Emerging Research and Solutions in ICT 1(1):75–87 DOI: 10.20544/ERSICT.01.16.P08
UDC: 004.421.032.24:[004.272.3:004.6

Artificial Bee Colony Optimization Technique for Effective Resource Allocation

Atanas Hristov[1], and Plamenka Borovska[2]

[1] University of information science and technology, Partizanska bb,
6000 Ohrid, Macedonia
atanas.hristov@uist.edu.mk
[2] Technical university of Sofia, Kliment Ohridski 8,
1000 Sofia, Bulgaria
borovska@tu-sofia.bg

Abstract. Accelerating the development and deployment of advanced communication technologies and complex databases will require a comprehensive strategy integrating efforts from invention to deployment. The concurrent high-performance computing systems are composed of hundreds of thousands of computational nodes, as well as deep memory hierarchies and complex interconnect topologies. Existing high performance algorithms and tools already require courageous programming and optimization efforts to achieve high efficiency on current supercomputers. On the other hand, these efforts are platform-specific and non-portable. Since most of the existing optimization algorithms and tools are not optimized for modern computer architectures and cannot efficiently exploit massively parallel systems, one aim of this research is to identify and to analyze the general problems and modern trends in this research area. This paper investigates the efficiency of artificial bee colony optimization algorithm for effective resource allocation. Parallel version of the algorithm have been proposed based on the flat parallel programming model with message passing for communication between the computational nodes in the platform and parallel programming model with multithreading for communication between the cores inside the computational node. Parallel communications profiling is made and parallel performance parameters are evaluated on the basis of experimental results.

Keywords: high-performance computing, optimization, parallel algorithm, multithreading.

1. Introduction

The modern high-performance computing systems (HPCS) are composed of hundreds of thousand computational nodes. An effective resource allocation in HPCS is a subject for many scientific research investigations. Many programming models for

effectively resources allocation have been proposed. The main purpose of those models is to increase the parallel performance of the HPCS. Currently, most of the high-performance systems are based on convectional sequential programming languages like C, C++, FORTRAN, etc. In order to achieve better parallel performance the flat parallel programing model with message passing in distributed memory systems, supported by the MPI standard and parallel programming model with multithreading in shared memory systems using the OpenMP programming interface have been included as a template libraries. The main disadvantages of the parallel programming based on conventional programming languages are: process synchronization, deadlocks, workload balancing, and thread concurrency.

In order to improve this situation, Intel provides a range of tools specifically designed to help developers parallelize their applications. Three sets of complementary models for multithreading programming in shared memory systems are supported by Intel: Intel Cilk Plus, Intel Threading Building Blocks (Intel TBB) and Intel Array Building Blocks (Intel ArBB). The main purpose of those models is to increase the reliability, portability, scalability and the parallel performance of the application during the multithreading execution [1-3].

The complexity class of decision problems NP-complete can be used as a pattern for benchmarking and parallel performance evaluation of multi-core and multi-machine architectures. Parallel versions of several NP-complete problems, such as N-Queens Problem, Travelling Salesman Problem, Sam-Loyd Puzzle etc., will be proposed in order to determinate the overall parallel performance of the system.

The paper investigates the efficiency of parallel algorithms for resource management optimization based on Artificial Bee Colony (ABC) metaheuristic while solving a package of NP-complete problems on multi-processor platform. Hybrid parallel model is proposed based on the flat parallel programing model with message passing for communication between the computational nodes in the platform and parallel programming model with multithreading for communication between the cores inside the computational node. The Intel Threading Building Blocks (TBB) programming model has been chosen as a standard for multithreading computations in shared memory systems. The Message Passing Interface (MPI) has been chosen as standard for communication in distributed memory systems.

2. Background and related work

Swarm intelligence is very important area in the field of optimization, involving the study of collective behavior in decentralized systems [4]. In the last decades, the researchers have developed various algorithms by modeling the behaviors of various swarms of insects and animals such as ants, termites, bees, birds, fishes. Recently, one of the most widely studied algorithms that have been applied to solve optimization problems in various areas are inspired by the intelligent behaviors of bee swarm. The Artificial Bee Colony (ABC) algorithm simulates the collective behavior of the honeybees in nature. The optimization tool of ABC algorithm provides a population-based search procedure in which foods positions are modified with time by the individuals called artificial bees. On the other hand, the bee's objective is to discover the

location of the food sources with high nectar amount and lastly the one with the highest nectar [5, 6].

The ABC algorithm consists of three main key groups of artificial bees that fly around in a multidimensional search space. These groups are represented by: the employed bees, the scout bees, and the onlooker bees. Initially, all food source positions are discovered by scout bees. The employed bees basically collect nectar from the food sources and once they have emptied the food source they become scouts who then start searching for new food sources. The number of employed bees is the same with the number of food sources since each employed bee is associated with one and only one food source. The scout bees choose the food sources randomly without using experience. If the nectar amount of a new source is higher than source that is previous recorded in their memory, they memorize the new position if the source and remove the previous one. The onlookers observe the behavior of the other bees and choose food sources accordingly.

The common scheme of the ABC algorithm is as follows:

```
Phase 1: Initialization
REPEAT
          Phase 2: Employed Bees
          Phase 3: Onlooker Bees
          Phase 4: Scout Bees
          Memorize the best solution achieved so far
UNTIL (Cycle=Maximum CPU time or Maximum Number of Cycle)
```

All population of the food sources are initialized by scout bees in the Phase 1. Also all control parameters are set in this phase. Since each food source is a solution of optimization problem, each solution vector holds n parameters which need to be optimized in the next three phases. In phase 2 the employed bees search for new food source having more nectar within the solution vector in their memory. Once they find the next food source, they evaluate its fitness and a greedy selection is applied. Employed bees share their food source information with onlooker bees in Phase 3 and then onlooker bees probabilistically choose their food sources depending on this information. In this algorithm, an onlooker bee chooses a food source depending on the probability values calculated using the fitness values provided by employed bees. Employed bees which solutions cannot be improved through a predetermined number of examinations, become scouts and the Phase 4 is activated. The converted scouts start to search randomly for new food sources. On the last step the ABC algorithm memorize the best solution that is achieved after phase 2, 3 and 4. This procedure is repeated until the maximum number of cycles is reached or until maximum number of CPU time is gained [7, 8].

In general, ABC system combines local search methods, implemented by employed and onlooker bees, with global search methods, operated by onlookers and scouts, attempting to balance exploration and exploitation process. In the proposed algorithm, the location of a food source denotes a possible solution to the problem and the nectar amount of a food source links to the quality of the associated solution.

3. Implementation of the Optimization Algorithm for Effective Resource Allocation

An effective resource utilization of the modern high performance computing (HPC) platforms is a subject for many scientific research investigations. The resource management optimization for those platforms is an essential part for optimal resource allocation while solving NP-complete problems. The complexity class of decision problems NP-complete can be used as a pattern for benchmarking and parallel performance evaluation of multi-core and multi-machine architectures. The main idea behind using NP-complete problems for evaluation of the parallel performance of the optimization algorithm for effective resource allocation based on ABC metaheuristic in multi-processor platforms is that those problems cannot be solved in polynomial time in any known way which require high computational power and time. In order to evaluate the overall performance of the system and effectiveness of the proposed optimization algorithm, a package of parallel implementations of NP-complete problems including Traveling Salesman Problem (TSP) [9], N-Queens Problem [10], and Sam-Loyd Puzzle [11] have been proposed.

3.1. Parallel computing model for implementation of the optimization algorithm

An effective optimization algorithm strongly determines the overall parallel performance of the high-performance computing system [12-14]. The proposed algorithm for effective resource allocation in multi-core and multi-machine platforms is based on Artificial Bee Colony (ABC) metaheuristic.

The basic approach during implementation process is building a computer model which will simulate the collective behavior of the bees while collecting nectar. Parallel computing model for effective resource allocation based on Artificial Bee Colony (ABC) metaheuristic is present on figure 1. In order to implement the parallel optimization algorithm on multiprocessor platform, the Intel ArBB programming model and ABC metaheuristic have been used. The proposed algorithm used dynamic resource allocation. The activities of each beehive simulate one processor, while the actions of bees simulate threads. The number of bees that simulate each thread depends on the architecture of the target platform. The algorithm supports two types of global data: table of available resources and a table of unfinished tasks. The tables should be visible to all bees as bees carry out direct access to the data founded in the tables.

In the proposed algorithm, the bees are divided in two beehives, beehive of the scout bees (beehive 1) and beehive of the onlooker and worker bees (beehive 2). When the algorithm is started, beehive 1 generates N number of scout bees, where N represents the number of processors in the system. Each scout bee checks whether a processor is free or busy by execution of specific task on it. If a free resource is found the scout bee record the ID of the processor into the table of available resources and returns to the beehive 1, where it is terminates. The main purpose of the beehive 2 is to generate M number of onlooker bees, where M is the optimal number of parallel threads. After generation, the onlooker bees search in to table of available resources. If the onlooker

bee finds a free resource, it takes the ID of the processor and removes it from the table. If the onlooker bee didn`t find a free resource in the table, the bee will return to the beehive 2 and will be terminate. Once the onlooker bee takes the available resource it starts to behave as a worker bee. Thus obtained K number of worker bees initially turned to the table of outstanding tasks where they taking certain sub-problem, remove it from the table and submit it for the performance by the processor which ID has been taken from the table of available resources. After the processor solves a sub-problem, it provides the solution to a worker bee. The worker bee with the current solution returns to beehive 2, where it is terminated.

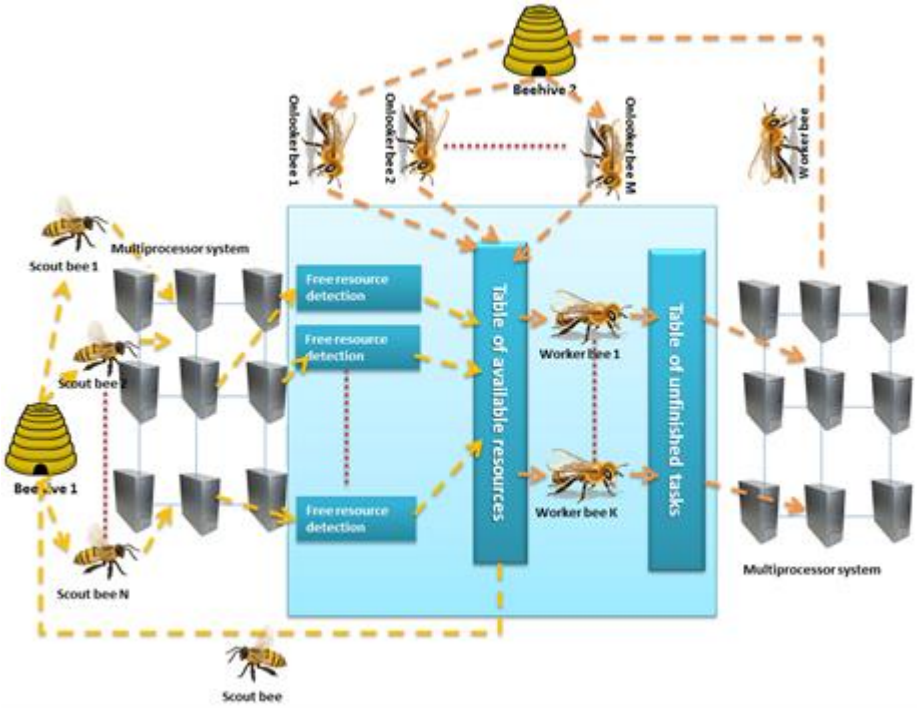


Fig. 1. Parallel computing model for effective resource allocation based on Artificial Bee Colony metaheuristic.

3.2. Pseudocode of the algorithm

The pseudocode of the proposed parallel algorithm for effective resource allocation based on artificial bee colony optimization technique is given below.

```
1. Input parameters:

    -  Nproc – Number of CPU`s/cores of the system;
```

- Nmax – Optimal number of parallel threads that will be generate to search for available resources;
- M – Optimal number of parallel threads that will process the table of available resources;
- ID() – an array that keep the ranges of the CPU`s/cores;
- T – optimal number of time slicing required for solving the main problems;
- Global data:
  - Tr – table of available resources;
  - Tz – table of outstanding tasks.

2. Output parameters:
   - Solutions of the main problems;

3. Functions:
   - AvailableResources()

   Generate Nmax parallel threads to check available resources;

   If (the resource is available)

      Record the ID of the processor at the table of available resources and terminate the thread;

   Else

      Terminate the thread;

   - SolvingSubProblems()

   Generate M parallel threads to search at the table of outstanding tasks;

   If (available resource is found)

      Record the ID of the processor and remove it from the table of available resources;

      Check the table of outstanding tasks;

      If (unsolved sub-problem is found);

          Proceed the unsolved sub-problem for execution to the processor with rang ID;

     Else

          flag = 1;

```
4. Algorithm framework:
BEGIN
Initialization of the input parameters;
flag ← 0;
If (Nproc ≤ Nmax)
     Nmax ← Nproc;
     Reserve P ← 2 CPU`s/Cores for the implementation of the
     algorithm;
Else
     Reserve P ← Ceiling((Nproc/Nmax)+1) CPU`s/Cores for the
     implementation of the algorithm;
Time Sliding 1 – CPU`s/Cores – {1 …. (P-1)} :
     AvailableResources();
Until flag=0 repeat
Time Slicing 2 …. K
     CPU`s/Cores - {1 …. (P-1)} - AvailableResources ();
     CPU/Core P - SolvingSubProblems ();
END.
```

4. Experimental Results

The experimental results were conducted by using two different multi-processor platforms. The first platform is represent by a homogeneous cluster composed of twelve Blade servers, HS21, Xeon Quad Core E405 80w 2.00GHz/1333MHz/12MB L2 and hard disk drive subsystems IBM 750GB Dual Port HS SATA HDD and Windows Server 2008 operating system. The second platform is represent by a heterogeneous cluster composed of eight Blade servers of which six are composed of Quad core E5405 2GHz processor, one composed of 2 x Quad core E5405 2GHz processor and one composed of Dual Core 5140 2.33 GHz processor. Each server also consist 16 GB of RAM memory and Windows Server 2008 operating system. The respective clusters are located at the High-Performance and GRID Computer Lab, Computer System Department, Technical University of Sofia.

4.1. Parallel implementation of the algorithm on homogeneous cluster

The experimental results were conducted by using the homogeneous cluster detailed descripted in above section. Parallel implementation of the algorithm was realized by using MPICH-2 message passing model and Intel TBB programming model built in

Intel Parallel Studio 2010. For virtualization of resources a virtual machine of Intel ArBB, built-in Intel Parallel Studio 2010 has been used.

The load of the computational resources while solving a package of three NP-Complete problems – Traveling Salesman Problem, the N-queens problem, and the Sam-Loyd puzzle are presented on figure 2. The package is started without the ABC optimization algorithm for effective resource allocation. From the charts shown on the figure, it is clear that the load of the processors while solving a package of three Np-Complete problems is not good balanced, because only at certain point of time processors have good load balance i.e. there are only few processors where the number of cores corresponds with the number of active processes. On the other hand, during the most of the time some of the processors have less number of active processes than cores, while some processors are overloaded i.e. the number of active processes exceeds number of cores in the processor. The package was started without using the proposed optimization algorithm.

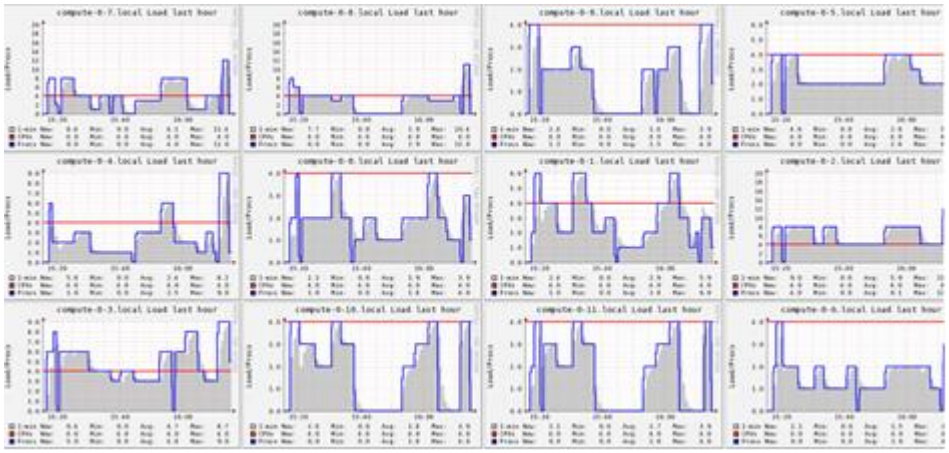


Fig. 2. CPU load while solving package of three NP-Complete problems without the ABC optimization algorithm on homogeneous cluster.

In order to improve this situation, the proposed algorithm for resource management optimization was implemented on the target platform. On figure 3, the load of the processors while solving a package of three NP-Complete problems by using the algorithm for resource management optimization based on ABC metaheuristic is shown.

According to the results shown on figure 3 it is clear that after the implementation of the optimization algorithm, the load of the processors is almost optimal as the overloading of the processors is avoided i.e. starting a bigger number of threads than cores on single processor, while dissatisfied load is presented only during the timeslots reserved for implementation of the algorithm for resource planning.

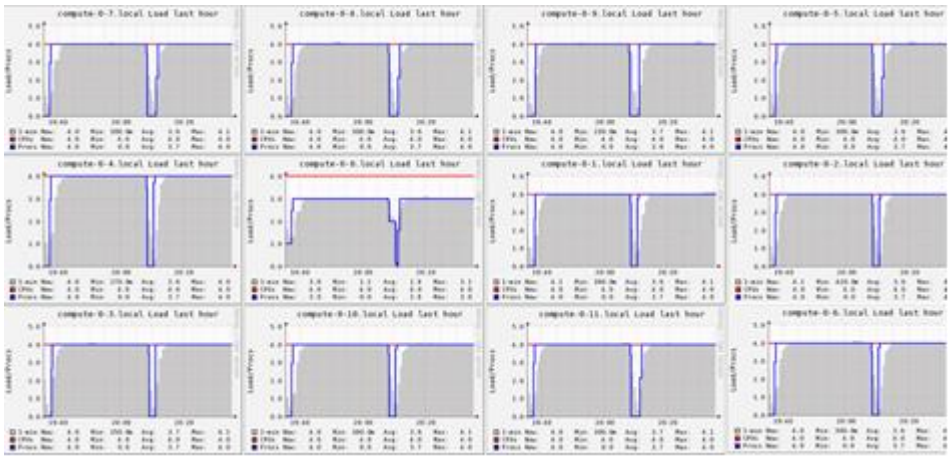
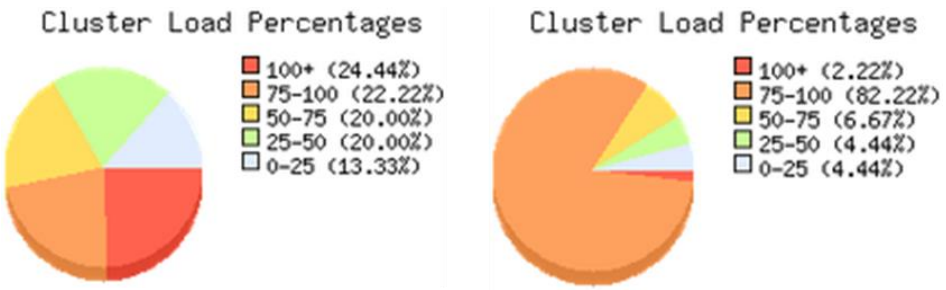


Fig. 3. CPU load while solving package of three NP-Complete problems by using the ABC optimization algorithm homogeneous cluster.

Figure 4 presented the load of the cluster during the execution of a package with three NP-Complete problems. During the execution of package with three NP-Complete problems without the proposed algorithm for resource management optimization, only 22,22% of the resources of the cluster have optimal load balancing with 75-100%, while the remaining resources are overloaded with 100%+ or not good loaded i.e. below 75%. On the other hand, during the execution of the package by using the algorithm for resource management optimization, 82.22% of the resources of the cluster have optimal load balancing and only 17.77% of the resources have poor load balance. These 17% of the resources with poor load balance appears mainly due to the time required for implementation of the algorithm for resource planning as well as other system costs of the system.



(a) (b)

Fig. 4. Cluster load during the execution of a package with three NP-Complete problems (a) without the optimization algorithm (b) by using the optimization algorithm.

4.2. Parallel implementation of the algorithm on heterogeneous cluster

The experimental results for these set experiments were conducted by using the heterogeneous cluster detailed descripted in section 3. Parallel implementation of the algorithm was realized by using the software tools descripted in section 3.1.

On figure 5, the load of the heterogeneous cluster while solving a package of three NP-Complete problems – Traveling Salesman Problem, the N-queens problem, and the Sam-Loyd puzzle is shown. The results are conducted when the package is started without proposed ABC optimization algorithm. According to the results presented on the figure below, the load of the cluster is not good balanced, and the problems are similar to the previous experiments. The worst load balance leads to decreasing of the parallel performance of the overall system.

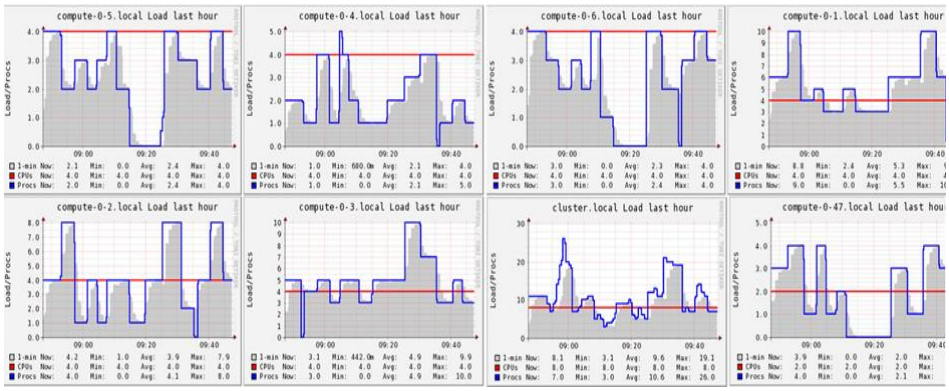


Fig. 5. CPU load while solving package of three NP-Complete problems without the ABC optimization algorithm on heterogeneous cluster.

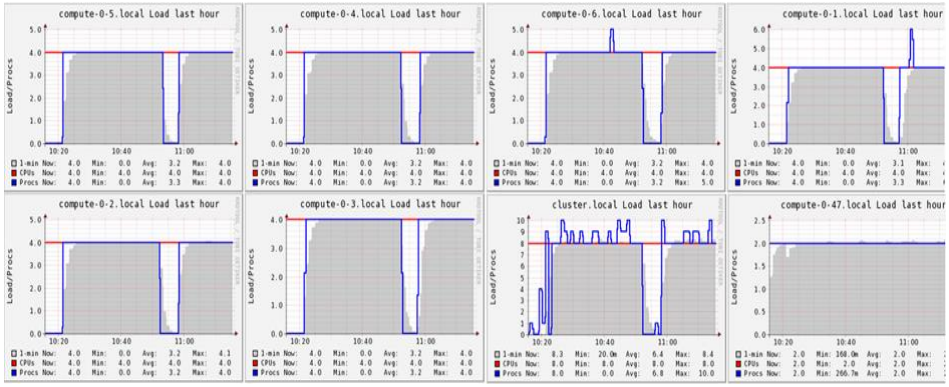
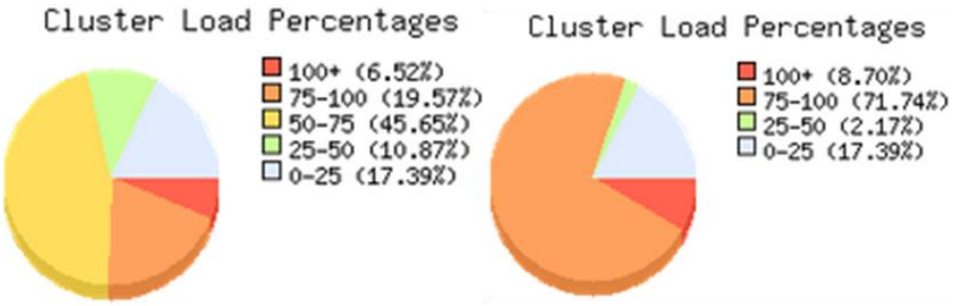


Fig. 6. CPU load while solving package of three NP-Complete problems by using the ABC optimization algorithm homogeneous cluster.

The load of the processors while solving a package of three NP-Complete problems by using the algorithm for resource management optimization based on ABC metaheuristic is presented on figure 6.

After the implementation of the proposed optimization algorithm, the load of the processors is almost optimal as the overloading of the processors is avoided while dissatisfied load is shown only during the timeslots reserved for the implementation of the algorithm.

On figure 7 the load of the cluster during the execution of a package with three NP-Complete problems on heterogeneous cluster is presented. During the execution of package with three NP-Complete problems without the proposed algorithm for resource management optimization, only 19,57% of the resources of the cluster have optimal load balancing with 75-100%, while the remaining resources are overloaded with 100%+ or not good loaded i.e. below 75%.



(a) (b)

Fig. 7. Cluster load during the execution of a package with three NP-Complete problems (a) without the optimization algorithm (b) by using the optimization algorithm.

On the other hand, during the execution of the package by using the algorithm for resource management optimization, 71.74% of the resources of the cluster have optimal load balancing and only 28.26% of the resources have poor load balance. These 28% of the resources with poor load balance appears mainly due to the time required for implementation of the ABC optimization algorithm as well as other system costs of the system.

5. Conclusion and Future Work

An effective resource utilization of the modern high performance computing (HPC) systems is a subject for many scientific research investigations. The resource management for those platforms is an essential part for optimal resource allocation while solving NP complete problems. An effective resource management algorithm strongly determines the overall parallel performance of the high-performance computing system.
This paper suggests an innovative algorithm for effective resource management in multi-processor platforms based on parallel metaheuristic “Artificial Bee Colony” (ABC)

optimization. The efficiency of the proposed optimization algorithm for effective resource allocation in homogeneous and heterogeneous platforms was evaluated on the basis of the software tools of Intel Array Building Blocks build-in Intel Parallel Studio.

Moreover, parallel programming implementations of three NP-Complete problems: the N-Queens problem, the Sam-Loyd puzzle, and the Traveling Salesman Problem (TSP) were executed in order to evaluate the overall parallel performance of the platforms. The proposed parallel implementations were developed on the basis of Message Passing Interface (MPI) and Intel Threading Building Blocks (TBB) programming models.

Finally, we applied the proposed algorithm a homogeneous cluster composed of twelve Blade servers HS21 and heterogeneous cluster composed of eight Blade servers. This allows us to observe the behavior of the cluster while simultaneously is started a package of three NP-Complete problems. From the experimental results we conclude that in the cases when the proposed algorithm is run on the target platform, the cluster has very good load balanced, which leads to increasing of the overall parallel performance of the system. Also the experimental results confirm that the proposed optimization algorithm is not platform-specific and can be easily migrate between different HPC systems.

Future objectives of this research include implementation of our algorithm on very large-scale systems and on the new generation of ExaScale machines.

Acknowledgment

The results reported in this paper are part of the research project, Center of excellence “Supercomputing Applications” - DCVP 02/1, supported by the National Science Fund, Bulgarian Ministry of Education and Science.

References

1. Kristof, P., Yu, H., Li, Z., Tian, X.: Performance study of SIMD programming models on intel multicore processors. In Proceedings of the IEEE 26th International Symposium Workshops & PhD Forum (IPDPSW) on Parallel and Distributed Processing, 2423-2432. (2012)
2. Kim, W., Voss, M.: Multicore desktop programming with intel threading building blocks. IEEE software, Vol. 28, No.1, 23-31. (2011)
3. Newburn, C. J., So, B., Liu, Z., McCool, M., Ghuloum, A., Toit, S. D., Guo, P.: Intel's Array Building Blocks: A retargetable, dynamic compiler and embedded language. In Proceedings of the 9th annual IEEE/ACM international symposium on Code generation and optimization (CGO), 224-235. (2011)
4. F. Cicirelli, G. Folino, A. Forestiero, A. Giordano, C. Mastroianni, G. Spezzano: Strategies for Parallelizing Swarm Intelligence Algorithms. In Proceedings of the 23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP), 329 – 336. (2015)

5. Banharnsakun, A., Achalakul, T., Sirinaovakul, B.: Artificial bee colony algorithm on distributed environments. In Proceedings of the Second World Congress on Nature and Biologically Inspired Computing, 13 – 18. (2010)
6. Marinakis, Y., Marinaki, M., Matsatsinis, N.: A hybrid discrete Artificial Bee Colony - GRASP algorithm for clustering. In Proceedings of the International Conference on Computers & Industrial Engineering, 548 – 553. (2009)
7. Ming, Z., Xiao-Yu, S., Yi-Chen, G.: Emergency scheduling optimization based on improved artificial bee colony algorithm. In Proceedings of the 6th IEEE International Conference on Software Engineering and Service Science (ICSESS), 886 – 889. (2015)
8. Loganathan, J., Veronica, A.: Distributed resource management scheme using enhanced artificial bee-colony in P2P. In Proceedings of the 2nd International Conference on Electronics and Communication Systems (ICECS), 1035 – 1039. (2015)
9. Jin-Qiu, Y., Jian-Gang, Y., Gen-Lang, C.: Solving Large-Scale TSP Using Adaptive Clustering Method. In Proceedings of the Second International Symposium on Computational Intelligence and Design, 49 – 51. (2009)
10. Khademzadeh, A., Sharbaf, M.A., Bayati, A.: An Optimized MPI-based Approach for Solving the N-Queens Problem. In Proceedings of the 7th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 119 – 124. (2012)
11. Bishop, K.A., Madsen, D. L.: Demonstration of a Solution Algorithm to the Sam Loyd Puzzle of NxN. (2009)
12. Ying, S., Hui, W., Yaqiong, L., Binquan, F.: Multi-Tiered On-Demand Resource Scheduling for VM-Based Data Center. In Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 148 – 155. (2009)
13. Zhenzhong, Z., Limin, X., Yongnan, L., Li, R.: A VM-based Resource Management Method Using Statistics. In Proceedings of the IEEE 18th International Conference on Parallel and Distributed Systems, 788 - 793. (2012)
14. Bini, E.: Resource Management on Multicore Systems: The ACTORS Approach. Micro, IEEE journal, Vol. 31, No. 3, 72 – 81. (2011)