Distributed Computing*,Vol. 61, No. 6, pp 810-837.

[12] R. Armstrong, D. Hensgen, and T. Kidd,(1998) "The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions," *in 7th IEEE Heterogeneous Computing workshop,* pp 79–87.

[13] T.Qablan, Q Al-Radaideh, S. Abu Shuqeir,(2012) " A Reduct Computation Approach Based on Ant Colony Optimization",*ABHATH AL-YARMOUK*: "Basic Sci. & Eng." Vol. 21, No. 1, 2012, pp 29-40.

[14] G. Ritchie and J. Levine, (2003) "A fast, effective local search for schedulingindependent jobs in heterogeneous computing environments". Proceedings of the 22nd Workshop of the UK Planning and Scheduling Special Interest Group.

[15] R. Freund, M. Gherrity, S. Ambrosius, M. Campbell, M.Halderman, D. Hensgen, E. Keith, T. Kidd,M. Kussow, J.Lima, F. Mirabile, L. Moore, B. Rust, and H. Siegel(1998) "Scheduling resources inmulti-user, heterogeneous,computing environments with SmartNet," *in 7th IEEEHeterogeneous Computing Workshop,* pp 184–199.

[16] Stützle T. and Hoos H.(2000) "MAX-MIN ant system". *Future Generation ComputerSystems*.Vol. 16 ,No. 8,pp 889–914.

[17] Md. K. Hossain and A. El-Saleh, (2013) "cognitive radio engine model utilizing soft fusion based genetic algorithm for cooperative spectrum optimization", *International Journal of Computer Networks & Communications (IJCNC),* Vol.5, No.2, pp 23-36 .

[18] D. Sensarma and K. Majumder, (2013) "An Efficient Ant Based QOS Aware Intelligent Temporally Ordered Routing Algorithm For MANETS", *International Journal of Computer Networks & Communications (IJCNC),* Vol.5, No.4, pp 189-203 .