Int. J. Nonlinear Anal. Appl. In Press, 1–8
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2024.35006.5225

Optimizing scheduling in cloud computing using the Cuckoo optimization algorithm

Kasra Rashidi

Department of Computer, Isfahan University, Isfahan, Iran

(Communicated by Seyed Hossein Siadati)

Abstract

The cuckoo optimisation algorithm is a meta-exploratory optimization algorithm used to solve nonlinear continuous optimization problems. Optimization is to find a way of fulfilling a task in the best manner. Principally, optimization refers to changing a series of primary information and using problem information to achieve more appropriate responses. Recently, graphics processors have been proposed as a multi-purpose computational device due to low cost, parallel architecture and improved access provided by programming environments such as the CUDA framework. The Master-Slave Model is one of the parallelizing models of optimization algorithms. In the present paper, shared memory, reduction operations and other factors affecting GPUS have been employed using CUDA and the Master-Slave model with a fine-grain technique to increase the efficiency of the parallel algorithm. In this work, the implementation time of parallel and serial algorithms has been evaluated using a benchmark function. The experiments' results revealed an increase in speed-up of the parallel algorithm compared to the serial algorithm.

Keywords: optimizer algorithm, graphic processing units (GPU), parallel and series algorithms, cloud computing
2020 MSC: 65D15, 68U10

1 Introduction

Cloud computing is a service model in which computational and storage facilities, such as processors, memory, bandwidth, and various software, are provided for users in online and network forms with easy and fast access. In this model, some companies attempt to provide computational and storage facilities at a wide level for users. To use these services, users are required to store their data in a physical environment that provides cloud computing. Therefore, other users do not have physical control over their data, providing cloud computing controls data. Under these conditions, users are required to trust a cloud computing provider to store their data in providing data center and use its services. During the recent decades, regarding optimization discussion, engineers seek methods which can meet their ends and minimize existing problems and barriers of the classic methods route in these methods. Meta-heuristic methods such as genetic algorithm, birds' population and ants' colony have helped engineers to their goals somehow. These natural event-based methods have both advantages and disadvantages. As the advantages of these methods, they can be referred to as simplicity and comprehensibility of concepts, as well as an appropriate and large search space (since an optimal response cannot be found in all search space through non-heuristic methods). Their main disadvantage is a lack of high stability, which means that their responses cannot always be trusted. In the present research, it has been made to completely discuss the proposed algorithms. Also, it has been made to optimize cloud

Email address: kasra.rashidi.1994@gmail.com (Kasra Rashidi)

Received: June 2024 *Accepted:* August 2024

databases path scheduling by presenting an appropriate mixed algorithm as well as comparing it with the previously presented algorithms under identical conditions.

The advent of cloud computing basic concepts dates back to the 1947s. These concepts were proposed by John McCarthy for the first time. The proposed concepts have considerably progressed over time. During recent years, as one of the most important technologies to deliver advanced services demanded through the public internet, cloud computing has become necessary. A cloud computing environment is capable of safely and easily storing data. It can also provide computational power through the Internet. Virtualization, resource scheduling problem, capability of distribution and development are integral characteristics of cloud computing. Resource scheduling problems in cloud computing environments are very important issues for which many methods have been employed. Nature-based algorithms are one of the methods used in this regard. In the following, a review of the related studies on scheduling in the cloud is presented.

In [6], presenting an article entitled "cloud database route scheduling based on ant colony optimization algorithm", introduced the problem of industry development as a way of effective access to saving resources route in the cloud. As they concluded, employing the ant colony optimization algorithm for cloud database scheduling route has led to many works such as intelligent routing, rapid access to the database and overall distributed computations optimization. Doing so, they stated that most of the relations between ants and ants-environment are based on a chemical substance produced by them. This chemical substance called Pheromone is used by ants to mark routes such as the route of food resources, to formicary on the earth. Food resource to formicary has two nodes, and these nodes are considered in a database cloud computing environment. The ants' traffic route in the nodes is regarded as the connection between the database, and the access to the required node is regarded as a food resource.

Scheduling policies in a cloud environment highly depend on the development model in the cloud. Most of the NP-Complete heuristic algorithms are used for various problem-solving. To solve the problem of scheduling in a distributed environment, some algorithms such as genetic algorithm, tabu search and simulated annealing are employed [3].

In [2], an FCFS algorithm-based method has been presented to schedule parallel work in cloud computing. In [7], FCFS scheduling has been used with the Backfilling algorithm, in which small works are brought forward. This method can prevent the gap between resources created by FCFS. In [4], where the study was conducted based on a genetic algorithm, chromosomes are the same grid environment input works. The fitness function is a coefficient of the time required for output work processing. To compute the output work processing, it is needed to compute $f = (ni/ri)$ where ni indicates the discussed work and ri denotes the resource of fulfilling the work. Once, the value of the function is computed for all the chromosomes. To compute the best chromosome, the roulette selection method is used (in this method, the element with the highest fitness value is selected). Therefore, the chromosome with the highest target function value is selected. Firstly, based on the method, two chromosomes are selected. Secondly, an integer value between 1 and the resource's average, such as crossover, is selected for the chromosome. Thirdly, two values of the crossover point are displaced. As a sequence, the new chromosome value is replaced with the previous one. The performance of a mutation is also a base performance, which is measured based on its probability. The probability of finding an optimal value never becomes zero.

In [9], where the study was performed based on the ant colony algorithm, the user sends his/her request for processing to the resource. Then, the details of work, such as the total number of works, the length of each work, CPU necessary to fulfil the work, are put in the request. After receiving the request of the customer, the resource mediator begins to compute parameters related to scheduling. Then, the resource with the highest pheromone value is selected for processing work. Global updating is performed after complete work allocation by resource, and implementation results are sent by the user. Initializing the pheromone matrix is sent based on estimating time values, and work implementation time is computed by the resource after allocating work to the resource. The highest pheromone value indicates the best resource, which is selected in each repetition. That is, the work of j will be processed by resource i. When the work I is attributed to resource v, local updating will be performed. Local updating is performed only for attributed works.

In [12], an algorithm based on the behaviour of fish and birds, unlike separated routes used in the ants colony algorithm, has been used. This algorithm searches the solution space by adjusting various factors. The PSO algorithm consisted of ten steps to schedule work in the grid. In the first step, through initializing, the speed vector and place vector are initialized. In the second step, the value of the repetition variable is updated. In the third step, the speed vector is updated for each particle using the mentioned formula. In the fourth step, the position vector of the particles is updated. In the fifth step, permutations are computed using the SPV law. In the sixth step, the operators' vector is computed. In the seventh step, the best local point for each particle is updated by comparing the best previous point and the operator's vector obtained in the previous step. In the ninth step, the best general point is updated, and in

the tenth step, the repetition variable is investigated. In this step, if the repetition point exceeds, the algorithm is concluded, and if not, the algorithm goes ahead. The place of each particle shows a possible solution in the problem space.

In [9], inspiring the behaviour of molecules in chemical reactions, work scheduling in a grid has been optimized. As is known, in nature, substances try to reach the minimum level of potential energy. In this regard, and chemical reactions, 4 states occur for a molecule. It is assumed that molecules are placed in a closed environment. Therefore, they encounter each other or with capsule wall.

2 Cloud database

A cloud can be defined as a parallel and distributed system with several virtual and interrelated computers. This fact is regarded as a unified computational resource or dependent on the service-level agreement (SLA). Cloud has three well-known computational paradigms, including infrastructure as a service, software as a service and platform as a service. These services include a distributed operating system, a distributed database and other services. Cloud computing databases are immediately and effectively required to decrease routing configuration heaviness. Cloud databases consist of website sets including nodes connected by a communication network. Each node is only a database class. Each database has its own database, terminals, central processor, and local database management system. A database is an organised set of data. A database management system is a software package with computer programs which control creating, maintaining and using a database. This capability allows organizations to develop databases for various applications. A database includes an integrated set of data, files and other objects. A database management system allows various programs of software programs to have simultaneous access to the database. The database management system may employ various database models, such as the relational model or the target model, to easily describe and support applicable programs. The term of database refers to the information and structures of data, but not to a database management system. A database with a database management system is called a database system.

A cloud database is usually established on cloud computing platforms such as Amazon, GoGrid and Rackspace. There are two typical establishment models: users can independently set up a database on the cloud through a virtual machine, or purchase an access account to database services maintained by cloud database providers. Some of them are based on SQL and some others are based on NOSQL data model. Database technologies have been promoted to support cloud computing.

2.1 Task Scheduling Methods in Cloud Computing

During recent years, with the increasing growth of processing information volume, there is a greater need for distributed systems and parallel processing than before. Grid computations belong to distributed systems and cloud computing systems infrastructure. Using communication infrastructures and computer networks, this technology provides the possibility of access to various resources from a remote location. Heterogeneous software and hardware computational resources can be connected, without geographical limitation, such that the total system structure is seen as a virtual machine. Then, very complex and large applicable programs which require very high processing power and a huge volume of data can be implemented over this virtual machine. The purpose is to make use of computational resources systems when they are free to implement immense works. In fact, cloud computing is a pattern of distributed computations, composed of a large number of resources and requests, with the request aiming at sharing resources as a service over the internet platforms. Services in cloud computing are presented at three levels:

1. Software as a Service (SaaS)
2. Platform as a Service (PaaS)
3. Infrastructure as a Service (IaaS)

This classification provides the possibility of supporting different management for each level, as well as virtualisation technology. In a cloud environment, processing is allocated over a pool of dynamic and virtual resources in an online form and based on users' needs. Task scheduling is a key process in Iaas to implement requests entered into the system over resources in an efficient manner, considering other characteristics of the cloud environment. Task scheduling considers VMS as scheduling units to allocate heterogeneous physical resources to fulfil tasks. Each virtual machine is an abstract unit of computational and storage capacities provided in the cloud.

Due to the dynamic characteristics of the cloud computation environment and its heterogeneity, task scheduling is regarded as an NP-complete problem. In such a system, the scheduling process should be performed automatically and rapidly.

2.2 Cuckoo optimization algorithm

Just like other heuristic algorithms, COA starts to work with a primary population. This population of cuckoos lays eggs in the nest of other birds, called hosts. A number of these eggs, which are more similar to the host birds' eggs, will have a better chance of growing up and maturing. Other eggs are identified and disappear by the host bird. The number of grown-up eggs indicates the appropriateness of that region's nests. The higher the number of eggs living in a region is, the higher the willingness of that region will be. Therefore, the situation in which more eggs are saved will be a parameter which COA tends to optimize.

To maximize the savings of their eggs, cuckoos search for the best region. After hatching and maturing, cuckoos form communities and groups. Each group has its own specific settlement. The best settlement for all groups will be the next destination of cuckoos in other groups. All the groups immigrate to the best current region. Each group stays in the region near the best current situation. Considering the number of eggs laid by each cuckoo as well as the distance of cuckoos from the current optimal region for settlement, several egg-laying radiuses are computed and formed. Then, cuckoos start to randomly lay eggs in nests within their egg-laying range. This process continues to reach the best place for laying eggs (the region with the highest benefit). This optimal place is the place where the highest number of cuckoos meet each other.

After growing up and maturing, cuckoos live in their own environments and groups for a while, but when they approach egg-laying, they immigrate to better settlements where the chance of egg survival is more. After forming cuckoo groups in various environments (problem search space), the group with the best situation is selected as the target point for other cuckoos to immigrate. When mature cuckoos live in various settlements, it is difficult to specify which cuckoo each belongs to. To solve this problem, cuckoos are clustered by K-means.

After several repetitions, all cuckoo population come to an optimal point with the maximum similarity of eggs to birds' eggs and reaches a place with the highest food resources. This place will have the highest overall benefit, and the lowest number of eggs will disappear. Convergence with more than 95%, all the cuckoos, towards a point, reaches COA to their end.

3 GPU Architecture and CUDA Tool

The necessity of the CUDA programming pattern is to separate the proposed problem into some sub-problems which can be solved in parallel. On the one hand, each of these problems can be divided into smaller problems and implemented in parallel. In CUDA discussion, CPU and computer memory are called the host and GPU, and their memories are the so-called memories of that device. Also, the software element describing instructions for each thread's implementation is called the kernel. Unlike ordinary functions implementation which is implemented over the CPU only once, the kernel is implemented n times in parallel with n different CUDA threads. The value of n is determined by the programmer. When a kernel is called on the host, the number of thread blocks should be determined.

GPUs have several types of memory with different access. Each thread has reading-writing access to its own specific local memory, and the most rapid memory is in terms of accessing the same block's threads. Global memory for all defined processing threads can be accessed in read-write form. This memory acts as a covert memory to fulfil graphic computations.

4 Parallelizing COA

The only dependence between cuckoos and their settlements pertains to data which should be shared at the end of each generation among cuckoos. Therefore, it can be claimed that COA is naturally parallel. Table 1 shows the implementation time of various parts of COA in a series implementation. In the table, the time bottlenecks of COA can be easily identified.

Given the implementation time of subfunctions and their appropriateness for parallel implementation, kernels can be identified. The stage of egg-laying has not been parallelized due to some problems with generating random figures in the graphics card memory. Fitness function computation operations, convergence and clustering have been selected

Table 1: Computing COA parts' time (based on percentage)

phase	Percentage
initialization	0.013%
Egg laying	16.258%
Remove eggs in same positions	0.004%
Fitness function	42.801%
Kill cuckoo	0.761%
convergence	5.528%
Clustering cuckoo	29.782%
Select goal point	0.004%
Migration	4.849%
Total	100%

for parallelization on the graphics card. The simplest parallelization technique of the fitness function and other time-consuming functions of COA is to use the Master-Slave Model. Such that, the master processor is regarded CPU and the slave processors are regarded as GPU processing threads. Therefore, initializing is performed on the host, and if parallelization is necessary, data should be transmitted from host memory to device memory. After performing parallelization operations over the GPU, the results are copied to host memory. Figure 1 shows the flowchart of COA based on the Master-Slave Model over a GPU.

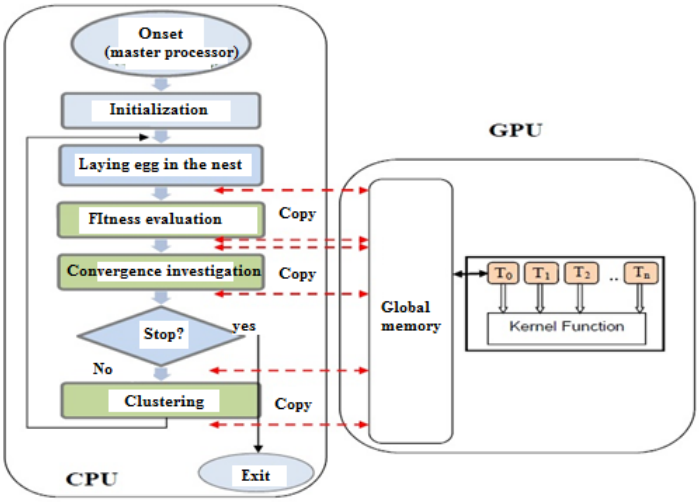

Figure 1: The Flowchart of COA based on Master-Slave Model over GPU

Each thread has a unique number among all defined threads. On the one hand, each thread among the threads of a block has its own specific number. In other words, each thread has access to global and shared memory with two identification numbers. Therefore, having these two numbers, the thread can read a certain data from global memory and transmit it to a certain place in the shared memory of its block.

5 Experiments and results analysis

In this section, the version presented on some of the common benchmark functions used to compare various optimization methods has been implemented. The results of the two different experiments have been investigated. The first experiment investigates the behaviour of the presented algorithm by keeping the number of problem dimensions and reiterations constant and increasing the number of cuckoos. By keeping the number of cuckoos and iterations constant and increasing the number of problem dimensions, the next experiment investigates the parallel algorithm. The employed benchmark functions have been listed in the following:

$$F1 = \sum_{i=1}^{n} x_i^2$$

$$F2 = \sum_{i=1}^{n} \sum_{j=1}^{i} x_j^2$$

$$F3 = \sum_{i=1}^{n-1} (100\ (x_i + 1)^2)$$

$$F4 = 10n + \sum_{i=1}^{n} (x_i^2 + 1).$$

In these experiments, implementations have been developed using the latest version of CUDA and Visual Studio. The maximum number of eggs of each cuckoo is 3 in both experiments. The average time has been measured for 30 implementations based on seconds. Since the results of the standard deviation for the measured values were slight, they have not been reported.

Table 2: The characteristics of the mentioned benchmark functions

The minimum global value	Search space	The name of function
F1	$[-5.12, 5.12]^n$	0
F2	$[-65.536, 65.536]^n$	0
F3	$[-2.048, 2.048]^n$	0
F4	$[-5.12, 5.12]^n$	0

The first experiment

In this experiment, the variety of cuckoos varies from 1 to 1024 cuckoos. The problem dimensions have been constant and initialized with 100. The number of reiterations has also been considered constant. Figure 2 shows the results of the first experiment.

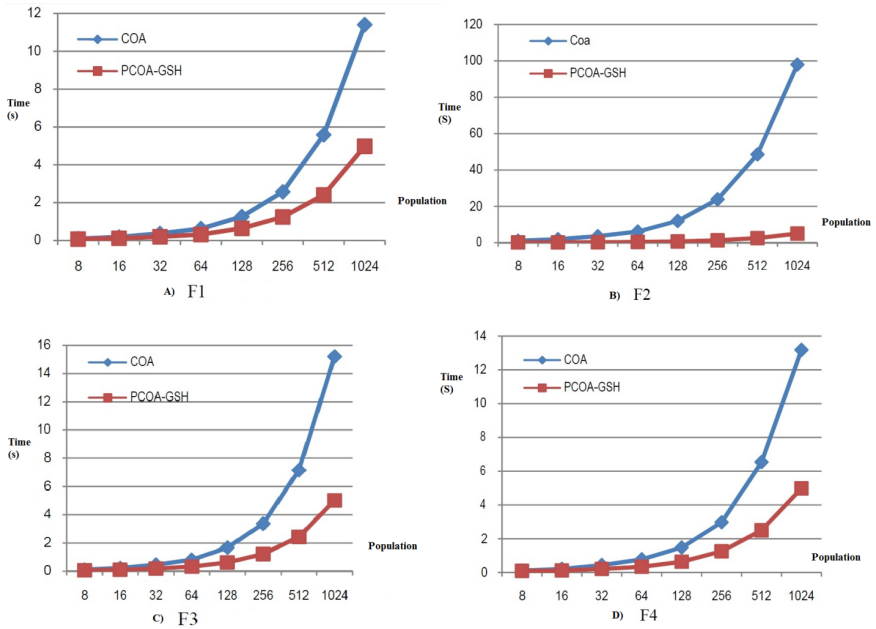

Figure 2: The results of the first experiment based on the mentioned benchmark function

As shown in Figure 2, A) indicates F1 function in a state in which speed up equals 2.90 if the population of cuckoos is increased up to 1024; B) indicates function F2 for the difference between parallel and series implementation in a state in which speed up equals 19.54 for the increase of population up to 1024; C) indicates function FC with speed up rate of 3.03 for the increase of population up to 1024; and D) indicates function F4 with speed up rate of 2.65 for the increase of population up to 1024.

6 The Second Experiment

In this experiment, the variety of the problem dimensions varies from 1 to 1024 dimensions. The number of cuckoos has been constant and is considered to be at most 15. Just like the previous experiment, the number of reiterations has also been considered constant. Figure 3 shows the results of this experiment.

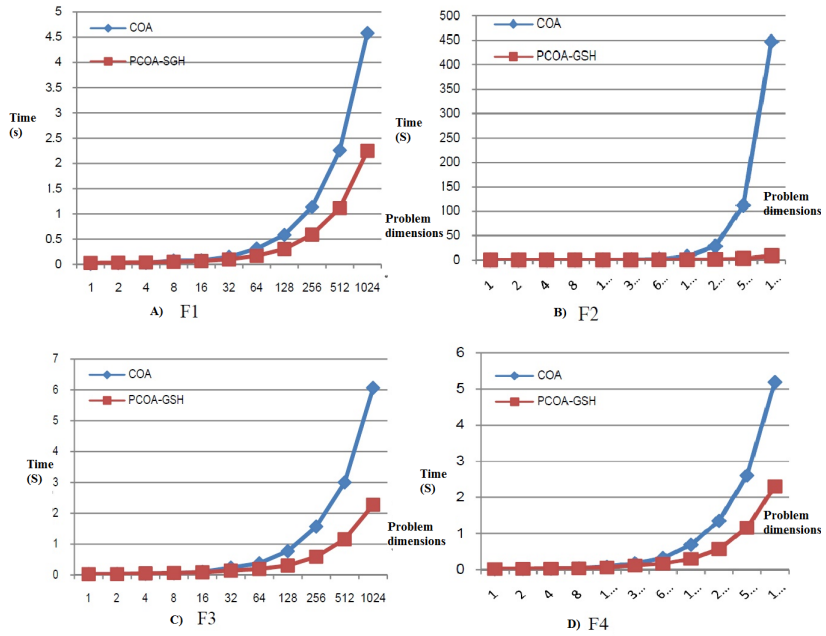

Figure 3: The results of the second experiment based on the mentioned benchmark functions

Shown in Figure 3, A) indicates the comparison of implementation time of parallel and series COA for F1 function in a state in which speed up equals 2.04 if the problem dimensions is increased up to 1024; B) indicates the difference between parallel and series implementation of function F2 in a state in which speed up equals 50.87 for the increase of the problem dimensions up to 1024; C) indicates the results of comparing implementation time of series and parallel COA for evaluating function FC with speed up rate of 2.67 for the increase of the problem size up to 1024; and D) indicates the results of comparing implementation time of series and parallel COA for unction F4 with speed up rate of 2.25 for the increase of the problem size up to 1024.

7 Conclusion

In the present paper, the cuckoo optimization algorithm was implemented on a GPU. In the Master-Servant Model, the master processor was considered the CPU and the servant processors were considered the GPU. The program is initialized and implemented in the master node, and in case of any need for parallelization, the required data is transmitted to the GPU through global memory. The stages of the algorithm which allocate themselves higher time from the CPU in implementation are called by the software element, namely the kernel. Processing threads simultaneously implement the kernel. Each thread calls data related to its ID from global memory and copies it into shared memory. At the end of the kernel, the results are returned to the master node through global memory by computation reduction action. In the proposed algorithm, computation of fitness, convergence and clustering has been implemented over the GPU.

The experimental results revealed that:

1. Changing the problem dimensions and the algorithm population to better speed up is being proceeded with for the GPU version. Obviously, increasing the number of reiterations to compute the mentioned functions' estimation results in similar results. Given the performed experiments, it can be concluded that the speed-up rate is better when the number of problem dimensions and population is increased.

2. In memory classification in GPU, it was revealed that the access of threads to global memory, compared to shared and local memory, as well as data copying operations between CPU and GPU, is performed through global memory with a high delay. Therefore, shared memory is used as much as possible, and it is employed to communicate with the global memory host.

References

[1] D.J. Abadi, *Data management in the cloud: Limitations and opportunities*, IEEE Data Eng. Bull. **32** (2009), no. 1, 3–12.

[2] M. Dorigo and G.D. Caro, *The ant colony optimization: A new metaheuristic*, Evolut. Comput., Proc. 1999 Cong. Pub. Vol. **2**, 1999, pp. 1477.

[3] J. Grefenstette and J. Baker, *How genetic algorithms work: A critical look at implicit parallelism*, Third Int. Conf. Genetic Algorithms, Morgan Kaufmann, San Mateo, CA, 1989, pp. 20–27.

[4] R. Hassan, B. Cohanim, and O.D. Weck, *A compmarison of Particle Swarm Optimazation and the Genetic Aalgorithm*, Vanderplaats Research and Development, Inc., Colorado Springs, CO, 80906.

[5] R.L. Haupt and S.E. Haupt, *Practical Genetic Algorithms*, second ed., John Wiley& Sons, New Jersey, 2004.

[6] S. Heng-liang, B. Guang-yi, T. Zhen-min, and L. Chuan-ling, *Cloud database dynamic route scheduling based on ant colony optimization algorithm*, Comput. Sci. **37** (2010), no. 5, 143–145.

[7] J. Kennedy, *Bare bones particle Swarms*, Proc. IEEE Swarm Intell. Symp., 2003, pp. 80–87.

[8] S. Mangalampalli, G.R. Karri, and G.N. Satish, *Efficient workflow scheduling algorithm in cloud computing using whale optimization*, Procedia Comput. Sci. **218** (2023), 1936–1945.

[9] V. Mateljan, *Cloud database-as-a-service (daas)-ROI*, Proc. 33rd Int. Convect., 2010, pp. 1185-1188.

[10] K. Michael, A. Bernstein, and M. L. Philip, *Database Systems*, Pearson Education, Inc, 2006.

[11] E.G. Talbi, *Metaheuristics: From Design to Implementation*, Wiley, ISBN, 2009.

[12] H. Thomas, *A veritable bucket of facts' origins of the database management system*, Proc. Medford, New Jersey, 2006, pp. 33–49.

[13] Zh. Yan-huaa, F. Leia, and Y. Zhia, *Optimization of cloud database route scheduling based on combination of genetic algorithm and ant colony algorithm*, Procedia Engin. **15** (2011), 3341–3345.

[14] Q. Yu, C. Chen, and Z. Pan, *Parallel genetic algorithms on programmable graphics hardware*, L. W. et al. (ed.) Advances in Natural Computation, Lecture Notes in Computer Science , LNCS 3612, 2005, pp. 1051-1059.

[15] W. Zhu and J. Curry, *Parallel ant colony for nonlinear function optimization with graphics hardware acceleration*, IEEE Int. Conf. Syst. Man Cybernet., IEEE, 2009, pp. 1803–1808.