VOL. 16, NO. 10, MAY 2021 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

GeneTaS - AN OPTIMIZED TASK SCHEDULING STRATEGY USING GENETIC ALGORITHM FOR PARALLEL AND DISTRIBUTED COMPUTING ENVIRONMENT

P. Muthulakshmi[1], D.I. George Amalarethinam[2] and P. Yogalakshmi[1]
[1]Department of Computer Science, SRM Institute of Science and Technology, Kattankulathur Campus, Chennai, Tamil Nadu, India
[2]Computer Science and Director (MCA), Jamal Mohamed College, Trichirappalli, Tamil Nadu, India
E-Mail: muthulap@srmist.edu.in

ABSTRACT

The proposed genetic algorithm could solve the problem of task scheduling with a new initialization strategy to generate the initial population and new genetic operators to ensure best task–resource mapping that preserves good characteristics of the found solutions. Genetic Algorithm for Task Scheduling (GeneTaS) uses bio-inspired genetic algorithm to find an optimal schedule and adapts new fitness function to find the suitability of task and resource pair for best allocation. The use of evolutionary operators; crossover and mutation are found to move the solution very close towards optimality. The proposed algorithm is implemented using Gridsim, a simulator for task allocation problems and tested with arbitrary task graphs that are generated using DAGitizer. In the experimental setups, thousands of arbitrary task graphs are used and it is observed that the results of the proposed GeneTaS algorithm is found better than the compared scheduling algorithms, when scaled on performance metrices namely; makespan, resource utilization and speed up.

Keywords: parallel and distributed computing, genetic algorithm, fitness function, cross over, mutation, makespan, resource utilization, speed up.

INTRODUCTION

Parallel and distributed computing facilitates the utilization of computational power of globally distributed resources. Also, it combines the utilities like infrastructure and software in order to provide an extensive support to the demand of computational power, storage and application services. Applications involve large complex procedures need lot of computational power to execute. Super computers that may have extra ordinary computation speed to complete the executions in short span of executing time may fulfil these demands. However, owning and maintaining super computers are very expensive and require specific geographic regions to install. Many surveys state that the CPU cycles of computers on the internet are remain unused. These idle computing powers can be used to meet the demand of large-scale applications that are used to solve problems in the field of medical science, astronomy, aeronautics, engineering, and commerce. Parallel and distributed computing technology aggregates the globally found resources and allows people to use them on agreements and policies. A business environment may support buying and selling of computing speed, storage spaces, application services. This can be achieved by the parallel and distributed commerce environment where computing power, applications, and devices are traded as commodities. The performance of a parallel and distributed system largely depends on the effective utilization of resources. Best allocation methods help in assigning large-scale business, scientific applications to the available resources for execution and it results in remarkably great performance. These allocation methods are known as scheduling algorithms. In general, scheduling algorithms focus to minimize the overall time required to complete the execution of applications.

This study proposes an evolutionary based scheduling algorithm called Genetic Algorithm for Task Scheduling in parallel and distributed computing environment (GeneTaS) to schedule a group of tasks on the available resources. Genetic algorithm (GA) is an adaptive heuristic random search algorithm based on the evolutionary ideas of natural selection and genetics. GA was proposed by Holland [10]. GAs help to get optimal solutions and found to outperform several algorithms in the task scheduling problems [11, 12]. Also, it is found from the literature survey [13 through 27] [30 through 35] that GA provides best solutions for problems.

Generally, bio-inspired algorithms are optimization algorithms that may convert the feasible solutions into optimal solutions. These algorithms take a set of solutions as initial population. In GA, each solution is represented as a chromosome. The genetic operators namely, selection, crossover, and mutation are used to transform chromosomes in to better chromosomes that forms a better population. To retain good features from the previous generation, the crossover operator exchanges the information between two chromosomes. The mutation operator in a random fashion alters single bit of a chromosome to increase the diversity of chromosomes and hence, prevents the premature convergence of solutions. The genetic process is repeated until the stopping criterion is satisfied. At the end of process, new set of chromosomes will be generated as population that could be considered as new generation. The best chromosomes from all generations are reported to be the best solutions.

The proposed methodology employs genetic algorithm to discover an optimal schedule. The study presents a new initialization strategy to generate the initial population and new genetic operators. GeneTaS algorithm adapts new fitness function that finds the best resource for

a task to execute. Also, GeneTaS algorithm uses the evolutionary operators; crossover and mutation, which can move the solution very close to optimality. The proposed algorithm is implemented using Gridsim Simulator and tested with wide range of arbitrary task graphs that are generated using DAGitizer [29], a simulated environment to generate random task graphs of size ranging from 3 to 1550 tasks. The results achieved through various testing in the experimental set ups show that the proposed GeneTaS algorithm has outperformed the compared Hybrid Genetic Algorithm (HGA) [13] and Triplet Bin Task Grouping and Prioritizing algorithm (TBTGP) [28] scheduling algorithms.

The Proposed Genetic Algorithm for Task Scheduling

The present study incorporates genetic algorithm to schedule applications on parallel and distributed. In the present study, GA is used to schedule workflow/DAG model of applications. In general, workflow preserves precedence constraints. GeneTaS algorithm focuses on minimizing the makespan and communication cost while maximizing the utilization of resources by balancing the load across parallel and distributed resources. Genetic Algorithms in the literature have not ensured the validity of both the initial population and the offsprings generated by genetic operators. GeneTaS overcomes these difficulties by adding the feasible solutions generated by TBTGP algorithm to the initial population. The string representation of GeneTaS involves the schedules of tasks on individual resources, Strings that preserve the precedence constraints are absorbed as valid strings. GeneTaS ensures not only the validity of the initial population but also guarantees the validity of offspring generated by genetic operators. All the strings generated by GeneTaS represent valid executable schedules. Table-1 presents the pseudocode of GeneTaS algorithm.

Table-1. GeneTaS algorithm.

Algorithm Genetic Algorithm for Task Scheduling (DAG, Resource List)
1. {
2. Generate the chromosome population (popsize) through TBTGP algorithm and through random generation
3. for chro= 1 to popsize do
4. Find the validity through fitness function f(chro)
5. while (not termination condition)
6. {
7. Select the chromosomes from the population having high fitness values
8. Apply single point cross over operation on the parents to generate two new offsprings
9. Mutate new offspring at each position in chromosome; reorder and restore
10. Check the new offspring for its validity
11. Valid chromosomes are accepted for producing new generation
12. }
13. if (best solution is obtained from the current generation)
14. return(the best solution)
15. else
16. goto step 2
17. }

Chromosome representation of GeneTaS

Proper representation of chromosome is the first step in GA. The chromosome may be known as string. A string is a candidate solution in GA and therefore, it should represent a complete schedule. The representation of a string involves the parameters available in the solution. It is found that the value encoding method is suitable for scheduling problem. The solution sequence consists of task identification number and resource identification number and they are represented as integer values. The assignment of task 'm' on resource 'r' for execution can be shown as (m,r). Valid schedules are considered for representing the strings and these strings form the initial population. Figure-1 shows a simple workflow application. Table-2 consists of computation costs of tasks represented through circles numbered from 1 to 8 and the communication costs of the edges connecting the tasks are shown on the edge links of Figure-1.

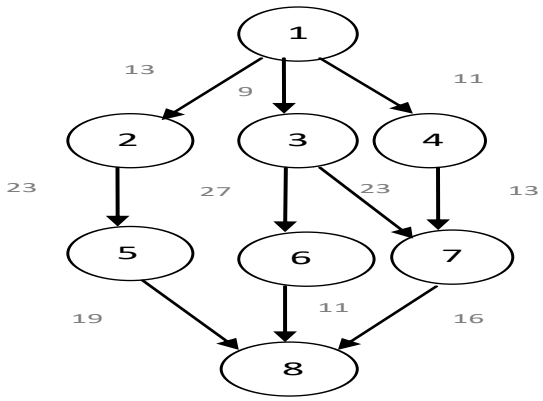
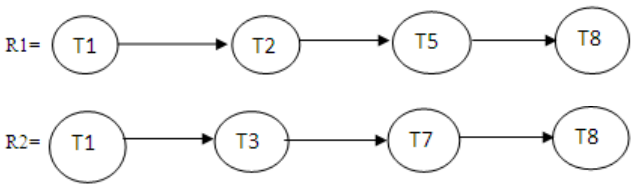


Figure-1. Simple DAG.

Table-2. Computation cost of tasks on resources.

Computation Cost		
Tasks	**R1**	**R2**
T1	14	16
T2	13	19
T3	11	13
T4	13	8
T5	12	13
T6	13	16
T7	7	15
T8	5	11

Valid execution sequence for the DAG illustrated in Figure-1 that uses computational cost given in Table-2 can be shown as,



Each sequence of computational tasks will be executed on the respective resource and the order of tasks in the sequence indicates the order of execution. The schedule will be encoded as (T1, R1), (T2, R1), (T3, R1), (T4, R2), (T5, R1), (T6, R1), (T7, R1), (T8, R1).

Initial Population of GeneTaS

The size of the population is fixed to be 20. Half of the population capacity is filled using schedules generated using TBTGP algorithm and the remaining part would be filled by random generation of task, resource pair. The validity of initial populations is checked using fitness function. The initial population of GeneTaS algorithm is generated using the pseudocode shown in Table-3.

Table-3. GeneTaS population.

```
Algorithm GeneTaS_population
1.  for i = 1 to 20 do
2.  {
3.  if( i <=10)
4.  use TBTGP algorithm to generate schedule(i)
5.  else
6.  generate random valid schedule(i)
7.  }
8.  for i = 1 to 20
9.  encoded_Strings_List=Encode_Schedule(schedule(i))
10. initial_population_List = encoded_Strings_List
```

The proposed Fitness function of GeneTaS

A fitness function is used to scale the quality of an individuals in the population according to the given optimization objective. The fitness function is observed to be different for different problems. Considering scheduling problems, the fitness function can be defined based on deadline or budget constraints. Fitness function in GA involves the factors that influence the objective function of the problem. The proposed fitness function is designed to emphasize minimum makespan, maximize resource utilization by balancing load across resources and reservation of appropriate number of efficient resources. In the fitness function definition, makespan is the minimizing function whereas resource utilization is the maximizing function and reservation of resources depends on maximum number of tasks that is contained in the level of DAG. The fitness function of the string is evaluated by,

$$F(i)=\{\max(FT(ti,rj))/\max(\text{sum}(FT(ti,rj)))\}*(vs/ts) \quad (1)$$

Where FT represents the finish time, vs and ts denotes valid sequences and total sequences respectively. The fitness function is used to evaluate the fitness value of strings. Based on the fitness value, roulette wheel mechanism selects the best strings.

Roulette wheel selection of GeneTaS

The reproduction process forms a new population by selecting strings form the old population based on their fitness values. The selection criterion focuses to select strings with higher fitness value and given priorities to survive to the next generation. Roulette wheel selection method is used to implement reproduction in GeneTaS algorithm. This selection method is adopting the methodology that was proposed by Kalyanmoy Deb [36]. The Roulette wheel selection needs six computations to select the chromosome that could form the initial population. The fitness value of each population (F(i)) is calculated and arrived at the average fitness function ($\overline{F}$) followed by the computation of the expected count of each string A.

$$A(i)=F(i)/(\overline{F}) \quad (2)$$

The probability (PB) of each string is the ratio between the expected count A and the population size N. The probability of the string is given as,

$$PB(i)=A/N \quad (3)$$

The cumulative probability is calculated using the probability ratio of each string. The value of cumulative probability of a string is expected to be one in order to get it selected for reproduction. The cumulative probability is expressed as follows,

$$C(i)=C(i)+PB(i) \quad (4)$$

To select the strings for next generation, the following steps will be followed: first, a random number, ‘rs’ is generated between 0 and 1, then the string that has the very close cumulative value with respect to ‘rs’is

chosen, the steps are repeated when the population strength is less than the population size. The selected strings found in the mating pool will be allowed to undergo crossover operation to produce new offsprings.

Single point crossover operation of GeneTaS

The crossover operator facilitates the utility of producing new offsprings from the existing population of strings. This study uses single point crossover to generate offspring for the next generation and the following steps are used to carry out the process,

a) Select a random resource R1 from the resource list.
b) Select a random task scheduled on the resource selected in step 1
c) Select another random resource R2 form the resource list (resources selected in steps 1 and 3 should not be the same)
d) Randomly select a task that is scheduled to execute on the resource selected in step 3 (selection of tasks in steps 2 and 4 should not violate the precedence constrains)
e) Exchange the sequence of tasks of R1 that comes after the tasks selected in step 2 with the sequence of tasks of R2 that comes after the tasks selected in step 4.
f) Now save the arrangement as offspring 1 and offspring 2 obtained from steps 3 and 5 respectively.

The probability of crossover rate is 0.65. In addition, offsprings are created by exchanging the portions of a selected string by preserving the precedence constraints and the new offsprings will be validated for their survival. The remaining invalid strings will be allowed to mutate.

The Mutation operation of GeneTaS

Mutation is an occasional random alteration of the value of a string. The invalid strings undergo mutation operation to transform them into valid strings. Some of the ideas that could be used while mutation operator is applied: (1) Swapping the sequence of tasks between two randomly chosen resources, (2) swapping the positions of tasks in a sequence and (3) swapping the task that is scheduled to execute after a long idle time to one of the schedules that falls before the long idle time. Similarly, if the invalid string does not satisfy the precedence constraints, repeated swapping of two tasks can be done to meet the precedence constraint. GeneTaS algorithm incorporates one or single point mutation. The probability rate of mutation is 0.01. The following procedure helps to adopt mutation in an invalid string to convert the same into valid string.

GeneTaS algorithm allows 50 iterations in order to meet the convergence condition but it achieves favorable conditions between 25 and 35 iterations itself.

RESULTS AND DISCUSSIONS

The proposed Genetic Algorithm for Task Scheduling is implemented using Gridsim. Simulation based on GridSim uses the following parameter values,

- Resources: 1-15
- Processing Elements (PE): 1-10
- PE rate (MIPS): 5-50
- Gridlets: 512
- Gridlet length: 5k-10k
- Input file/output file size: 50-300 MB

The study underwent exhaustive experiments with arbitrary task graphs to analyze its performance. The results are compared with the results of TBTGP, which is a list scheduling algorithm and HGA, an evolutionary algorithm. It is analyzed that HG algorithms is found to design with complex crossover operation; also it is found unsuitable for large graphs and graphs with small Computation Communication CCR values. The results of the proposed algorithm are found to be better than the compared algorithm. When comparing the algorithms to scale the performances, it is observed that GeneTaS algorithm outperforms the other algorithms by meeting the objective of the study. GeneTaS is examined to be the best in makespan, resource utilization and in speed up values. Sample arbitrary task graph information is shown in Figure-2. Figure-3 shows the pictorial view of the generated task graph information for the inputs given via the user interface. Summary of the generated DAG is displayed in Figure-4. Figure-5 illustrates the plots of schedules and the makespan of GeneTaS and HG algorithms. The investigational results on makespan, average resource utilization and speed-up ratios for varying experimental setups are shown in Table-4 through Table-6. Correspondingly, Figure-6 through Figure-8 show the Gantt chart for the values of Table-4 through Table-6. The proposed GeneTaS algorithm gives better result consistencies for all kinds of graphs with different size, topology and CCR values. For instance, the arbitrary task graph of size 100 with CCR = 1.2 scheduled on 8 resources results in a makespan value of 10529 (msec.), 9560 (msec.) and 8553 (msec.) for TBTGP, HGA and GeneTaS algorithms respectively. Similarly, the DAG size of 500 tasks executed on 64 resources shows the makespan values as 36521 (msec.), 35657 (msec.) and 34294 (msec.) for TBTGP, HGA and GeneTaS algorithms respectively. The values in the Table-4 are computed for various input parameters and the analysis concluded that GeneTaS algorithm proves itself by showing remarkable execution times for arbitrary task graphs ranging from 50 to 1000.

Similarly, the arbitrary task graph of 250 tasks with CCR=0.6 executed on 32 resources shows better resource utilization for GeneTaS than the compared algorithms and the results are 65.62, 66.49 and 85.29 for TBTGP, HGA and GeneTaS algorithms respectively. It is observed that GATs algorithm shows more than 70 percentage of resource utilization for both computational and communicational intensive graphs whereas the compared algorithms are found be less in utilization for

the same input values. With respect to speed-up values, GeneTaS algorithm gets almost close to one for most of the runs with different parameter values. The consistency of results produced by GeneTaS with the change in parameter values proves that the proposed algorithm is very efficient and effective. For example, the speed-up values of TBTGP, HGA and GeneTaS are found to be 0.6927, 0.7381 and 0.9226 respectively while executing a task graph of size 100 with CCR value as 2.1 on 16 resources. Similarly, it can be noted that the speed-up values of GeneTaS is greater than the compared algorithms for various input specifications. Hence, it can be confirmed that GeneTaS algorithm is highly productive in nature for any kind of input values, whereas HGA algorithm is confined to certain limitations to prove its efficiency and TBTGP algorithm is a not an evolutionary algorithm however, it proves its performance to a better level.

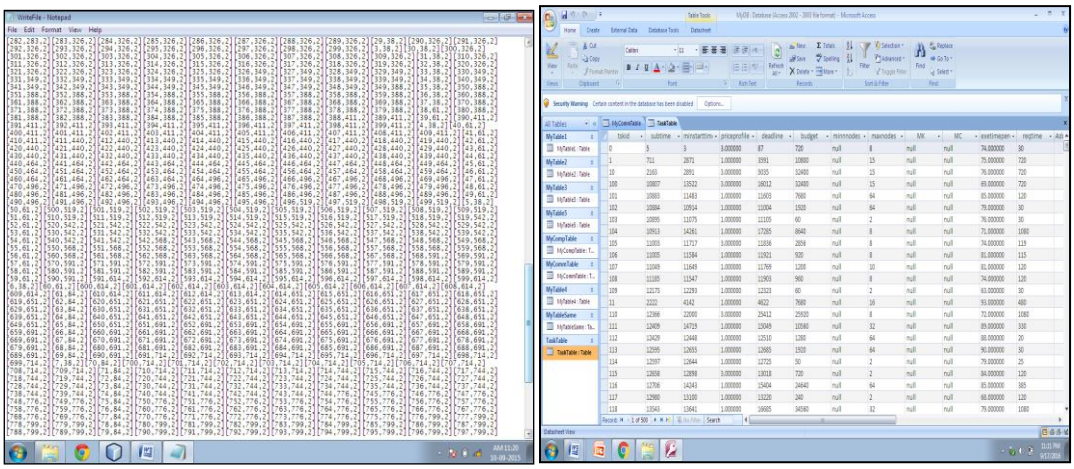


Figure-2. Arbitrary task graph information.

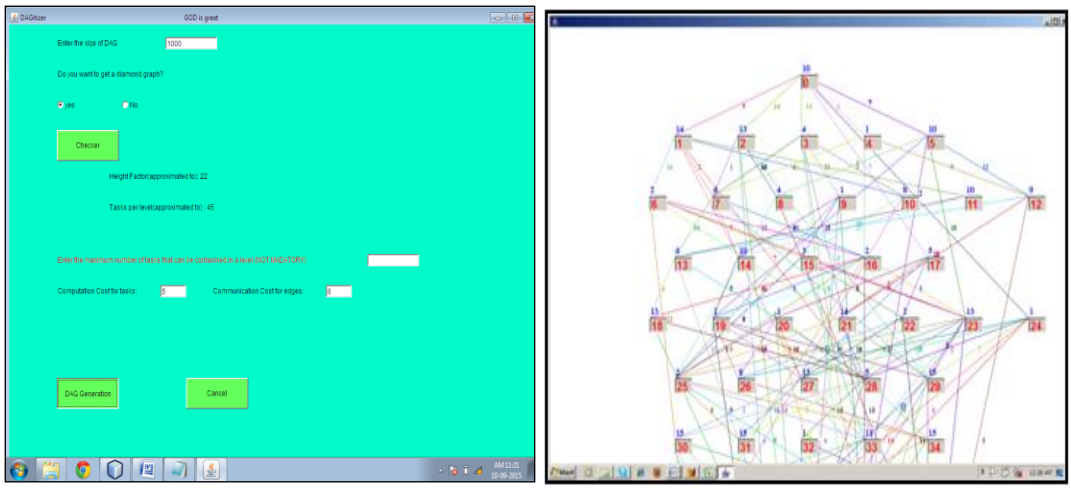


Figure-3. User Interface for input values to generate DAG generated DAG.

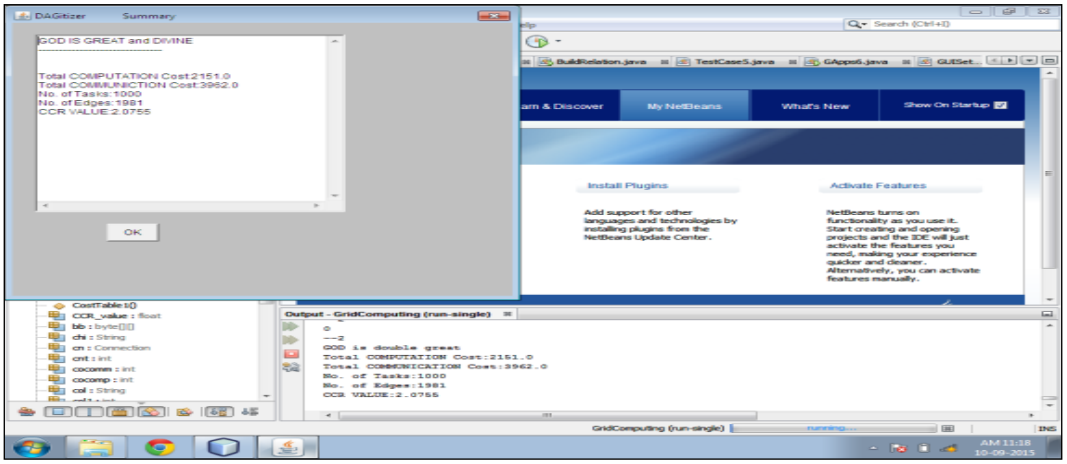


Figure-4. Summary of the generated DAG with CCR value.

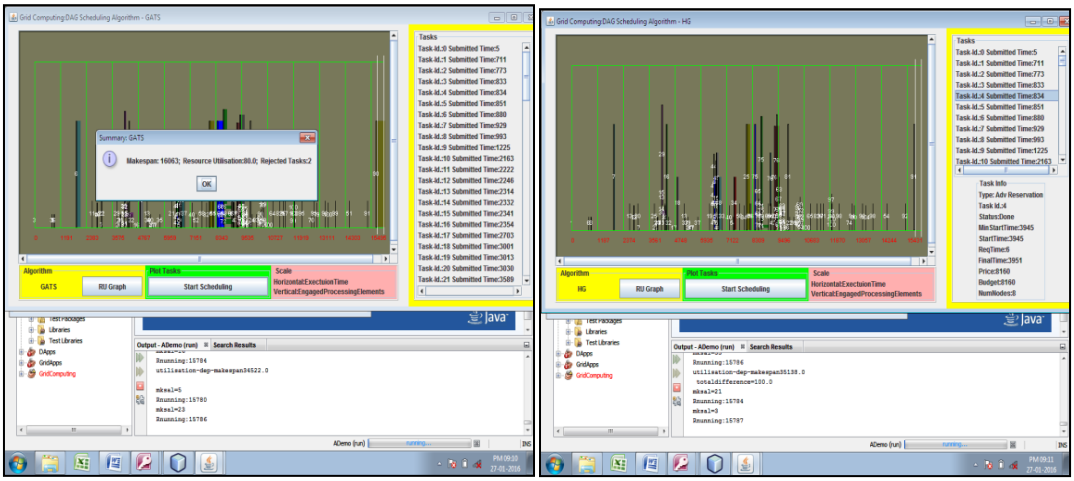


Figure-5. Schedule information generated for GeneTaS and HG algorithms.

Table-4. Makespan of TBTGP, HGA and GeneTaS algorithms for arbitrary task graphs.

No. of Tasks	Resources	Algorithms								
		TBTGP	HGA	GeneTaS	TBTGP	HGA	GeneTaS	TBTGP	HGA	GeneTaS
		Makepan (msec.) for Arbitrary Task Graph of CCR=0.6			Makespan (msec.) for Arbitrary Task Graph of CCR=1.2			Makespan (msec.) for Arbitrary Task Graph of CCR=2.1		
50	4	6645	5741	5171	8565	7786	5925	11393	10319	8852
	8	4072	3653	3267	7691	6876	5571	14327	12995	11716
	16	3445	2637	2332	4417	3786	2501	8772	7671	6585
500	32	31937	31386	30878	33505	32701	31313	36521	35657	34294
	64	20959	19324	18267	22671	21768	19666	24861	23970	23020
	128	14884	14179	12871	16069	15566	12751	20584	19538	18217
1000	64	44176	42457	39983	46048	44971	42360	48645	47714	45873
	128	35316	34276	32086	38777	37557	34943	41797	40953	38623
	256	25889	24865	23007	28374	27393	24821	31814	31420	29490

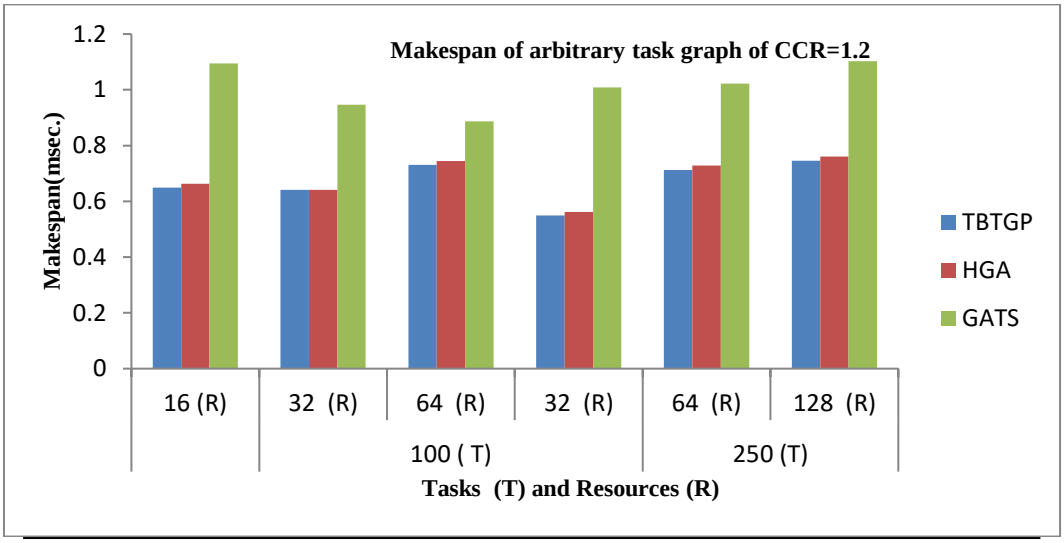


Figure-6. Makespan of TBTGP, HGA and GeneTaS algorithms.

Table-5. Average Resource Utilization (ARU) of TBTGP, HGA and GeneTaS algorithms for arbitrary task graphs.

No. of Tasks	Resources	Algorithms								
		TBTGP	HGA	GeneTaS	TBTGP	HGA	GeneTaS	TBTGP	HGA	GeneTaS
		ARU (%) for Arbitrary Task Graph of CCR=0.6			ARU (%) for Arbitrary Task Graph of CCR=1.2			ARU (%) for Arbitrary Task Graph of CCR=2.1		
250	16	63.45	67.41	84.79	69.73	70.36	72.41	69.45	72.30	75.24
	32	65.62	66.49	85.29	69.96	72.03	77.94	67.01	71.68	70.97
	64	65.95	75.38	73.33	69.47	71.36	70.08	68.06	68.23	76.21
500	32	71.20	75.26	72.34	70.28	68.81	80.05	66.72	66.56	77.89
	64	65.79	74.44	74.30	69.99	72.05	82.19	69.71	68.75	75.48
	128	65.06	67.09	75.97	65.72	70.31	77.38	67.66	72.33	76.85
1000	64	71.59	68.15	74.24	65.78	67.76	80.14	65.58	67.58	74.38
	128	69.67	71.59	83.06	66.34	69.53	83.22	66.97	66.54	74.92
	256	73.54	72.55	81.11	69.47	73.04	73.21	65.05	70.46	73.33

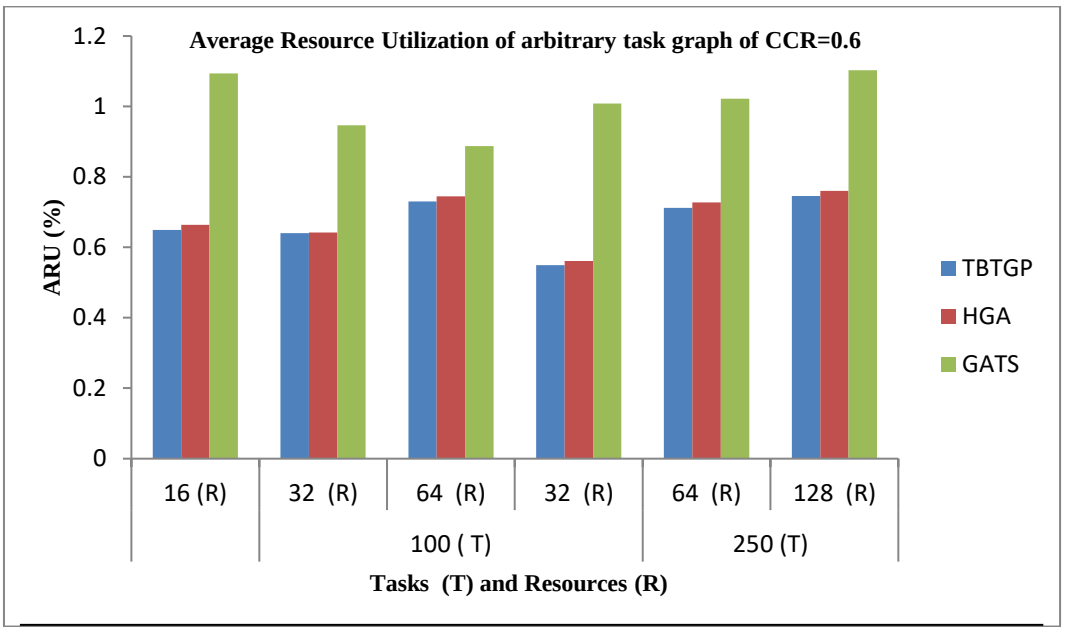


Figure-7. Average Resource Utilization of TBTGP, HGA and GeneTaS algorithms.

Table-6. Speed-up values of TBTGP, HGA and GeneTaS algorithms for arbitrary task graphs.

No. of Tasks	Resources	Algorithms								
		TBTGP	HGA	GeneTaS	TBTGP	HGA	GeneTaS	TBTGP	HGA	GeneTaS
		Speed-up Ratio for Arbitrary Task Graph of CCR=0.6			Speed-up Ratio for Arbitrary Task Graph of CCR=1.2			Speed-up Ratio for Arbitrary Task Graph of CCR=2.1		
100	8	0.5760	0.5797	0.8607	0.6495	0.6635	1.0941	0.6453	0.7135	0.9900
	16	0.5856	0.5961	0.5979	0.6407	0.6417	0.9462	0.6927	0.7381	0.9226
	32	0.5906	0.6573	1.0093	0.7304	0.7450	0.8871	0.6565	0.6521	0.7800
250	16	0.6152	0.6639	0.9261	0.5494	0.5614	1.0088	0.6956	0.7036	0.9428
	32	0.5284	0.5916	1.4968	0.7120	0.7279	1.0220	0.6890	0.7459	0.9649
	64	0.5821	0.6187	1.5721	0.7461	0.7607	1.1028	0.7149	0.7152	0.8103
500	32	0.5909	0.6410	1.1485	0.6293	0.6619	0.9518	0.7241	0.7271	0.8073
	64	0.5922	0.6825	0.7875	0.6453	0.6501	0.7241	0.7380	0.7565	0.9376
	128	0.6191	0.6633	1.0044	0.6528	0.6588	0.7699	0.7150	0.7083	0.7855

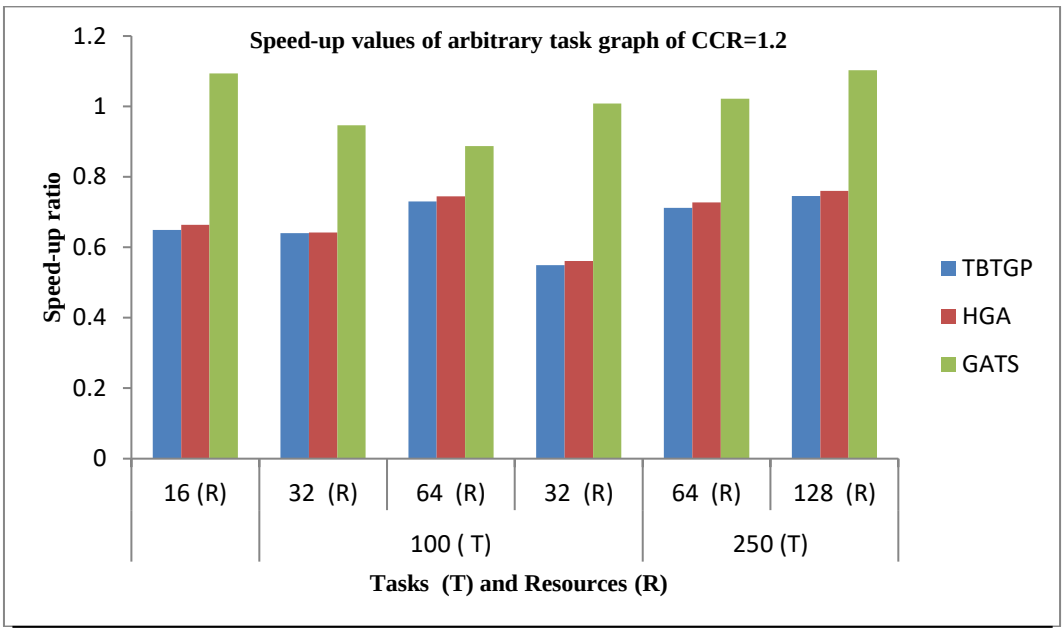


Figure-8. Speed-up values of TBTGP, HGA and GeneTaS algorithms.

CONCLUSIONS

In this paper, a nature-inspired algorithm called Genetic Algorithm for Task Scheduling is proposed to schedule tasks in parallel and distributed environment. It is designed to give optimal schedules with the available resources. Genetic Algorithm is an evolutionary approach to derive optimal solution from the existing solutions. Hence, the initial population consist of individuals (schedules) from TBTGP algorithm a list scheduling algorithm is used for the evolution of generations to produce optimized schedules. Randomness is found to favor most of the situations and therefore a partial population is filled with random combination of tasks and resources. The random combinations are ensured to be valid by preserving the precedence constraints. The genetic operators are applied suitably on strings to get optimized results. After a considerable number of iterations, the results are found to converge as optimum. In this methodology, a new fitness function has been proposed to validate the strings that can be considered as an individual in the population for next successive generations. The fitness function is designed to focus on both the task and resource. According to experimental results conducted on different types of task graphs with varying range of CCR values and various kinds of resource selections, GeneTaS gives consistent performance with lower makespan and higher resource utilization. In order to schedule the regular graphs, we consider the system of simultaneous equations viz., Gauss Elimination Method, Laplace Method and Fast Fourier Transformation Method. In the proposed work, we focus on makespan, resource utilization and load balancing, other performance metrics can also be considered that include flow time, throughput, memory specification, contention awareness, etc. to explore the hidden potential of the parallel and distributed system. While applying optimization techniques the initial population could be generated using the combined strategies of critical tasks and duplication mechanism. Making use of rescheduling point in the pre-emptive mode of execution can be considered in the future works.

REFERENCES

[1] Haluk Topcuoglu, Salim Hariri, Min-You Wu. 2002. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Transactions on Parallel and Distributed Systems.

[2] Liou J, Palis M A. 1996. An Efficient Task Clustering Heuristic for Scheduling DAGs on Multiprocessors. Proceeding of workshop on resource management, symposium of parallel and distributed processing. pp. 152-156.

[3] Tang X, Li K, Liao G, Li R. 2010. List Scheduling with Duplication for Heterogeneous Computing Systems. Journal of Parallel Distributed Computing. 70(4): 323-329.

[4] Omara F A, Arafa M M. 2010. Genetic Algorithms for Task Scheduling Problem. Journal of Parallel Distributed Computing. 70(1): 13-22.

[5] Rewini HE, Lewis T, Ali H. 1994. Task Scheduling in Parallel and Distributed Systems. Prentice Hall, New Jersey.

[6] Hyunjin K, Sungho K. 2010. Communication-Aware Task Scheduling and Voltage Selection for Total Energy Minimization in a Multiprocessor System

using Ant Colony Optimization. Information Science. 181(18): 3995-4008.

[7] Mirabi M. 2011. Ant Colony Optimization Technique for the Sequence-Dependent Flowshop Scheduling Problem. International Journal of Advanced Manufacturing Technology. 55(1-4): 317-326.

[8] Liu H, Abraham A, Snášel V, McLoone S. 2012. Swarm Scheduling Approaches for Work-Flow Applications with Security Constraints in Distributed Data-Intensive Computing Environments. Information Science. 192: 228-243.

[9] Tao Q, Chang H Y, Yi Y, Gu C Q, Li W J. 2011. A Rotary Chaotic PSO Algorithm for Trustworthy Scheduling of a Grid Workflow. Computing and Operational Research. 38(5): 824-836.

[10] Holland J H. 2011. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

[11] Falzon G, Li M. 2012. Enhancing Genetic Algorithms for Dependent Job Scheduling in Grid Computing Environments. Journal of Super Computing. 62(1): 290-314.

[12] Wu A S, Yu H, Jin S, Lin K, Schiavone G. 2004. An Incremental Genetic Algorithm Approach to Multiprocessor Scheduling. IEEE Trans Parallel Distributed System. 15(9): 824-834.

[13] Yan Kang, Defu Zhang. 2012. A Hybrid Genetic Scheduling Algorithm to Heterogeneous Distributed System. Applied Mathematics. pp. 250-254.

[14] Rajabioun R 2011. Cuckoo optimization algorithm. Applied soft computing.

[15] Saurabh Kumar Garg, Srikumar Venugopal, James Brober, Rajkumar Buyya. 2013. Double Auction-Inspired Meta-Scheduling of Parallel Applications on Global Grids. Journal of Parallel and Distributed Computing.

[16] Lizhe Wang, Samee U. Khan, Dan Chen, Joanna Kołodziej, Rajiv Ranjan, Cheng-zhong Xu, Albert Zomaya. 2013. Energy-aware parallel task scheduling in a cluster. Future Generation Computer Systems.

[17] Chung-Hsing Hsu, Wu Chun Feng. 2005. A Feasibility Analysis of Power Awareness in Commodity-Based High-Performance Clusters. CLUSTER.

[18] Xiaomin Zhu, Rong Ge, Jinguang Sun, Chuan He. 2013. 3E: Energy-Efficient Elastic Scheduling for Independent Tasks in Heterogeneous Computing Systems. The Journal of Systems and Software.

[19] Ye Huang, Nik Bessis, Peter Norrington, Pierre Kuonen, Beat Hirsbrunner. 2013. Exploring Decentralized Dynamic Scheduling for Grids and Clouds Using the Community-Aware Scheduling Algorithm. Future Generation Computer Systems.

[20] Wei Liu, Wei Du, Jing Chen, Wei Wang, Guo Sun Zeng. 2014. Adaptive Energy-Efficient Scheduling Algorithm for Parallel Tasks on Homogeneous Clusters. Journal of Network and Computer Applications.

[21] Rajni, Inderveer Chana. 2013. Bacterial Foraging Based Hyper-Heuristic for Resource Scheduling in Grid Computing. Future Generation Computer Systems.

[22] K.M. Passino. 2002. Biomimicry of Bacterial Foraging for Distributed Optimization and Control. IEEE Control Systems Magazine.

[23] Sucha Smanchat, Maria Indrawan, Sea Ling, Colin Enticott, David Abramson. 2013. Scheduling Parameter Sweep Workflow in the Grid Based on Resource Competition. Future Generation Computer Systems.

[24] Ramya R and Shalini Thomas. 2012. An Optimal Job Scheduling Algorithm in Computational Grids. Special Issue of International Journal of Computer Applications.

[25] Anis Gharbi *et al.* 2013. An Effective Genetic Algorithm for a Complex Real-World Scheduling Problem. International Journal of Mechanical, Industrial Science and Engineering.

[26] Bsoul, M, Phillip I, Hinde C. 2012. Micosim: A Simulator for Modelling Economic Scheduling in Grid Computing. World Academy of Science, Engineering and Technology, International Science.

[27] Maryam Rabiee and Hedieh Sajedi. 2013. Job Scheduling in Grid Computing with Cuckoo Optimization Algorithm. International Journal of Computer Applications.

[28] D.I. George Amalarethinam, P. Muthulakshmi. 2014. A Proficient Low Complexity Algorithm for

Preeminent Task Scheduling Intended for Heterogeneous Environment. Journal of Theoretical and Applied Information Technology. 67(1): 1-11.

[29] D.I. George Amalarethinam, P. Muthulakshmi. 2012. DAGITIZER - A Tool to Generate Directed Acyclic Graph through Randomizer to Model Scheduling in Grid Computing. Advances in Computer Science, Engineering and Applications, Springer Verlag. pp. 969-978.

[30] Walaa AbdElrouf, Adil Yousif and Mohammed Bakri Bashir. 2016. High Exploitation Genetic Algorithm for Job Scheduling on Grid Computing. International Journal of Grid and Distributed Computing. 9: 221-228.

[31] T. Selvakumar *et al.* 2016. Implementation of Pervasive Semantic Grid Computing in Hospital Scenario. International Research Journal in Advanced Engineering and Technology. 2: 517-522.

[32] S. Vaaheedha Kfatheen *et al*. 2017. ETS: An Efficient Task Scheduling Algorithm for Grid Computing. Advances in Computational Sciences and Technology. 10: 2911-2925.

[33] Hajara Idris *et al*. 2017. An Improved Ant Colony Optimization Algorithm with Fault Tolerance for Job Scheduling in Grid Computing Systems. Plos One.

[34] Firas Albalas *et al*. 2017. Optimized Job Scheduling Approach based on Genetic Algorithms in Smart Grid Environment. International Journal of Communication Networks and Information Security. 9: 172-176.

[35] F Kurus Malai Selvi *et al*. 2012. Grid Scheduling Strategy using GA. International Journal of Computer Technology and Applications. 3: 1800-1806.

[36] Kalyanmoy Deb. 2001. Multi-objective Optimization Using Evolutionary Algorithms. John Wiley & Sons, Inc. New York, USA.