Journal of Computer Science 7 (6): 877-883, 2011
ISSN 1549-3636

Impatient Task Mapping in Elastic Cloud using Genetic Algorithm

Nawfal A. Mehdi, Ali Mamat, Hamidah Ibrahim and Shamala K. Subramaniam
Department of Computer Science,
Faculty of Computer Science and Information Technology,
University Putra Malaysia, 43400 Serdang, Malaysia

Abstract: Problem statement: Task scheduling is the main factor that determines the performance of any distributed system. Cloud computing comes with a paradigm of distributed datacenters. Each datacenter consists of physical machines that host virtual machines to execute customers' tasks. Resources allocation on the cloud is different from other paradigms and the mapping algorithms need to be adapted to the new characteristics. This study takes the problem of immediate task scheduling under an intercloud infrastructure using a genetic algorithm. An impatient task needs to be scheduled as soon as it enters the system taking into account the input and output files location and its QoS requirements. **Approach:** This study proposes an algorithm that can find a fast mapping using genetic algorithms with "exist if satisfy" condition to speed up the mapping process and ensures the respecting of all task deadlines. Cloudsim simulator was used to test the proposed algorithm with real datasets collected as a cloud benchmark. Mapping time and makespan are the performance metrics that are used to evaluate the proposed system. **Results:** The results show an improvement in the proposed system compared to MCT algorithm as illustrated throughout the study. **Conclusion:** Batch mapping via genetic algorithms with throughput as a fitness function can be used to map jobs to cloud resources.

Key words: Cloud computing, datacenter, genetic algorithms, Virtual Machine (VM), scheduling, intercloud paradigm, impatient task, QoS requirement

INTRODUCTION

Task scheduling means finding the best resource or Virtual Machine (VM) (cloud computing speaking) for the requested task. It plays an essential role in the system performance because it controls the data flow and the execution order of the incoming tasks. The optimal solution for task scheduling on a heterogeneous system is proved to be an NP problem and a long list of research has been done on many distributed system paradigms. The interoperating between multiple clouds is called Intercloud (Bernstein *et al*., 2009). It can be defined as a "cloud of cloud" (Metz, 2010) and it is a metaphor for the Internet Network of Networks. Intercloud Computing has an infrastructure that can be illustrated as a set of datacenters (i.e., cloud providers) connected to the Internet. These datacenters can be accessed from anywhere via simple Internet access tools. The popularity of the Internet and web services make cloud computing one of the best IT solutions to many current problems. One kind of problem is the problem of immediate or impatient tasks that need to be executed as soon as possible. To describe this problem in more detail, let us take the case of a natural disaster that may happen anywhere and anytime. These disasters need huge IT departments to analysis and recover the situation by executing immediate or urgent tasks.

Cloud computing is assumed to be the best solution for these kinds of problem because of its elasticity and growing on demand property. Dave Murphy, senior vice president at a performance testing vendor said, "The cloud allows for large amounts of computing power over short periods of time- like during a disaster- so government agencies can respond to anything in the world". He also noted "The government could utilize the cloud compute power on an as-needed basis. All of a sudden- like with the oil spill off the coast of Louisiana-they would need to bring resources to bear, such as money, people and equipment. Part of the equipment is the computing power". The primary Benefits of Cloud Computing, which make it important for many critical problems, need to be understood. The main benefits are summarized below:

- Virtualization, which is the abstracting and separation of the services from the infrastructure needed to run it

Corresponding Author: Nawfal A. Mehdi, Department of Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia, 43400 Serdang, Malaysia

- Flexibility to choose multiple vendors that provide reliable and scalable business services, development environments and infrastructure that can be leveraged out of the box and billed on a metered basis with no long term contracts
- Elastic nature of the infrastructure to rapidly allocate and de-allocate massively scalable resources to business services on a demand basis
- Cost allocation flexibility for customers wanting to move Capital Expenditure (CapEx) into operational expenditure (OpEx)
- Reduced costs due to operational efficiencies and more rapid deployment of new business

Based on the above, many researchers have tried to define the term Cloud Computing. Dr. Buyya defines cloud computing as "A Cloud is a type of parallel and distributed system consisting of a collection of inter-connected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resource(s) based on service-level agreements established through negotiation between the service provider and consumers" (Buyya *et al*., 2009). Managing and providing computational resources to user applications is one of the main challenges for the high performance computing community.

Task scheduling has been conducted in many ways and via many heuristics. Genetic Algorithms (GA) (Hormwichian *et al*., 2009; Sarabian and Lee, 2010) are search algorithms, which are based on the principles of evolution and natural genetics. GA are successfully applied to solve NP-complete problems. It is based on the idea of random guided search, which can give a better result for a large search space compared to an enumerative guided search. GA starts with a generation of individuals, containing feasible solutions. A certain fitness function is used to evaluate the fitness of every individual. This study takes the problem of task scheduling in immediate mode using genetic algorithms. This problem is divided into sub-problems, which are: the problem of data location aware, the problem of GA exit condition and the problem of task starting deadline. Because of the huge infrastructure for cloud computing, which covers the whole globe and its ability to create a large number of virtual machines, it is difficult to test the proposed work on a real system. The Cloudsim simulator has been used with two real datasets to evaluate the performance of the proposed system. The main contribution of this study is: adopting GA for the intercloud paradigm to serve data-intensive and time critical tasks.

Data location plays an essential role in determining the efficiency of the distributed system because of the huge growth in the application data compared to the application itself. It is inefficient to move megabytes of data, which might need dozens of minutes to a program, which might need a couple of minutes or less to execute. Recently, many researchers have studied the case of data intensive application. Ranganathan and Foster (2002) examine scheduling heuristics and the impact of data replication on it. While the work presented in Raicu *et al*. (2008) proposed a data diffusion approach that combines dynamic resource provisioning, on demand data replication and caching and data locality-aware scheduling, to achieve improved resource efficiency under varying workloads. They define an abstract "data diffusion model" that takes into consideration the workload characteristics, data accessing cost, application throughput and resource utilization; they validate the model using a real-world large-scale astronomy application. Work presented in Jin *et al*. (2005) proposed the adaptive scheduling model for data-intensive applications. They assumed a data grid model, which consists of multiple sites. Every site has different computational capabilities and data stores and the input datasets are replicated among them. Fatos in his study (Xhafa *et al*., 2007) considered the problem of allocation tasks using the immediate mode in a grid environment. They implemented five scheduling methods and used four parameters to measure the performance of the system; namely, (1) makespan, (2) flowtime, (3) resource utilization and 4) matching proximity. In this study, they did not consider the data-location of the scheduling process, but based it on the task execution time. The work presented in Orlando *et al*. (2002) examined the on-line Minimum Completion Time (MCT) heuristic strategy for scheduling high performance data mining tasks on top of the Knowledge Grid. Genetic Algorithms have been used in many scheduling problems because of their NP-complete complexity. Most GA techniques use the task and resource length for the fitness function. Another paper (Zhao *et al*., 2009) introduces an application of GA in task scheduling in order to adapt to the memory constraints and high request of performance in cloud computing. Many researchers (Wang *et al*., 1997; Budin *et al*., 2010; Hou *et al*., 1994; Grajcar, 1999; Hernane *et al*., 2010) proposed algorithms for scheduling tasks on grid and other distributed paradigms.

MATERIALS AND METHODS

Intercloud is a paradigm that has an infrastructure of datacenters that are distributed around the globe and connected with each other via the Internet. Each

datacenter consists of a storage site and computation site, which, in turn, has sets of physical machines used to host predefined VM images. Each task deployed to the cloud should be mapped to one or more VM based on its requirements. A virtual machine, which is assumed as the basic computation unit for cloud computing needs some delay time for its installing, booting and starting. The resource allocation process should take these characteristics into consideration.

Problem formulation: Each task comes with a set of input files, a set of output files and a set of QoS requirements. The time needed to stage in the input files or stage out the output files may be too expensive to be cancelled, so, the scheduling algorithm should take these two times into consideration while mapping the tasks to the VMs. Datacenters use VMs as a basic computation unit; these VMs need a particular time for installing, booting and starting up. This time should be considered and vary from one datacenter to another. To describe the problem of impatient tasks under the cloud paradigm formally, let: D Set of datacenters/cloud providers such that $D = \{d_0, d_1,..., d_{|D|}\}$. Let T set of tasks, $T = \{t_0, t_1,..., t_{|T|}\}$. Let QoS(t) be the set of QoS requirement for task, Fin_t is the set of input files for task t and $Fout_t$ is the set of output files for task t. Let V be the set of virtual machine image types, such that $V = \{v_0, v_1,..., v_{|V|}\}$ and SpC(v) is the set of virtual machine specifications.

Each task has two deadlines, namely, (sdt and dlt). Formally:

$$st_t \geq BET_{tvd} \quad (1)$$

and:

$$dl_t \geq FT(t) \quad (2)$$

Where:

st_t = The allowed start time such that any execution after it is useless

BET_{tvd} = The Begin Execution Time, which returns the time that the task t on VM v on datacenter d starts its execution

dl_t = The allowed finish deadline, such that the results after it are useless

FT(t) = A Finish Time function, which returns the estimated time that task t finishes. Mapping a task t to any VM v should not violate any QoS requirement, as written formally:

$$QoS(t) \leq SpC(v) \quad (3)$$

The main objective function of this work is to minimize the task starting time by speeding up the mapping process with the satisfaction of all QoS requirements. For the purpose of GA, we use throughput as a main objective function (i.e., fitness function), which can be defined as the number of tasks executed per time as shown in Eq. 5. Formally, it can be written as:

$$x_t = \begin{cases} 1, & \text{job t has been executed} \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

So, the throughput can be written as:

$$\text{Throughput} = \sum_{\forall t \in T} x_t \quad (5)$$

Where:

x = A decision veritable used to indicate whether task

t = Executed or not

Makespan is the maximum execution time among all received tasks, which can be written formally as:

$$\text{Makespan} = \max\{FT_t\} \quad \forall t \in T \quad (6)$$

where, FT is the finish time of task t.

From the GA side, there is a problem of mapping time because of its exhaustive search. This study addresses this problem and tries to solve it.

GA for scheduling in cloud computing: GA is a search technique that simulates the process of natural evolution. This heuristic is used to generate useful solutions to search problems and optimization. Algorithm (1) illustrates the main steps that are used in our implementation.

Encoding: Encoding is the process of designing the chromosomes in such a way that it is possible to encode all the tasks and resources in one string of bits. System performance depends totally on this design.

Algorithm 1 genetic algorithm:

1: Initialization: Great initial random population
2: Evaluate
3: Keep the best
4: While termination not true do
5: Selection
6: Crossover
7: Mutation
8: Evaluate

9: Elitist
10: Check exit
11: End while
12: Return mapping result

For mapping resources under the cloud computing environment with the requirement of impatient tasks, it is important to speed up the system as fast as possible by encoding the individuals (chromosome) efficiently. As described before, the intercloud paradigm is based on the infrastructure of datacenters connected to each other via the Internet, each one has a set of predefined VM images. The dynamism in the number of datacenters and the number of offered VM images is less than the dynamism under grid computing and other distributed systems, so it is possible to use the number of datacenters multiplied by the total number of VMs as the number of resources for our proposed system. Figure 1 depicts the proposed chromosome encoding.

Each t represents a single task while each res represents an index to a mapping table. This table represents the list of all datacenters and their VMs. This algorithm takes into account the available resources in each datacenter (i.e., How many images each datacenter can create), which is done at the evaluation (i.e., compute fitness function) process.

Population: The population size is set to 25 to have potential solutions in a population. Random generation is done on the initial chromosomes. These parameters can be adjusted as needed.

Crossover: Crossover is the process of reproduction. It is used to change the programming of a chromosome(s) from one generation to another. Two chromosomes are picked randomly from the population and apply crossover on them when a random value is less than a threshold (cthr). Another random variable is generated and used to select two genes for the exchange operation in each chromosome.

Mutation: This study uses random uniform mutation to implement the process of mutation. The selected variable is replaced by a random value in range between the lower and upper bands of the selected variable.

Each gene in each chromosome has a random value, which nominates it for mutation operation or not if its value is less than the first mutation threshold ($mthr_1$). If a gene is selected, its value is replaced by another random value between the lower and upper bounds ($mthr_2$).

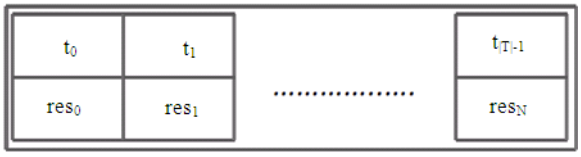

Fig. 1: Chromosome design

Elitist: The best chromosome of the previous generation is stored as the last in the array. If the best chromosome of the current generation is worse than the best chromosome of the previous generation, the latter one would replace the worst chromosome of the current population.

Stopping condition: Two ways to exit from the mapping process have been implemented. The first way is to do all the iterations, which is exhaustive for impatient tasks. The second one is when all the tasks are mapped with the meeting to all their QoS parameters.

Evaluation: The algorithm has to evaluate each chromosome to find the best fitness value. The fitness values represent the number of tasks mapped to VMs (i.e., throughput). A bigger number gives better throughput. This work needs some pre-requested data, such as the list of datacenters, the list of VMs for each datacenter and the specifications of each VM image. The algorithm finds the estimated completion time of each task via Eq. 7:

$$TCT_{tvd} = VMC_{vd} + Sin_{tvd} + Exec_{tvd} + Sout_{tvd} \quad (7)$$

TCT stands for Task Completion Time, which is the estimated time to finish executing the task and sending the result back. While cloud computing is based on virtualization, each virtual machine needs a time for creating and loading. VMC is the virtual Machine Creating time, which is pre-defined to the algorithm. Exec is the time needed to execute the task after all the input files are fetched and it is equal to the task length over VM speed. Sin is the process of fetching in all the required input files and Sout is the process of sending out the result data to pre-define destinations as shown in Eq. 8 and 9, respectively:

$$Sin_{tvd} = \sum_{\forall f \in Fin_t} \frac{size(f)}{\min(BW(VM_v), BW(dist))} \quad (8)$$

$$Sout_{tvd} = \sum_{\forall f \in Fout_t} \frac{size(f)}{\min(BW(VM_v), BW(dist))} \quad (9)$$

System model and experiments: In the intercloud paradigm, datacenters might have the ability to create VM, or provide storage only, or both. An example of a cloud provider that only has a computation service is GoGrid, while a cloud storage provider can be found in 3Tera and, lastly, Amazon as a cloud provider offering both services. Figure 2 depicts the proposed model. It consists of three datasets, meta-scheduler (i.e., computer icon) and users to submit their requests. Meta-scheduler is the main broker that controls the execution of tasks among the cloud providers (i.e., datacenters).

Dataset: The proposed algorithm has been tested on real datasets, which are:

- Montage: Montage was created by the NASA/IPAC Infrared Science Archive as an open source toolkit that can be used to generate custom mosaics of the sky using input images in the Flexible Image Transport System (FITS) format.
- Ligo (Brown *et al*., 2007): The Laser Interferometer Gravitational Wave Observatory (LIGO) is attempting to detect gravitational waves produced by various events in the universe as per Einstein's theory of general relativity.

Simulation: Because of the difficulty in testing the proposed system on a real system, a simulation evaluation has been conducted on the two datasets described previously. CloudSim (Calheiros *et al*., 2009) is a discrete event simulator that is used to simulate cloud environments.

Cloudsim has the ability to create datacenters, virtual machines and physical machines and configure system brokers, system storage. The proposed algorithm has been tested and evaluated on the four real datasets.

Table 1 shows the main configurations for cloudsim simulator.

Table 1: Cloudsim configurations

Item	Value
Number of datacenters	10
Number of VM	400
Number of CPU/VM	1
CPU Speed/VM	1, 2, 2.5 and 4 GHZ
Number of tasks	10, 15, 20, 40, 60, 80, 100

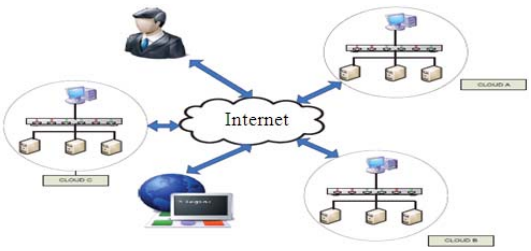

Fig. 2: General system model

All the experiments are done on a computer with CPU Intel Core 2 Duo 2.4GHz, RAM 4GB and using Windows Server 2008 operation system. Two performance metrics have been used to test the proposed algorithm. The mapping time, which is the time, needed to finish the mapping process and it is computed as a computer time difference before and after the mapping procedure call. Makespan is used to find the impact of speeding GA on total execution time. The Minimum Completion time (MCT) heuristic has been adopted to the cloud computing paradigm and used to compare the proposed algorithm.

RESULTS

One of the main features of GA is the random guided search, which is faster than an enumerative search (which is required to test every point in the search space) if the iterations are controlled. This study looks for the nearest local optima, which means the nearest mapping for the given tasks. The first experiment is used to calculate the mapping table of GA and MCT algorithms. Figure 3 and 4 depict the mapping time on the two datasets. Seven bags of tasks were assumed to be impatient tasks submitted to the simulator. These bags differ in the number of tasks. The results show the divergence in mapping time of the MCT algorithm because of the growth in search space. GA gives a simple increase in mapping time.

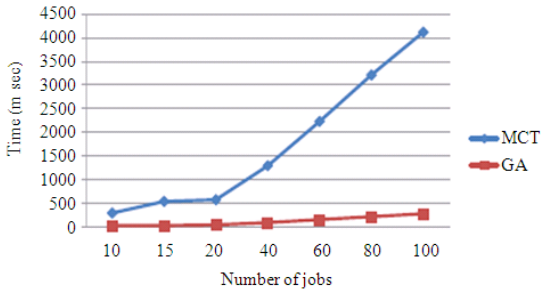

Fig. 3: Mapping time of tasks from montage dataset using MCT and GA algorithm

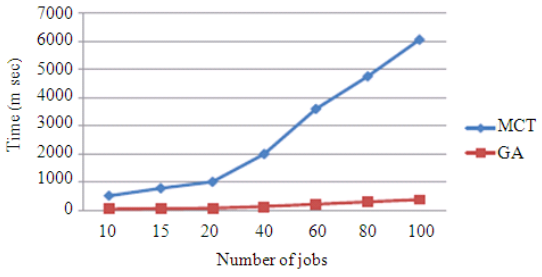

Fig. 4: Mapping time of tasks from ligo dataset using MCT and GA algorithm

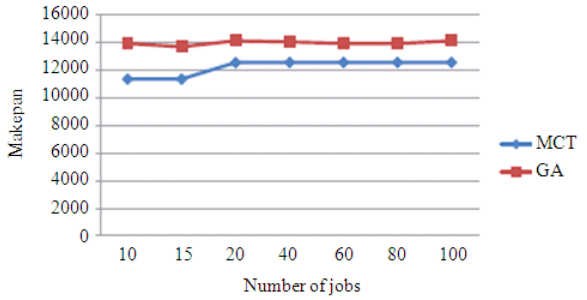

Fig. 5: Makespan of tasks from Montage dataset using MCT and GA algorithm

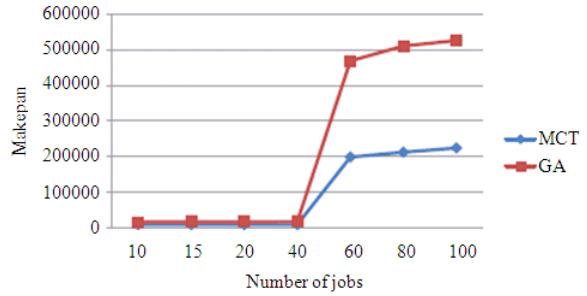

Fig. 6: Makespan of tasks from ligo dataset using MCT and GA algorithm

The second experiment is used to compute the makespan. Makespan, which is defined as the maximum finish time among all the given tasks, is depicted in Fig. 5 and 6. These two figures represent two different real projects. Each project has its own tasks and file length. This explains the difference in makespan between the two datasets. In both figures, the MCT is superior to GA in makespan because the makespam in MCT is the best choice among all search spaces.

DISCUSSION

Cloud computing has resources charged per use. These resources assumed as virtual machines and need for booting and loading. The main benefit from this new paradigm is the design of datacenters, which is assumed as cloud providers. Each datacenter has set of physical machine and list of virtual machine images.

The chromosome design depicts this new paradigm by map the jobs to virtual machines images.

These two experiments show the trade-off between the makespan and waiting time. If the set of jobs needs fast attention then the proposed model can give better results, while the MCT can be better from makespan prospective. In this study, we tried to explore the ability of genetic algorithm in mapping jobs to resources.

The real datasets reflect real workloads that have been used in real projects. The assumption of this work is “the user needs a fast attention to his/her jobs”, which leads to the problem of designing the fitness function. The fitness function is designed to ensure the execution of all/most submitted job list. The experiments that have been sown illustrate the ability to use genetic to map the jobs to resources but with some lose in global optima solution.

CONCLUSION

The two experiments show the superiority of the random guided search, represented by GA, to the enumerative search, represented by MCT, in mapping time if the GA stop condition is controlled on exit if satisfied. In addition, the way of using the throughput as an objective function can increase the mapping time. Our future work is to discover the message passing between the clouds for impatient tasks.

REFERENCES

Bernstein, D., E. Ludvigson, K. Sankar, S. Diamond and M. Morrow, 2009. Blueprint for the intercloud-protocols and formats for cloud computing interoperability. Proceedings of the 2009 4th International Conference on Internet and Web Applications and Services, May 24-28, IEEE Computer Society, Venice/Mestre, Italy, pp: 328-336. DOI: 10.1109/ICIW.2009.55

Brown, D.A., P.R. Brady, A. Dietz, J. Cao and B. Johnson *et al*., 2007. A case study on the use of workflow technologies for scientific analysis: Gravitational wave data analysis. Workflows e-Science, 39-59. DOI: 10.1007/978-1-84628-757-2_4

Budin, L., M. Golub and A. Budin, 2010. Traditional techniques of genetic algorithms applied to floating-point chromosome representations. Sign, 1: 11-52. http://www.zemris.fer.hr/~golub/clanci/krema96.pdf

Buyya, R., C.S. Yeo, S. Venugopal, J. Broberg and L. Brandic, 2009. Cloud computing and emerging IT platforms: Vision, hype and reality for delivering computing as the 5th utility. Future Generat. Comput. Syst., 25: 599-616. DOI: 10.1016/J.FUTURE.2008.12.001

Calheiros, R.N., R. Ranjan, C.A.F. De Rose and R. Buyya, 2009. CloudSim: A novel framework for modeling and simulation of cloud computing infrastructures and services. http://www.cloudbus.org/reports/CloudSim-ICPP2009.pdf

Grajcar, M., 1999. Genetic list scheduling algorithm for scheduling and allocation on a loosely coupled heterogeneous multiprocessor system. Proceeding of the 36th Design Automation Conference, June 21-25, New Orleans, LA, USA., pp: 280-285. DOI: 10.1145/309847.309931

Hernane, S., Y. Hernane and M. Benyettou, 2010. Migration algorithm of particle swarm optimization for a scheduling problem. J. Applied Sci., 10: 699-703. http://docsdrive.com/pdfs/ansinet/jas/0000/17217-17217.pdf

Hormwichian, R., A. Kangrang and A. Lamom, 2009. A conditional genetic algorithm model for searching optimal reservoir rule curves. J. Applied Sci., 9: 3575-3580. DOI: 10.3923/jas.2009.3575.3580

Hou, E.S.H., N. Ansari and H. Ren, 1994. A genetic algorithm for multiprocessor scheduling. IEEE Trans. Parallel Distribut. Syst., 5: 113-120. DOI: 10.1109/71.265940

Jin, H., X. Shi, W. Qiang and D. Zou, 2005. An adaptive meta-scheduler for data-intensive applications. Int. J. Grid Utility Comput., 1: 32-37. DOI: 10.1504/IJGUC.2005.007058

Metz, C., 2010. The meta cloud-flying datacenters enter fourth dimension. http://www.theregister.co.uk-/2009/02/24/the_meta_cloud/page3.html

Orlando, S., P. Palmerini, R. Perego and F. Silvestri, 2002. Scheduling high performance data mining tasks on a data grid environment. Eur. Par Parallel Proc., 121-137. DOI: 10.1007/3-540-45706-2_49

Raicu, L., Y. Zhao, L. Foster and A. Szalay, 2008. Data diffusion: Dynamic resource provision and data-aware scheduling for data-intensive applications. http://arxiv.org/abs/0808.3535

Ranganathan, K. and I. Foster, 2002. Decoupling computation and data scheduling in distributed data-intensive applications. Proceeding of the 11th IEEE International Symposium on High Performance Distributed Computing, pp: 352-358. DOI: 10.1109/HPDC.2002.1029935

Sarabian, M. and L.V. Lee, 2010. A modified partially mapped multicrossover genetic algorithm for two-dimensional bin packing problem. J. Math. Stat., 6: 157-162. DOI: 10.3844/jmssp.2010.157.162

Wang, L., H.J. Siegel, V.P. Roychowdhury and A.A. Maciejewski, 1997. Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. J. Parallel Distribut. Comput., 47: 8-22. DOI: 10.1006/JPDC.1997.1392

Xhafa, F., J. Carretero, L. Barolli and A. Durresi, 2007. Immediate mode scheduling in grid systems. Int. J. Web Grid Serv., 3: 219-236. DOI: 10.1504/IJWGS.2007.014075

Zhao, C., S. Zhang, Q. Liu, J. Xie and J. Hu, 2009. Independent tasks scheduling based on genetic algorithm in cloud computing. Proceedings of the 5th International Conference on Wireless Communications, Networking and Mobile Computing, Sept. 24-26, IEEE Press, Beijing, pp: 5548-5551. DOI: 10.1109/ WICOM.2009. 5301850