Efficiency of Parallel Genetic Algorithm for Solving N-Queens Problem on Multicomputer Platform

MILENA LAZAROVA
Department of Computer Systems
Technical University of Sofia
8, Kliment Ohridski St., Sofia 1000
BULGARIA

Abstract: The paper investigates the efficiency of parallel genetic algorithm for solving N-queens problem on a multicomputer platform. The proposed parallel computational model of the genetic algorithm is based on a parallel algorithmic paradigm of synchronous iterations. Dynamic migration of randomly selected chromosomes in a bidirectional circular model is utilized. The algorithm is implemented using both flat (pure MPI) and hybrid (MPI+OpenMP) programming models. The target parallel multicomputer platform is a cluster of SMPs. Performance profiling and scalability analyses have been made in respect of both the workload (board size) and the size of the parallel system.

Key-Words: N-queens problem, parallel genetic algorithm, island model, dynamic migration policy, multicomputer platform, profiling and performance analysis

1 Introduction

The N-queens problem is formulated as solving the task to place N queens on a N×N chessboard in such way that no queens attack each other. This a classical combinatorial search problem initially considered as computer toy problem but recently has also found practical scientific and engineering applications in the field of parallel memory storage schemes and parallel optical computing, VLSI testing, traffic control, deadlock prevention. The difficulty of the problem arises from the fact that there are $(N^2!)/((N!(N^2–N)!)$ possible ways of placing the queens on the board and only a very small number of them represent actual solutions [1]. Therefore the search space of the solutions is incredibly large even for small values of N.

One possible strategy for solving constraint satisfaction problems, such as N-queens problem, organizes the solution space into a tree and systematically searches this tree for the answer [2]. Dynamic programming, heuristic search techniques and neural networks are some of the approaches applied for solving N-queens problem [3, 4, 5].

Genetic algorithm (GA) is probabilistic, heuristic-based method for search of a sub-optimal solution in large search spaces of complex optimization problems [6, 7]. GAs are founded on the ideas of evolutionary processes in the biological individuals and are successfully applied in solving optimization and constraint satisfaction problems [8, 9].

There are several parallel genetic approaches that are used to reduce the large amount of computation time associated with the serial genetic algorithms [10, 11, 12]. One method of building a parallel genetic algorithm (PGA) is a global parallelization strategy that applies the master/workers paradigm [13, 14]. Island parallel genetic model implies migration between independently evolving populations on parallel processes in order to speedup the slowly evolving subpopulations by introducing chromosomes that are better than the locally best ones [15, 16].

There are several dynamic migration policies leading to different speed of distribution of the migrants between the "islands" [17]. The evidences of a higher efficiency, larger diversity maintenance, additional availability of memory/CPU, and their multi-solution capabilities, reinforce the importance of the research advances with PGAs [18]. The island genetic algorithms can easily be implemented on a multicomputer platform with distributed memory. Periodic migration of chromosomes can be involved in order to speedup the subpopulation evolution by increasing local diversity [19, 20].

The objective of this paper is to investigate the efficiency of parallel genetic algorithm for solving N-queens problem on a multicomputer platform. Dynamic migration of randomly selected chromosomes in a bidirectional circular model is utilized. The proposed parallel computational model of the genetic algorithm is based on a parallel algorithmic paradigm of parallel synchronous iterations. The algorithm is implemented using both flat (pure MPI) and hybrid (MPI+OpenMP) programming models. The target parallel multicomputer platform is a cluster of SMPs. Performance profiling, evaluation and analysis have been made for different workload (board size) and different sizes of the multicomputer platform.

2 Solving N-Queens Problem with Genetic Algorithm

The problem addressed requires N queens to be placed on a N×N chessboard so that no one attacks the others. The problem is an extension of the 8-queens problem originally introduced in 1850 by Carl Gauss. The total number of possible solutions for the 8-queens problem is 92 and 12 of them are unique, the rest can be derived by symmetric operations. The complexity of the N-queens problem is of O(N!) and the problem belong to the class of NP-complete problems requiring a brute-force algorithm to guarantee that the solution can be found for any value of N.

The basic classes of strategy for the N-queens constraint satisfaction problem are [2]: systematic search strategies – put one queen onto the board at a time and make sure that no constraint is violated, until all eight queens are placed, and repair strategies – put all eight queens onto the board initially at random and if any queen threatens another try to move it to a new place. A depth-first search backtracking algorithm can solve the N-Queens problem in reasonable time but only for small values of N.

A number of efforts have been made for efficient methods for solving the N-queens problem using various heuristics approaches including iterative local search, simulated annealing, tabu-search, genetic algorithm [8, 21, 22, 23, 24]. Synchronous global parallel genetic algorithm for the N-queens problem is discussed and evaluated on a single-processor platform [25].

Genetic algorithms provide search technique in computer science to find approximate solutions to optimization and search problems and present a particular class of evolutionary algorithms [26. A general sequence of steps for solving the optimization problem by a genetic algorithm is illustrated on fig.1.

Initial population is usually generated randomly providing given number of possible solutions to the problem. In the case of N-queens problem initial population will comprises randomly generated placements of the queens on the board represented as permutations of an N-tuple (1, 2, 3, …, N). A chromosome *i* shows the column where the queen in row *i* is placed. The fitness of each individual measures how close it is to the problem solution. Since a solution to the N-queens problem requires no queens to attack each other the fitness is calculated as the number of conflicts between queens. The individuals in the population are evaluated and sorted according to their fitness. A crossover process creates new individuals combining two parents. Mutation involves a random change of an individual that is exchange of the positions of two queens. Genetic algorithm finishes the search either if a solution of the queens' placement on the board is found or a predefined number of iterations is accomplished.

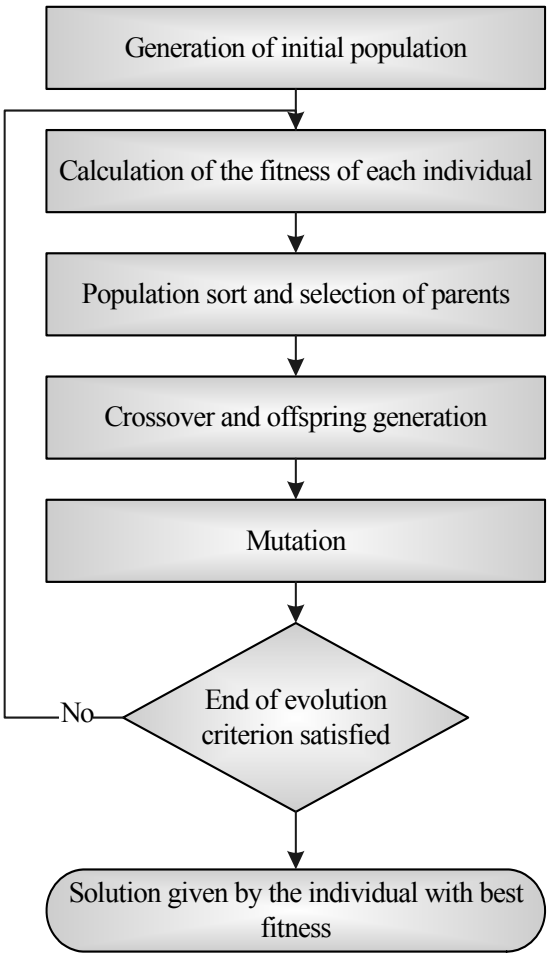

Fig.1. Genetic algorithm sequence for solving optimization problems

3 Parallel Computational Model of GA for Solving N-Queens problem

The way in which GAs can be parallelised depends on several elements [27]: evaluation of fitness; application of mutation; use of single or multiple subpopulations (demes); way of individuals exchange; global or local application of selection.

The suggested parallel computational model for solving N-queens problem (fig.2) utilizes parallelization method that divides the population into some number of demes (subpopulations) that are separated and evolve independently. The parallel computational model utilizes a parallel algorithmic paradigm "synchronous iterations". Each process evolves a subpopulation performing the genetic operations selection, crossover and mutation. Iterations of independent genetic evolution on each of the processes for certain number of generations are followed by a communication stage for migration of the best solutions found so far between the processes.

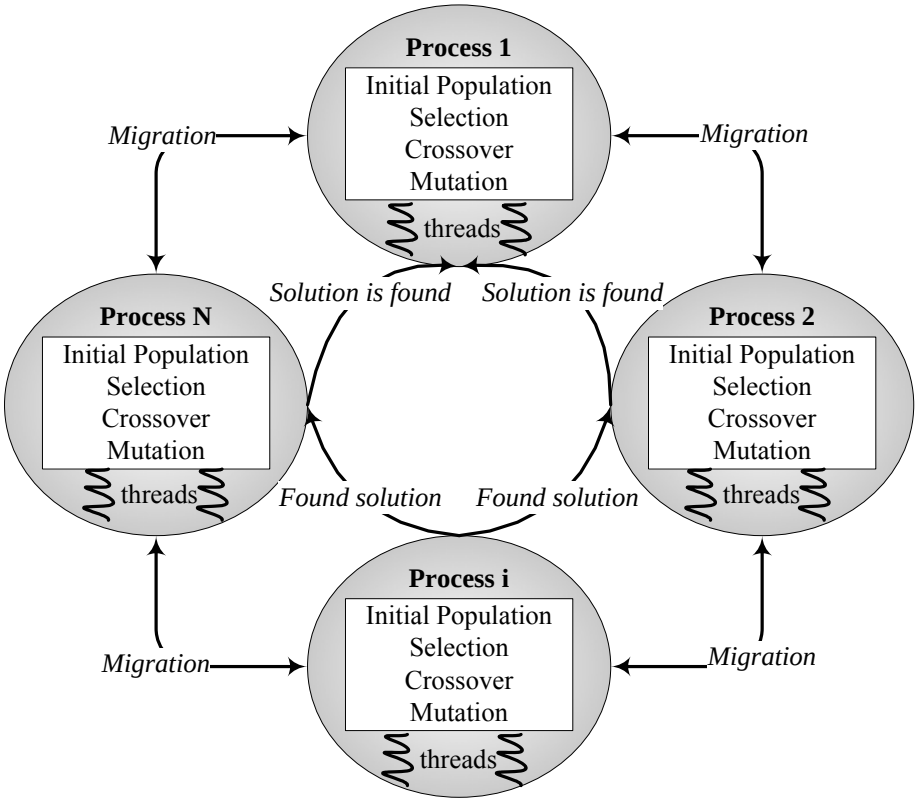

Fig.2. Parallel computational model

There are several parameters that control the migration of individuals:

- the topology of the connections between the processes that concurrently evolve the subpopulations;
- a migration rate that controls how many individuals migrate between the subpopulations;
- a migration scheme that controls which individuals from the source subpopulation (best, worst, random) migrate to another subpopulation and which individuals are replaced (worst, random, etc.);
- a migration interval that determines the frequency of migrations.

Commonly used topologies of the connections between the processes include ring, hypercube, two- or three-dimensional mesh, torus, etc. A master-slave policy of migration considers one of the islands as a dynamic sender of globally selected migrant chromosomes to all other islands causing "immediate" migration of individuals via global communication functions (MPI_Reduce, MPI_Bcast).

In the suggested parallel computational model "stepping stone" model of migration is utilized. The processes are organized in a logical ring and the migration involves bidirectional circular periodic chromosome movement that is each processor sends and receives certain amount of migrants to and from both neighbors in the ring. The advantage of this limited migration is small communication overhead introduced by the migration process compared to the master-slave migration policy. On the other hand the choice of ring topology with bidirectional migration paths leads to higher diversity of genetic material that is distributed over the islands preserving the evolution from fast convergence to a local optimum and maintaining better coverage of the search space throughout the parallel evolution process.

Because of the task is to find any solution and not all possible solutions of the N-queens problem when a process founds a solution it sends a message with a tag "SOLUTION" to its neighbors in the ring. If any process receives a message with a tag "SOLUTION" it sends the same message to its other neighbor and terminates itself.

Instead of utilization of a master process to generate and distribute unique initial subpopulations to the slaves, each process generates its initial subpopulation. In order to provide a high diversification of the independent parallel evolutions that are concurrent searches of different search sub-spaces, parallel random number generators is utilized for the generation of initial subpopulation. In the case of domain decomposition parallel algorithm requires fixed number of generators, equal to the number of parallel working processes. The leapfrog method can be used to generate random sequences that can be guaranteed not to overlap for a certain period. The leapfrog method is similar to a cyclic allocation of data to tasks: starting from one and the same sequential number generator each process with rank r takes every pth element of the sequence beginning with x_r.

4 Parallelism Profiling and Performance Analysis

The suggested parallel computational model for solving N-queens problem using island-based parallel genetic algorithm with bidirectional migration was implemented in C++ using Microsoft Visual Studio 2005 compiler, MPICH-2 as an implementation of the standard for message passing MPI, OpenMP was utilized to employ fork-join parallelism at a fine-grained level and Jumpshot v.3.0 is a tool for communication profiling.

The experiments for estimation of the performance parameters of solving N-queens problem by parallel genetic algorithm on a multicomputer platform are performed on a cluster comprising 10 workstations (Intel Pentium IV 3.2GHz, 1GB RAM, hyperthreading) connected by Fast Ethernet 100 Mbps switch. Since several runs were carried out for each tested case the total number of experiments conducted was about 100.

The experimental constant genetic parameters are given in table 1. Experiments were carried out with a board size 16×16 and 17×17.

Table 1. Genetic parameters

Generations	500
Population size	1000
Subpopulation size	population size / number of processes
Cross-over	random single-point
Mutation probability	0.3%
Migration topology	bidirectional ring
Migration period	50 generations
Number of migrants	20% of the subpopulation size

Both flat parallel model utilizing communication between parallel working processes in the multicomputer via message passing (MPI) and hybrid parallel model (MPI+OpenMP) implying multilevel parallelism: message passing between processes running concurrent evolution of the subpopulations and fork-join (multithreading) calculations of the genetic operations in each process.

Comparison of the results obtained for the speedup and the efficiency of both flat and hybrid implementations of the PGA are shown in fig.3 and fig.4 respectively. The results show good scalability of the parallel computational model in respect of both the parallel workload (board size) and the size of the cluster. The application scales well in respect to the size of the multicomputer with approximately proportional speedup and high efficiency of the parallel system. The speedup increases with the increase of the parallel machine size and it is 6.9 when 10 processors are utilized for finding solution on board size 17×17. Obviously, the hybrid implementation outperforms the flat implementation in respect to the speedup and hardware resources utilization – the speedup is further increased to 8.7 for the hybrid implementation of the case with 17 queens that is explained by more efficient utilization of the computational resources and reduction of the communication overhead due to employment of the shared memory programming model.

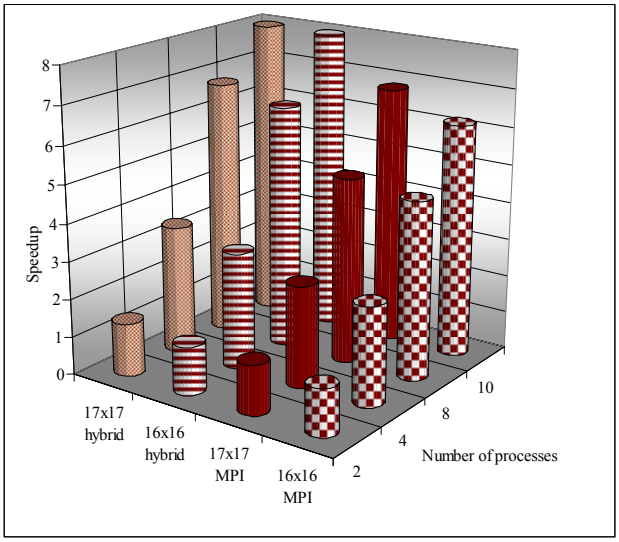

Fig.3. Speedup of the parallel flat and hybrid implementations of the PGA

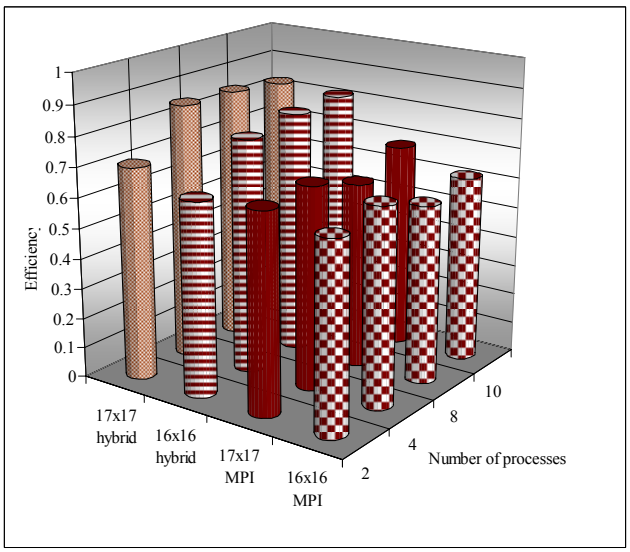

Fig.4. Efficiency of the parallel flat and hybrid implementations of the PGA

Gantt's chart for the parallel genetic computation on a multicomputer of ten workstations, showing communication transactions according to the suggested parallel computational model is given in Fig.5. The connected states of the communications corresponding to the behavior during the bidirectional migration process in a ring topology are presented in Fig.6.

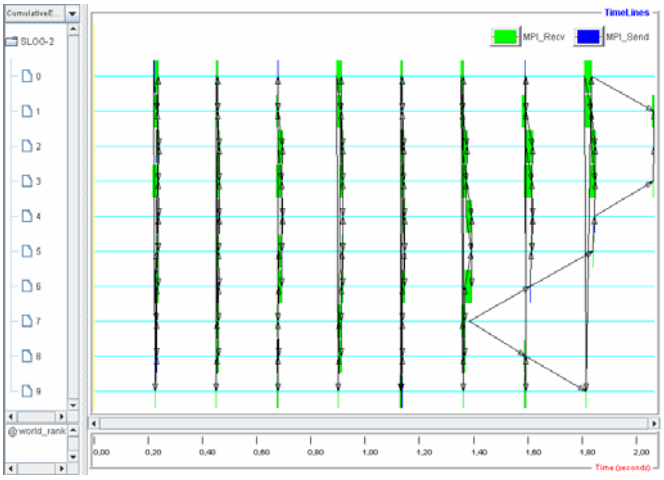

Fig.5. Gantt's chart of the island-based PGA for solving N-queens problem using periodic migration

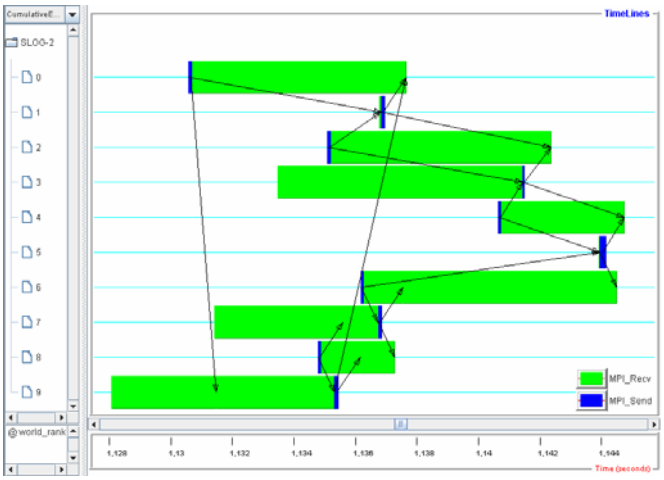

Fig.6. Communication transactions during bi-directional migration process

The communication transactions when one of process has found a solution are given in fig.7. In this case process 7 has found a solution and therefore it initiates a termination message exchange by sending messages to both its neighbors.

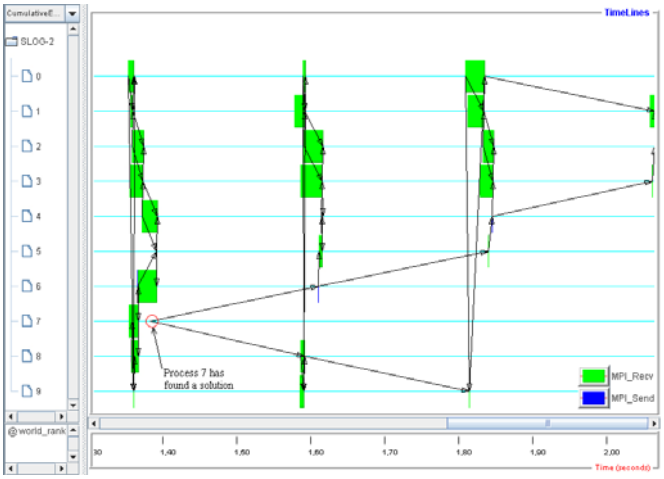

Fig.7. Communication transaction for termination of the genetic search

A histogram of the communication profile of each process is presented in fig.8 showing the total time spent for communications carried out by each process. As can be seen the time spent for data exchange during periodic chromosome migration and during termination messages exchange is much less than the computational time. This explains the values of the speedup obtained by the parallel computational model of PGA.

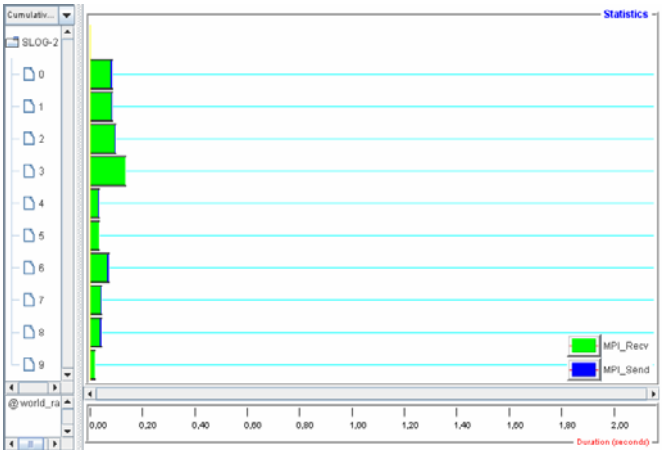

Fig.8. Histogram of the communication profile of each process

5 Conclusion

The paper investigates the efficiency of parallel genetic algorithm for solving N-queens problem on a multicomputer platform.

The suggested parallel computational model is based on a parallel algorithmic paradigm of parallel synchronous iterations. Island-based parallel genetic algorithm with bidirectional dynamic migration of randomly selected chromosomes in a ring topology is utilized.

The algorithm is implemented using both flat (pure MPI) and hybrid (MPI+OpenMP) programming models. The target parallel multicomputer platform is a cluster of SMPs. Performance profiling, evaluation and analysis have been made for different workload (board size) and different size of the multicomputer platform.

The results show good scalability of the parallel computational model in respect of both the parallel workload (board size) and the size of the cluster. The application scales well in respect to the size of the multicomputer with approximately proportional speedup and high efficiency of the parallel system.

Performance comparison shows that the hybrid parallel programming model better utilizes parallel hardware resources of the target multicomputer platform.

Future work should involve investigation of strategies for control of some genetic algorithm parameters, e.g. mutation rate and migration size, on the performance of the parallel computational model.

ISBN: 978-960-6766-58-9

ISSN: 1790-5109

References:

[1] J. Watkins, *Across the Board: The Mathematics of Chessboard Problems*, Princeton University Press, 2004.

[2] E. Tsang, A Glimpse of Constraint Satisfaction, *Artificial Intelligence Review*, Vol.13, №3, Kluwer Academic Publishers, 1999, pp.215÷227.

[3] I. Rivin, R. Zabih. A Dynamic Programming Solution to the N-queens Problem, *Information Processing Letters*, Vol.41, 1992, pp.253÷256.

[4] J. Mandziuk, B. Macuk, A Neural Network Designed to Solve the N-Queens Problem, *Journal Biological Cybernetics*, Vol.66, №4, 1992, pp.375÷379.

[5] I. Silva, A. Souza, M. Bordon, A modified Hopfield model for solving the N-Queens problem, *Proc. of the IEEE Int. Joint Conference on Neural Networks*, Vol.6, 2000, pp.509÷514.

[6] C. Reeves, J. Rowe, *Genetic Algorithms – Principles and Perspectives: A Guide to GA Theory*, Springer, 2002.

[7] M. Kantardric, *Data Minig: Concepts, Models, Methods and Algorithms*, John Wiley & Song, 2003.

[8] A. Eiben, P. Raue, Z. Ruttkay, GA-Easy and GA-Hard Constraint Satisfaction Problems, M. Meyer, (ed.), *Proc. of European Conference on Artificial Intelligence*, Amsterdam, 1994, pp.267÷283.

[9] A. Eiben, Evolutionary Algorithms and Constraints Satisfaction: Definitions, Survey, Methodology, and Research Directions, L. Kallel, B. Naudts, A. Rogers (eds.), Theoretical Aspects of Evolutionary Computing, *Natural Computing Series*, Springer, 2001, pp.13÷58.

[10] E. Alba, J. Troya, A Survey of Parallel Distributed Genetic Algorithms, *Complexity*, Vol.4, №4, John Wiley & Sons, 1999, pp.1÷22.

[11] A. Chipperfield, P. Fleming, Parallel Genetic Algorithms, *Parallel and Distributed Computing Handbook*, A. Zomaya (ed.), MacGraw-Hill, 1996, pp.1118÷1143.

[12] R. Shonkwiler, Parallel Genetic Algorithms, S. Forrest (ed.), *Proc. of the Fifth International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 1993, pp.199÷205.

[13] E. Cantu-Paz, *A Summary of Research on Parallel Genetic Algorithms*, IllGAL Report 95007, University of Illinois, July 1995.

[14] E. Cantu-Paz, *Designing Efficient Master-Slave Parallel Genetic Algorithms*, IllGAL Report 97004, University of Illinois, 1997.

[15] R. Tanese, Distributed Genetic Algorithms, J. Schaffer (ed.), *Proc. of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, 1989, pp.434÷439.

[16] E. Cantu-Paz, *Designing Scalable Multi-Population Parallel Genetic Algorithms*, IllGAL Report 98009, University of Illinois, 1998.

[17] P. Borovska, M. Lazarova, Migration Policies for Island Genetic Models on Multicomputer Platform, *Proc. of IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications*, Dortmund, Germany, 2007, pp.143÷148.

[18] E. Alba, J. Troya, Analyzing Synchronous and Asynchronous Parallel Distributed Genetic Algorithms, *Future Generation Computer Systems*, Vol.17, №4, 2000, pp.451÷465.

[19] L. Wang, A. Maciejewski, H. Siegel, V. Roychowdhury, B. Eldridge, A Study of Five Parallel Approaches to a Genetic Algorithm for the TSP, *Intelligent Automation and Soft Computing*, Vol.11, №4, 2005, pp.217÷234.

[20] E. Cantu-Paz, Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms, *Journal of Heuristics*, Vol.7, №4, 2001, pp.311÷334.

[21] K. Crawford, Solving the N-Queens problem Using Genetic Algorithms, *Proc. of ACM/SIGAPP Symposium on Applied Computing*, Kansas City, USA, United States, 1992, pp.1039÷1047.

[22] A. Homaifar, J. Turner, S. Ali, The N-Queens Problem and Genetic Algorithms, *Proc. of the IEEE Southeast Conference*, Vol.1, 1992, pp.262÷267.

[23] A. Kilic, M. Kaya, A New Local Search Algorithm Based on Genetic Algorithms for the N-Queens Problem, *Proc. of the 2001 Genetic and Evolutionary Computation Conference* (GECCO 2001), USA, 2001.

[24] I. Martinjak, M. Golub, Comparison of Heuristic Algorithms for the N-Queen Problem, *Proc. of the 29th International Conference on Information Technology Interfaces* (ITI 2007), 2007, pp.759÷764.

[25] M. Božikovic, M. Golub, L. Budin, Solving N-Queen Problem Using Global Parallel Genetic Algorithm, *Proc. of International Conf. EUROCON*, Ljubljana, Slovenia, Vol.2, 2003, pp.104÷107.

[26] R. Haupt, S. Haupt, Practical Genetic Algorithms, Wiley-Interscience, 2004.

[27] M. Nowostawski, R. Poli, Parallel Genetic Algorithm Taxonomy, *Proc. of the Third International Conference on Knowledge-Based Intelligent Information Engineering Systems* (KES'99), IEEE Press, 1999, pp.88÷92.