BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **16**, No 1
Sofia • 2016 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.1515/cait-2016-0006

Strategy for Individuals Distribution by Incident Nodes Participation in Star Topology of Distributed Evolutionary Algorithms

Todor Balabanov, Iliyan Zankinski, Maria Barova

Institute of Information and Communication Technologies, BAS, 1113 Sofia, Bulgaria
Emails: todorb@iinf.bas.bg iliyan.zankinski@gmail.com maria_b88@abv.bg

Abstract: *One of the strongest advantages of Distributed Evolutionary Algorithms* (*DEAs*) *is that they can be implemented in distributed environment of heterogeneous computing nodes. Usually such computing nodes differ in hardware and operating systems. Distributed systems are limited by network latency. Some Evolutionary Algorithms* (*EAs*) *are quite suitable for distributed computing implementation, because of their high level of parallelism and relatively less intensive network communication demands. One of the most widely used topologies for distributed computing is the star topology. In a star topology there is a central node with global EA population and many remote computation nodes which are working on a local population* (*usually sub-population of the global population*). *This model of distributed computing is also known as island model. What is common for DEAs is an operation called migration that transfers some individuals between local populations. In this paper, the term 'distribution' will be used instead of the term 'migration', because it is more accurate for the model proposed. This research proposes a strategy for distribution of EAs individuals in star topology based on incident node participation* (*INP*). *Solving the Rubik's cube by a Genetic Algorithm* (*GA*) *will be used as a benchmark. It is a combinatorial problem and experiments are done with a C++ program which uses OpenMPI.*

Keywords: *Distributed evolutionary algorithms, migration strategy, incident nodes participation.*

1. Introduction

The efficiency of Evolutionary Algorithms (EAs), as search methods, comes from the ideas of natural selection and recombination. EAs are applied successfully in finding acceptable solutions to problems in business, engineering, and science

[1, 2]. For complex problems EAs are used for finding good solutions in a reasonable amount of time. Most of the problems can be solved on a single machine, but when the problem is bigger or computationally harder, finding a solution on a single machine becomes inefficient. A promising speedup of EAs comes from the increasing popularity of distributed computing systems. There are two main ways for representing Distributed Evolutionary Algorithms (DEAs) population. In the first case there is only one global population and in the second case the population is divided between different computing nodes. Strict classification of distributed implementations can be found in literature [3, 4, 5]. In the case of a distributed implementation of EAs the strategy for distributing individuals is critical (how individuals migrate in the system).

When the global population is divided between many remote computing nodes, each node has a fraction of individuals called subpopulation. In this kind of organization recombination and fitness function evaluation are done locally. Convergence process can be faster if there is an exchange of solutions between the remote nodes. This process of solutions exchange is called migration or in the case of INP individuals distribution. One of the algorithms, which was proposed recently (Free Search [11]), uses the idea that search in the global space is locally restarted on a fixed interval. INP idea builds on this proposal.

In this study a distribution strategy, based on incident node participation (remote computing nodes are expected to join the system and contribute computing power) is proposed. Computing nodes are organized as star topology and island model [6, 7] for DEAs is applied. As benchmark GA based solver of Rubik's cube is used. The implementation is in C++ with OpenMPI and can be found in public source code repository [8].

The rest of this paper is organized as follows: Section 2 describes individuals distribution strategy in details. Section 3 presents the proposed model put into practical application. Section 4 is devoted to experiments and results. Finally, Section 5 concludes and presents some ideas for further research.

2. Problem description

“Genetic algorithms are a robust adaptive optimization method based on biological principles. A population of strings representing possible problem solutions is maintained. Search proceeds by recombining strings in the population. The theoretical foundations of genetic algorithms are based on the notion that selective reproduction and recombination of binary strings changes the sampling rate of hyperplanes in the search space so as to reflect the average fitness of strings that reside in any particular hyperplane. Thus, genetic algorithms need not search along the contours of the function being optimized and tend not to become trapped in local minima.” [16] The place of GAs is better presented in Fig. 1.

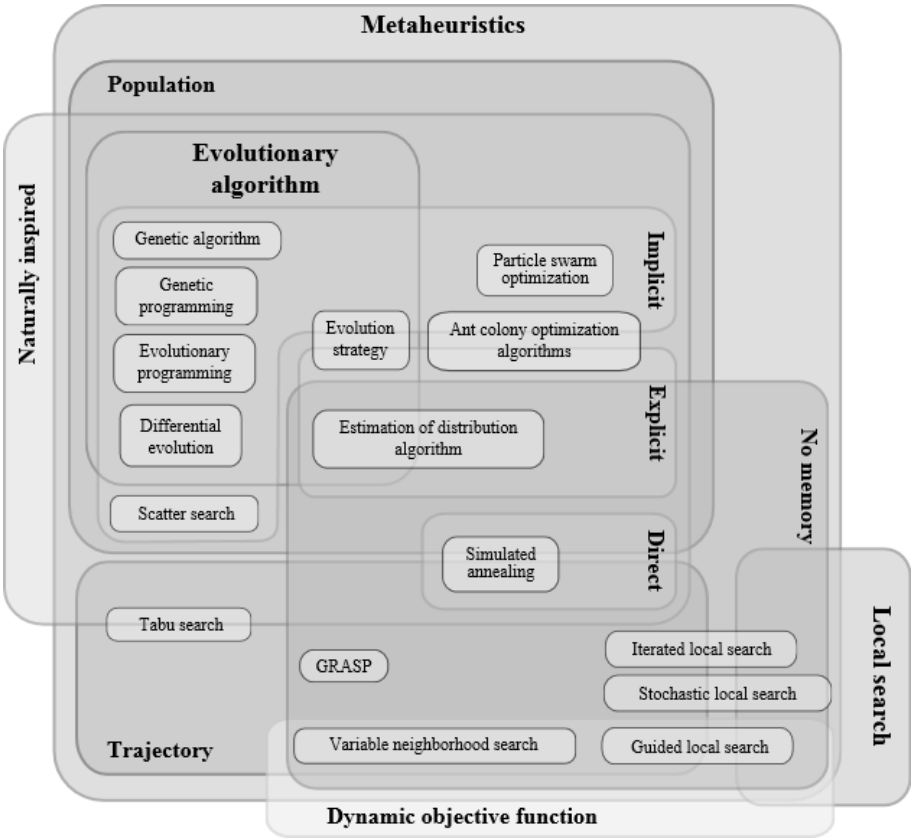

Fig. 1. Metaherustics diagram
(http://upload.wikimedia.org/wikipedia/commons/c/c3/Metaheuristics_classification.svg)

In terms of metaheuristics, GA is inspired by the nature, it is a population based and implicit method. The relation between GAs and Genetic Programming (GP) is pretty close and it is used in this study as representation of Rubik's Cube manipulation operations set as GAs chromosomes. When GAs are applied in distributed computing environment, individuals exchange between distributed computing nodes is as important as the other GA operators.

Because of the time consuming nature of EA-based combinatorial problems solving, the computing nodes in the project are relatively autonomous. The distributed system is organized as star topology with a central node and remote computing nodes. As DEA implementation, island model is used. There is a global EA population, located in the central node, and many local EA populations distributed on the remote computing nodes. Each remote computing node can join the system and leave the system at any moment in time, asynchronously (Fig. 2). For each joining client the central node sends a subset of the global population. The remote node evolves the local EA population as sequential EA.

The distribution of the individuals takes place only during the remote node joining process. Locally, best found solutions are reported to the central node and

eventually they migrate to the next remote node that joins. EA-based combinatorial problems solving is relatively slow so such distribution strategy is very efficient.

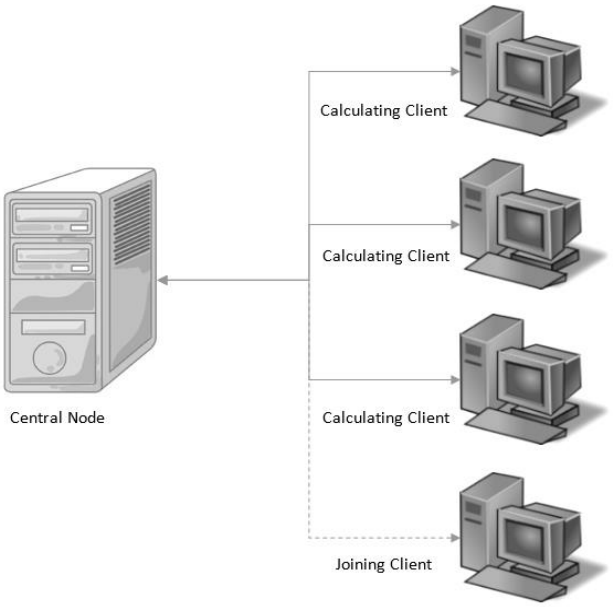

Fig. 2. Incident node participation individuals' distribution strategy

3. INP – practical application

As a benchmark problem a Rubik's cube solver is presented [8]. The cube was introduced by Erno Rubik in the late 70s of the 20th century. This is one of the most famous combinatorial puzzle games. The original version of the game consists of 3×3×3 cube (Fig. 3).

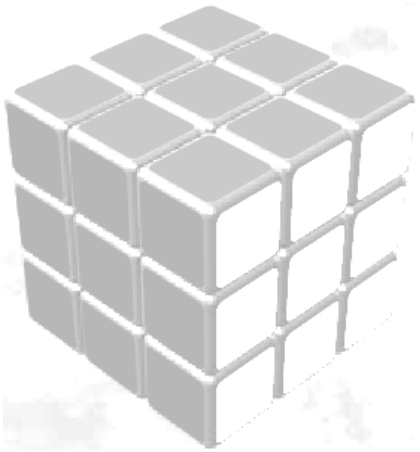

Fig. 3. Rubik's cube

There are different colored stickers on each of the exposed squares of the subcubes (cubies). Any 3×3×1 plane of the cube can be rotated or twisted in 90, 180, or 270 degrees, relative to the rest of the cube. In the initial state, all the squares on each side of the cube are in the same color. The puzzle is scrambled by making a number of random twists, and the task is to restore the cube to its original state. The problem is quite difficult. The Saganesque slogan printed on the package, saying that there are billions of combinations, is a considerable understatement. In fact, there are 4.3252×10^{19} different states that can be reached from any given configuration. By comparison, the 4×4 Fifteen Puzzle contains 10^{13} states, and the 5×5 Twenty-Four Puzzle generates 10^{25} states [12].

For a successful solution of the puzzle a sequence of moves should be generated, such that all cubies go to a side with their color. According to [12] it is estimated that such sequences use between 50 and 100 moves, when the puzzle is randomly scrambled. In this research the problem of finding optimal or sub-optimal solution by application of DEAs is addressed. In the experimental part two strategies for exchanging individuals are examined. In the first case ring topology is used and each traveler individual is sent to the neighbor node. In the second case star topology is used and INP strategy is used for exchange of individuals.

First step in the proposed model is the digital representation of the cube. In this research the cube was presented as 6 two-dimensional arrays (detailed description can be found in [8]). Each 3×3 array holds integer numbers matched to the cube colors. This is not the optimal cube representation as described in [12], but it is more convenient form a programming point of view.

The second step is related to the definition of basic operators. In this research six basic operators are used. They come from the theory of the minimal formal grammars. Each cube side rotation on 90 degrees clockwise is marked with letters as follows (a similar representation is used in [13]):

T (Top) – 90 degrees clockwise rotation of the top side;

L (Left) – 90 degrees clockwise rotation of the left side;

B (Back) – 90 degrees clockwise rotation of the back side;

R (Right) – 90 degrees clockwise rotation of the right side;

F (Front) – 90 degrees clockwise rotation of the front side;

D (Down) – 90 degrees clockwise rotation of the down side.

This set of six operations is the minimal fully functional grammar for the Rubik's Cube. Extended grammars are also possible, for example if counter-clockwise operators are included (+T, +L, +B, +R, +F, +D, –T, –L, –B, –R, –F, –D). Next level of extension is addition as number of turns (+1T, +2T, +3T, +1L, +2L, +3L, +1B, +2B, +3B, +1R, +2R, +3R, +1F, +2F, +3F, +1D, +2D, +3D, –1T, –2T, –3T, –1L, –2L, –3L, –1B, –2B, –3B, –1R, –2R, –3R, –1F, –2F, –3F, –1D, –2D, –3D).

The representation of the Rubik's Cube operators leads to a natural chromosome encoding in GAs. Each chromosome consists of six letters (presented above) as strings of different length. Each chromosome is a set of instructions that should be applied over the scrambled cube (more details can be found in [14]). In this research a single cut point crossover is applied, but other approaches are also possible as it is described in [15]. The simplest possible mutation is applied by

replacing a single instruction letter in the chromosome sequence. Parent selection is done at random and elitism rule is applied. Fitness value of individuals is calculated by Euclidean distance between a scrambled cube and a solved cube. Calculation of Euclidean distance is possible, because each cube's color is matched to an integer value. GA's optimization target is the minimum value for the Euclidean distance.

4. Experiments and results

All experiments were done on an 8 core server machine – Intel(R) Xeon(R) CPU E5504 @ 2.00 GHz, 64 GB RAM and Linux Version 3.13.0-24-generic, gcc version 4.8.2 (Ubuntu 4.8.2-19ubuntu1). For the parallel implementation OpenMPI is used.

"Open MPI represents more than a simple merger of LAM/MPI, LA-MPI and FT-MPI. Although influenced by previous code bases, Open MPI is an all-new implementation of the Message Passing Interface. Focusing on production quality performance, the software implements the full MPI-1.2 IQ and MPI-2 specifications and fully supports concurrent, multi-threaded applications (i.e., MPI_THREAD_MULTIPLE). To efficiently support a wide range of parallel machines, high performance drivers for all established interconnects are currently being developed." [18]

GA parameters used are as listed in Table 1 and *Constants.h* source file available at [8]. Parameters are selected according suggested values in [19].

Table 1. GA parameters

Para-meter	Local population size	Crossover rate	Mutation rate	Generation gap	Selection strategy	Migration interval
INP	50	0.95	1.00	100%	Elitist	67
Ring	50	0.95	1.00	100%	Elitist	67

Two individuals' distribution strategies are tested and compared in three different program runs (Figs 4-6). For each test execution 67 migrations were performed. Between each migration 1 M local GA optimization cycles were performed. The volume of exchanged data is the size of the local population, where the length of the chromosomes can vary. In all experiments there are 67 times of data exchange between the computing nodes (for more details *RubiksCubeGA.cpp* file at [8]). Data exchange is controlled by the node with identifier 0 and all other seven nodes communicate with it. There is no direct communication between nodes with identifiers from 1 up to 7.

In the three independent program runs it is obvious that the INP distribution strategy slightly outperforms the ring topology migration. On the y axis Euclidean distance between optimized cube and solve cube is measured. The main disadvantage, which is observed in ring topology experiments, is that sub-optimal solutions are distributed across the computing nodes and in this way escape from the local minima is harder. Better convergence of INP is based on the fact that solutions exchange is better than in the ring topology next node migration.

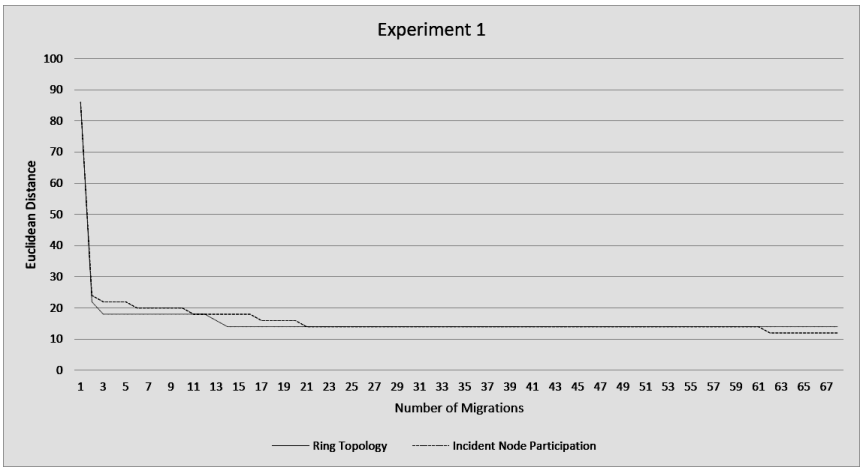

Fig. 4. Experiment 1

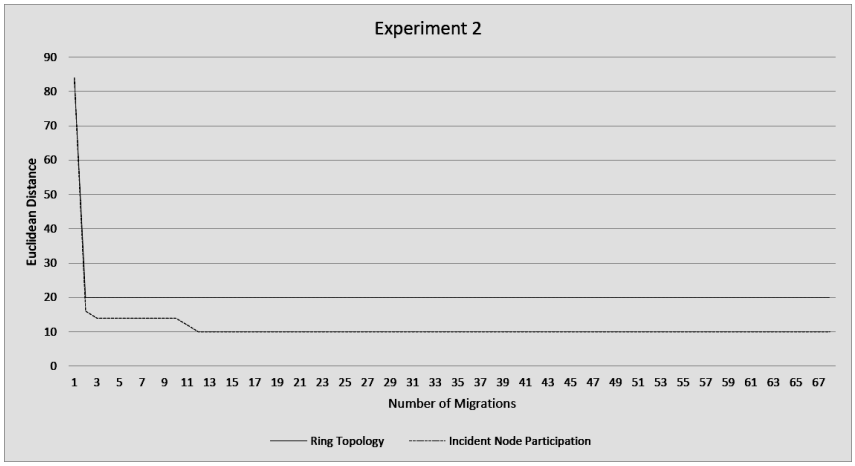

Fig. 5. Experiment 2

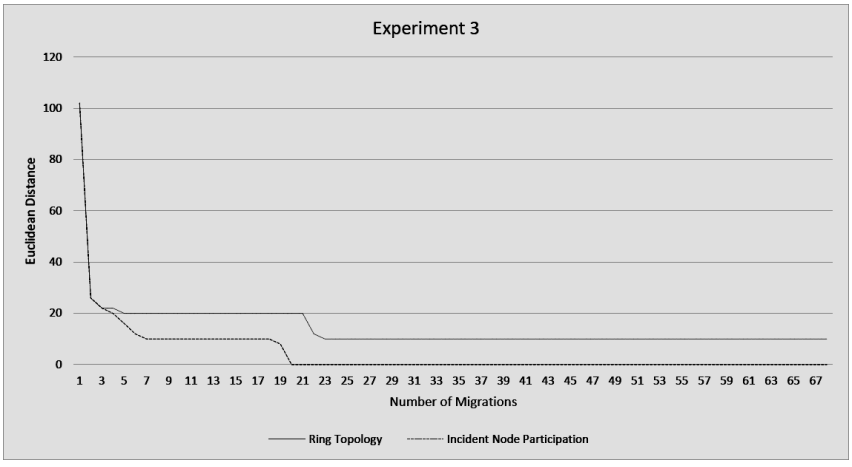

Fig. 6. Experiment 3

5. Conclusions

As it was shown in the experiments section, the presented distribution strategy, based on incident node participation outperforms ring topology distribution of individuals in distributed computing environment. The benchmark problem used is combinatorial and it is complex enough to represent the difference in convergence between the individuals distribution strategies used. The evolutionary approach seems a promising way of solving complex combinatorial problems. The best advantage of EAs comes from their perfect ability to be parallelized in distributed computing environment.

In further studies it would be interesting to observe if FPGA devices are tested for better acceleration of the computation process (as it is described in [9]) and hybrid evolutionary algorithms (presented in [17]). Another interesting direction for further research can be made in the direction of Generalized Nets, which are presented for solving similar problems [10].

Acknowledgements: This work was supported by a grant of the Bulgarian National Scientific Fund under the grants DFNI 02/20 Efficient Parallel Algorithms for Large Scale Computational Problems and DFNI 02/5 InterCriteria Analysis A New Approach to Decision Making.

References

1. Goldberg, D. E. Genetic and Evolutionary Algorithms Come of Age. – Communications of the ACM, Vol. **37**, 1994, No 3, pp. 113-119.
2. Pappa, G., G. Ochoa, M. Hyde, A. Freitas, J. Woodward, J. Swan. Contrasting Meta-Learning and Hyper-Heuristic Research: The Role of Evolutionary Algorithms. – Genetic Programming and Evolvable Machines, Vol. **15**, 2014, Issue 1, pp. 3-35.
3. Adamidis, P. Review of Parallel Genetic Algorithms Bibliography. Tech. Rep. Version 1. Aristotle University of Thessaloniki, Thessaloniki, Greece, 1994.
4. Gordon, V. S., D. Whitley. Serial and Parallel Genetic Algorithms as Function Optimizers. – In: S. Forrest, Ed. Proc. of 5th International Conference on Genetic Algorithms, Morgan Kaufmann (San Mateo, CA), 1993, pp. 177-183.
5. Lin, S.-C., W. Punch, E. Goodman. Coarse-Grain Parallel Genetic Algorithms – Categorization and New Approach. – In: Proc. of 6th IEEE Symposium on Parallel and Distributed Processing, IEEE Computer Society Press, 1994, Los Alamitos, CA.
6. Tanese, R. Distributed Genetic Algorithms. – In: J. D. Schaer, Ed., Proc. of 3rd International Conference on Genetic Algorithms, Morgan Kaufmann, 1989, pp. 434-439.
7. Uchida, T., T. Matsuzawa, Y. Inoguchi. The Inuence of Elitism Strategy on Migration Intervals of a Distributed Genetic Algorithm. – Proceedings in Adaptation, Learning and Optimization, Vol. **2**, 2015, pp. 363-374.
8. Balabanov, T. MPI Parallel Implementation of Genetic Algorithm Based Rubik's Cube Solver. **https://github.com/TodorBalabanov/RubiksCubeGeneticAlgorithmsSolver**
9. Ivanov, V. Using a PicoBlaze Processor to Traffic Light Control. – Cybernetics and Information Technologies, 2015, pp. 131-139.
10. Tashev, T., V. Monov. Modeling of the Hotspot Load Traffic for Crossbar Switch Node by Means of Generalized Nets. – In: Proc. of 6th IEEE International Conference Intelligent Systems, 2012, Sofia, Bulgaria, pp.187-191.
11. Penev, K. Free Search in Multidimensional Space II. – Numerical Methods and Applications Lecture Notes in Computer Science, Vol. **8962**, 2015, pp. 103-111.

12. K o r f, R. Finding Optimal Solutions to Rubik's Cube Using Pattern Databases. – In: Proc. AAAI-98, Madison, WI, AAAI Press, Menlo Park, CA, 1998, pp. 700-705.
13. R a n d a l l, K. Cilk: Efficient Multithreaded Computing. Doctor of Philosophy in Computer Science and Engineering, Massachusetts Institute of Technology, 1998.
14. A n t o n i s s e, H. A Grammar-Based Genetic Algorithm. Foundations of Genetic Algorithms, Indiana University, Bloomington, USA, 1991, pp. 193-204.
15. P o l i, R., J. K o z a. Genetic Programming. – Search Methodologies, 2014, pp. 143-185.
16. W h i t l e y, D., T. S t a r k w e a t h e r, C. B o g a r t. Genetic Algorithms and Neural Networks: Optimizing Connections and Connectivity. – Parallel Computing, Vol. **3**, 1990, Issue 3, pp. 347-361.
17. G u l i a s h k i, V., L. K i r i l o v. Hybrid Evolutionary Algorithm for Multiple Objective Convex Integer Problems. – In: Proc. of 28th International Conference on Information Technologies (InfoTech-2014), Varna, St. St. Constantine and Elena Resort, Bulgaria, 2014, pp. 19-28.
18. G a b r i e l, E., G. F a g g, G. B o s i l c a, T. A n g s k u n, J. D o n g a r r a, J. S q u y r e s, V. S a h a y, P. K a m b a d u r, B. B a r r e t t, A. L u m s d a i n e, R. C a s t a i n, D. D a n i e l, R. G r a h a m, T. W o o d a l l. Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation, Recent Advances in Parallel Virtual Machine and Message Passing Interface. – In: Lecture Notes in Computer Science, Vol. **3241**, 2004, pp. 97-104.
19. B o n i s s o n e, P. Genetic Algorithms Parameter Settings, GE Corporate Research & Development.
http://homepages.rpi.edu/~bonisp/fuzzy-course/99/L10/ga+flc_app_4printer.pdf