NESUS

cost IC1305

Network for Sustainable Ultrascale Computing

Proceedings of the Second International Workshop on Sustainable Ultrascale Computing Systems (NESUS 2015) Krakow, Poland

Jesus Carretero, Javier Garcia Blas
Roman Wyrzykowski, Emmanuel Jeannot.
(Editors)

September 10-11, 2015

Nature-Inspired Algorithm for Solving NP-Complete Problems

Atanas Hristov
University of Information Science and Technology, Ohrid, Macedonia
atanas.hristov@uist.edu.mk

Abstract

High-Performance Computing has become an essential tool in numerous natural sciences. The modern 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. A core challenge while solving NP-complete problems is the need to process these data with highly effective algorithms and tools where the computational costs grow exponentially. This paper investigates the efficiency of Nature-Inspired optimization algorithm for solving NP-complete problems, based on Artificial Bee Colony (ABC) metaheuristic. 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 Artificial Bee Colony, High-Performance Computing, Parallel Algorithm, NP-complete problems, message passing, multithreading

I. Introduction

Accelerating the development and deployment of advanced computing systems and cloud computing platforms will require a comprehensive strategy integrating efforts from invention to deployment. The modern high-performance computing systems (HPCS) 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. Currently, most of the HPCS 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, 2, 3].

The complexity class of decision problems NP-complete can be used as a pattern for benchmarking and parallel performance evaluation of HPCS. This paper investigates the efficiency of Nature-Inspired optimization algorithm for solving NP-complete problems, based on Artificial Bee Colony (ABC) metaheuristic. Highly parallel version of the well-known N-queens problem has been proposed based on ABC optimization. The 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. 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.

II. Background

The complexity class of decision problems NP-complete can be used as a pattern for benchmarking and parallel performance evaluation of HPCS. The main idea behind using NP-complete problems for evaluation of the overall parallel performance of the HPCS is that those problems cannot be solved in polynomial time in any known way which require high computational power and time. The N-queens problem belongs to the class of NP-complete problems, requiring a brute-force algorithm for finding all possible solutions. The N-queens problem is formulated as solving the task to place N queens on N x N chessboard in such way that no queens attack each other, i.e. on every row, column or diagonal, there is only one queen. The complexity of this algorithm is O(N!), which comes from the fact that there are (N2!)/(N!*(N2-N)!) possible solutions to place the queens on the board [4, 5, 6]. In order to provide efficient solutions for this problem and to minimize the time and space complexity, many various optimization techniques have been proposed. In this paper we provide the experimental results gained by solving N-Queens problem, using Artificial Bee Colony (ABC) optimization. The ABC algorithm belongs to the class of nature-inspired algorithms, which simulate the behavior of the honey bee swarms in the nature. The main advantage of ABC algorithm is that uses only common control parameters such as colony size and maximum cycle number. The ABC algorithm is very powerful optimization tool which provide a population based search procedure. The ABC algorithm also combine local search methods, by using artificial bees which fly around multidimensional search space, with global search methods, by using another kind of artificial bees which fly and choose the food source randomly without any experience and memorize the new position if it is a better than the one that is already in their memory. Thus lead to balancing of the exploration and exploitation process. The main idea behind using the ABC algorithm for solving the N-Queens problem is that ABC is very effective optimization algorithm for finding the best optimized solution, which is declared according to the position of the food source, and the amount of nectar found in that solution. [7, 8, 9].

III. Resource planning while solving NP-complete problems

During the resource planning process while solving NP-complete problems two strategies can be applied: static and dynamic [10]. In the static resource planning process, each node executes only one part of the search tree. On Figure 1, an example of static resource planning on octal-core platform is given.

The main problem with static resource planning strategy is that the spatial search trees are highly unbalanced. This can lead to significantly reduce of the parallel performance of the application. This strategy can be improved by simultaneously scanning of larger number of subtrees in order to balance the load of computational nodes.

Dynamic resource planning strategies for solving NP-complete problems, allow scanning a large number of subtrees simultaneously, providing better load balancing and higher parallel efficiency. In this paper two dynamic strategies have been proposed. The first strategy proposed a model in which the main process searching in depth the spatial search tree and the gener-

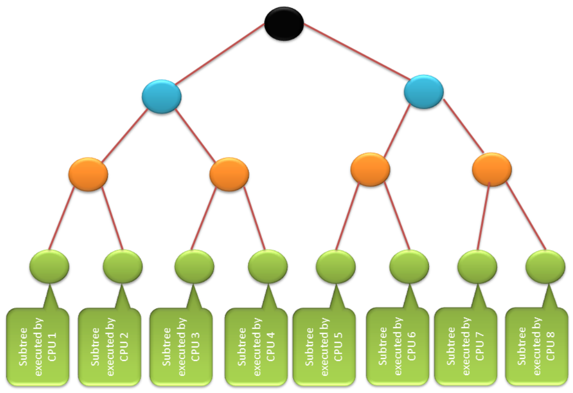
Subtree executed by CPU 1
Subtree executed by CPU 2
Subtree executed by CPU 3
Subtree executed by CPU 4
Subtree executed by CPU 5
Subtree executed by CPU 6
Subtree executed by CPU 7
Subtree executed by CPU 8

Figure 1: Static resource planning while solving NP-complete problems.

ated subtrees from this process are distributed among all available processes of the system.

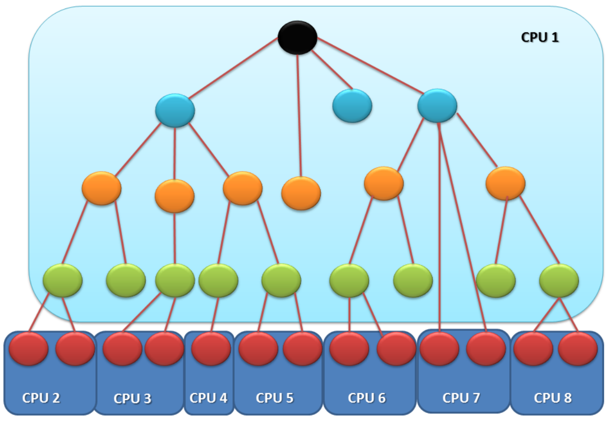

Figure 2: Depth search dynamic resource planning strategy while solving NP-complete problems.

On Figure 2, an example of dynamic resource planning by using depth search strategy on octal-core platform is presented. Thus the probability of an unbalanced load significantly reduced, while potential parallel performance of the system dramatically increases.

Another strategy for dynamic resource planning is by maintaining a list of nodes of the spatial search tree. All problems and subproblems from the spatial search tree are placed in the list and each process takes the last unprocessed knot.

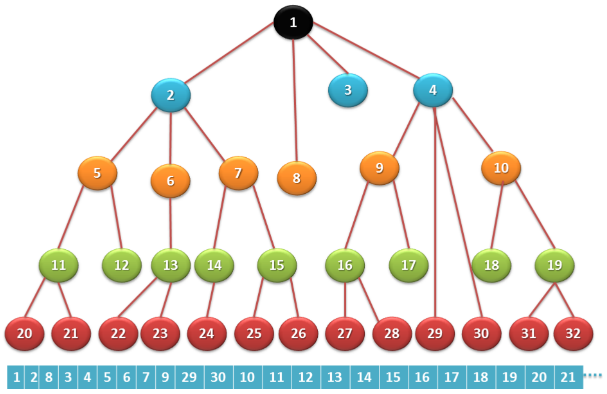

Figure 3: Maintaining a list of nodes dynamic resource planning strategy while solving NP-complete problems.

On Figure 3, an example of dynamic resource planning by using maintaining a list of nodes dynamic strategy is presented. This strategy provides an efficient way to allocate the workload among computational nodes of the system even in highly unbalanced trees of the search space. Thus the efficiency of the parallel algorithm, as well as the potential parallel performance of the system does not depend on the balance of the tree. The main disadvantage of this strategy is that it requires additional system resources for maintenance and searching in the list of nodes.

IV. Parallel implementation of ABC algorithm

An effective resource utilization of the modern high performance computing 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 hard problems. The proposed algorithm for solving NP-Complete problems is based on Artificial Bee Colony (ABC) metaheuristic. In this paper we will present a nature-inspired approach for solving the N-queens problem which belongs to the class of

NP-complete problems. The ABC simulates the collective behavior of the honeybees in nature. 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 solving N-queens problem based on Artificial Bee Colony (ABC) metaheuristic is present on Figure 4.

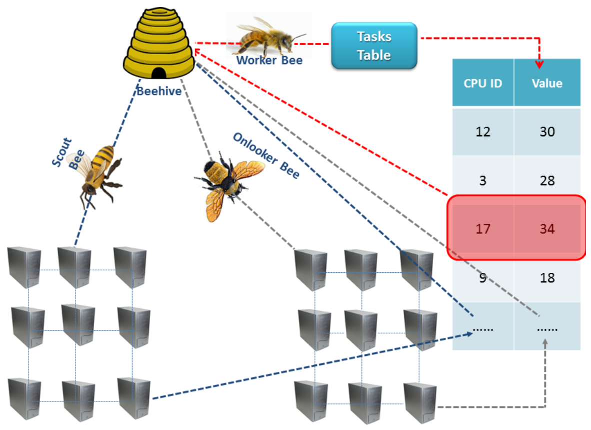

Figure 4: Parallel computing model for for solving N-queens problem based on Artificial Bee Colony metaheuristic.

In the proposed parallel model the number of all possible solutions should be determinate, which technically is the number of employed bees. Initially, a random population is generated, followed by repeating cycles of searching for employed, scout, and onlooker bees. The employed bee makes a modification of its initial food source in her memory and therefore finds a new food source. If the amount and quality of the nectar from the second position is better than, the employed bee forgets its initial food source. When the employed bees finish the search process, they pass their information to the onlookers located at the base. Then each onlooker makes a calculation on the received information and determines the food source. In order to implement the parallel algorithm for solving the N-Queens problem, the IntelTBB 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 located in beehive, so call beehive of the scout bees and beehive of the onlooker and worker bees. Initially, the main problem is divided in several sub-problems, which are stored to the table of outstanding tasks. When the algorithm is started, the beehive 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. Also by executing specific piece of code, the scout bee determinate the value of the processor, which basically evaluate the suitability of the processor to execute specific tasks. Depending of the suitability of the processor, the scout bee gains a value to the processor. Once these operations are done, the scout bee returns to the beehive, where it is terminates. On the next step, the beehive generates 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. The priority is given to the processor with highest value from the table. If the onlooker bee didn't find any free resource in the table, the bee will return to the beehive 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 execution by the processor which ID has been taken from the table of available resources. Once 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.

V. Experimental Results

The proposed algorithm for parallel solving of the N-queens problem, based on metaheuristic ABC, is verified and its effectiveness has been studied experimentally based on multithreaded implementation using Intel Threading Building Blocks (TBB) programming model. The target multiprocessor platform for conducting experimental results is IBM Blade HS22 server with two quad-core processors Xeon Quad Core 2.00GHz, 6GB RAM, operating system Windows Server 2008. For implementation of the parallel algorithm and TBB programming model, Intel Parallel Studio 2010 program environment have been used. Experimental results were conducted for a different workload, i.e. for different size of the chessboard: 8x8, 12x12, 14x14, and 16x16. Also, two parallel versions of N-Queens problem have been tested: the first one based on the proposed ABC algorithm and the second one based on well-known backtracking algorithm. On Figure 5 the executional time while solving the N-Queens problem using sequential, ABC, and backtracking algorithm is given.

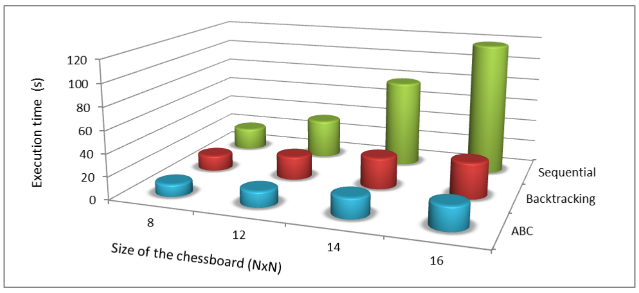

Figure 5: Executional time while solving the N-Queens problem using sequential, ABC, and backtracking algorithm.

From the chart shown on Figure 2 can be concluded that for size of the chessboard up to 10x10, the executional times of serial and parallel implementation of the program are relatively close due to the very short calculation time of the problem. The overall calculation time for the mention size of the chessboard is in the range of few milliseconds up to few seconds. On the other hand, when the size of the chessboard increases, the number of possible optimal and suboptimal solutions growths exponentially. The execution time of the sequential algorithm also growths exponentially, but the potential parallelism of the application increases for a given factor. This leads to very high executional time for sequential algorithm, and slightly increases in the parallel execution of the program.

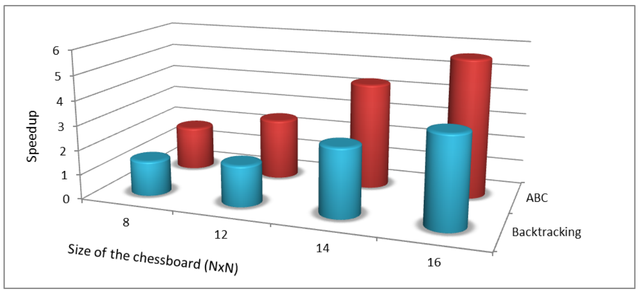

Figure 6: Executional time while solving the N-Queens problem using sequential, ABC, and backtracking algorithm.

On Figure 6 the speedup gained while solving the N-Queens problem using ABC and backtracking algorithm is given. When the size of the board increases to 16x16, the speedup gained by program implementation on octal-core platform using backtracking search algorithm is 3.68, while using ABC algorithm 5.2. The main reason behind this is that the ABC algorithm has better workload balance and better data structure in the search space through creating tables with tasks and adequate resources allocation of the relevant subproblems implementation.

VI. 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 parallel algorithm strongly determines the overall parallel performance of the high-performance computing system. This paper suggests an innovative algorithm for solving N-queens problem on multi-processor platforms based on parallel metaheuristic "Artificial Bee Colony" (ABC) optimization. The efficiency of the proposed algorithm was evaluated on the basis of the software tools of Intel Array Building Blocks build-in Intel Parallel Studio. The proposed parallel implementation was developed on the basis of Message Passing Interface (MPI) and Intel Threading Building Blocks (TBB) programming models. Finally, we applied the proposed algorithm on IBM Blade HS22 server with two quad-core processors. This allows us to observe the behavior of the cluster while solving the N-queens problem. From the experimental results we conclude that the speedup gained by program implementation on octal-core platform using backtracking search algorithm is 3.68, while using ABC algorithm 5.2. 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., Hongtao Yu, Zhiyuan Li, and Tian X., “Performance Study of SIMD Programming Models on Intel Multicore Processors,” in *26th International Symposium in Parallel and Distributed Processing*, Shanghai, China, 21-25 May 2012, pp. 2423-2432.

[2] Wooyoung Kim and Voss M., "Multicore Desktop Programming with Intel Threading Building Blocks," *IEEE Software journal* , vol. 28, no. 1, pp. 23-31, 2011.

[3] Newburn C.J., Byoungro So, Zhenying Liu, McCool M., Ghuloum A., and Toit S.D., “Intel’s Array Building Blocks: A retargetable, dynamic compiler and embedded language,” in *9th Annual IEEE/ACM International Symposium on Code Generation and Optimization*, Chamonix, France, 02-06 April 2011, pp. 224-235.

[4] Khademzadeh A., Sharbaf M.A., and Bayati A., “An Optimized MPI-based Approach for Solving the N-Queens Problem.,” in *7th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, Victoria, BC,Canada, 12-14 November 2012, pp. 119-124.

[5] P. Panwar, V. P. Saxena, A. Sharma, and V. Sharma, " Load Balancing using N-Queens Problem," *International Journal of Engineering Research Technology* , vol. 2, no. 1, 2012.

[6] Ayala A., Osman H., Shapiro D., and Desmarais J.M., “Accelerating N-queens problem using OpenMP,” in *6th IEEE International Symposium on Applied Computational Intelligence and Informatics*, Timisoara, Romania, 19-21 May 2011, pp. 535-539.

[7] Teodorovic D., Lucic P., and Markovic, G., “Bee Colony Optimization: Principles and Applications,” in *8th Seminar on Neural Network Applications in Electrical Engineering*, Belgrade, Serbia, 25-27 September 2006, pp. 151-15.

[8] Banharnsakun A., Achalakul T., and Sirinaovakul B., “Artificial bee colony algorithm on distributed environments,” in *Second World Congress on Nature and Biologically Inspired Computing*, Fukuoka, Japan, 15-17 December 2010, pp. 13-18.

[9] Marinakis Y., Marinaki M., and Matsatsinis N., “A hybrid discrete Artificial Bee Colony - GRASP algorithm for clustering,” in *International Conference on Computers Industrial Engineering*, Troyes, France, 6-9 July 2009, pp. 548-553.

[10] Xiaozhong G.,Gaochao Xu, and Yuan Z., “Dynamic Load Balancing Scheduling Model Based on Multi-core Processor,” in *Fifth International Conference on Frontier of Computer Science and Technology*, Changchun, Jilin Province, 18-22 August 2010, pp. 398-403.