Parallelization Clonal Selection Algorithm with MPI.NET for Optimization Problem

Ayi Purbasari, Achmad Nizar Hidayanto, Arief Zulianto

Abstract: ***The idea of immune system as a computing inspiration has given rise to the Artificial Immune System (AIS). AIS has contributed in the field of optimization of complex issues, one of which is a clonal selection algorithm (CSA) for solving the Optimization Problem such as Traveling Salesperson Problems (TSP). Parallel characteristic inherently possessed by the immune system, provided an opportunity to give parallel computing to achieve better computational performance. The study resulted in two parallel models for the clonal selection algorithm. First model is a master-slave model, there is a master process that controls all communication. In the second model, all processes equivalent and communicate with each other. Both models are prepared to be built in system development with parallel environment using Visual C# language with MPI.NET framework. For both datasets, experiment gave consistent results. Model 1 is superior in getting the best-tour's cost, although obtained with longer execution time, compared with Model 2. However, Model 2 is superior in less execution time needed.***

Keywords: Clonal Selection Algorithm; CSA; parallel Clonal Selection Algorithm; message passing model; MPI.NET; TSP.

I. INTRODUCTION

There are many inspirations from biological systems (bio-inspired) in computational problems solving. Research related to neural networks derived from observations of the brain [1]. Optimization algorithm based on ant colony behavior [2] or bees [3], cell system evolution underlying evolutionary algorithms, e.g., genetic algorithms [4]. The extracted behavior of biological systems then adopted and became an inspiration to handle computational problems in complex systems. One of bio-inspired system that observed and became the basis of computing is the immune system [5, 6] and is known as Artificial Immune Systems (AIS). In principle, AIS uses the metaphor of the vertebrate immune system to create new solutions for complex problems —or at least give new ways of looking at the problems. Several AIS contributions has been solving many problems in optimization, intrusion detection, and learning [6, 7]. Optimization problem is the problem of finding the best solution for all feasible solutions. One of the optimization problems fields that continuously investigated is a combinatorial optimization problem such as Traveling Salesperson Problems (TSP), Vehicle Routing Problem (VRP), and Winner Determination Problem (WDP). This combinatorial problem can be solved with problem solving procedure with deterministik or approximate approach.

Revised Manuscript Received on May 22, 2019.

Ayi Purbasari, Informatic Department, Universitas Pasundan, Bandung, Indonesia

Achmad Nizar Hidayanto, Computer Science Department, Jakarta, Indonesia

Arief Zulianto, Master of Informatic, Universitas Langlangbuana, Bandung, Indonesia

There are two approaches for approximate approach, single solution or population-based approached. Population-based approached concerns a population of solutions at a time and may take long time to evaluate the individuals. This population-based approached is the use of Genetic Algorithm and Ant Colony Optimization, and also Artificial Immune System with Clonal Selection Algorithms (CSA) [8],

Clonal selection algorithm first used to solve TSP in 2002 by Leandro and Von Zuben [9] with CLONALG, an algorithm that refers to the clonal selection theory. Some researchers conduct experiments and/or modifications to the CSA algorithm for machine learning and optimization problems [7], in particular TSP problem [10]. Watkins [11] expressed an idea that the immune system is inherently has parallel characteristics, in term of control, responses, maturation process and cell population management. Inspired algorithms parallelization becomes potency for algorithm development to solve more complex problems. As a heuristic optimisation algorithm, CSA promises to solve the TSP problem [2, 8, 12] and its parallel version used by [14]. CSA used to solve mTSP by [15] and parallel version of mTSP [16].

Various models of parallelism exist, such as message passing and shared memory. These models need to be reviewed to look for a match with the characteristics of the immune system, in particular the clonal selection. There are some models in parallel computation, including the model of message passing with the standard Message Passing Interface (MPI) [17]. This model uses Single Program Multiple Data (SPMD) [18][19] techniques. Some implementations of MPI have been used in [14] to solve TSP to solve mTSP. There is another implementation library name MPI.NET that used in NET environment [20]. This research will exploit parallel CSA algorithm to solve TSP with MPI.NET. The result will show how to use the MPI.NET and it's performance. Systematically, this paper contains an introduction, research methods, results and discussion, and conclusions.

II. RELATED RESEARCH

Clonal Selection Algorithm

The clonal selection theory is a theory used to describe the functioning of acquired immunity, specifically the diversity of antibodies used to defend the organism from invasion [3, 4, 6, 7]. This theory postulated by Burnet (1957), stated that antibodies production is the selection process triggered by antigens as their enemies.

The theory specifies that the organism have a pre-existing pool of heterogeneous antibodies that can recognize all antigens with some level of specificity [3, 4, 6, 7]. An antibody then chemically binds to the antigen if its receptors matched to an antigen. This chemically binding causes the cell to clone, replicate and produce more cells with the same receptor. During the cloning stage, cells genetic mutations occur and promote the match or affinity with the antigen. This allows the binding ability of the cells to improve with time and exposure to the antigen. Figure 1. below:

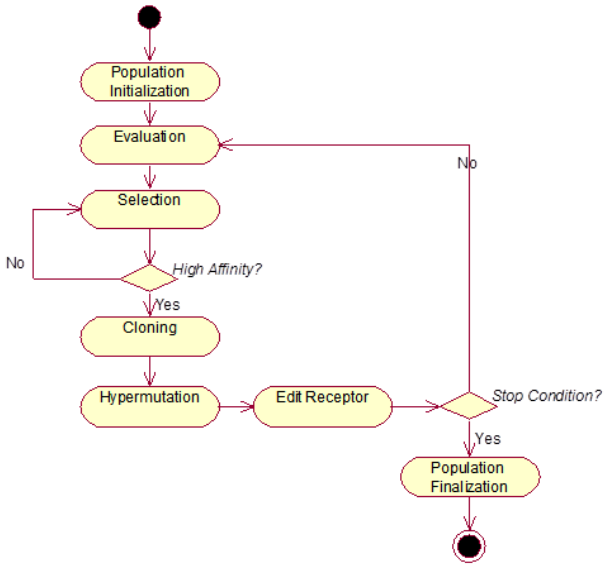

Fig. 1 Clonal Selection Algorithm

The clonal selection theory is inspiring deCastro and Von Zuben to create clonal selection algorithms [4][7] name ClonalG. This figure 1 shows the workflow of the Clonal Selection Algorithm technique. This algorithm has several parameters that are summarized in the following table:

Table. 1 Clonal Selection Algorithm Parameters

No.	Parameter	Descripton
1	N: antibody population size	Antibody maintained by system
2	M: memory pool size	Memory pool size for antibody
3	N: selection pool size	Amount of antibody with highest affinity that will be cloned
4	d: remainder replacement size	Amount of antibody with worst affinity that will be replaced
5	β: clonal factor	Constanta for cloning factor
6	G: number of generations	Number of generation, will be used as stopping criteria
7	the random number generator seed	Random number for population size

Clonal Selection Algorithms and Optimization Problem

When using algorithm for solving problem, we need mapping between problem and properties of algorithms. Here is the mapping between Optimization problems with the CSA. We use TSP problem as an example as follows:

1. Population initiated randomly of all possible solutions in form TSP tours and its cost (there are n!/2 Hamiltonian circuit, with n is vertices number). Total population defined as initial variable.
2. There is a function to select affinity which represented as minimum tour's cost. High affinity presented as low cost. Number of selections is defined as initial variable.
3. There is a function of cloning and hypermutation, to clone a number of selected population and then hypermutate the population. The number of copies defined as initial variable. Hypermutation of population copies done in accordance to affinity value. The higher value will give lower possibility to undergo mutation. Then cloned and mutated population added to the initial.
4. Population for further selection process. The population with the best cost will be chosen while the rest are ignored.
5. The steps above will be repeated until stopping criterion is satisfied. Stopping criterion can be: a maximum iteration number, selection results were converged, or all of the tour had been finished evaluated.

Parallel Clonal Selection Algorithm

Parallel models for CSA

Two parallel models for CSA are produced in this research. The first model is a fork-join model with the process 0 as the root (master) process. The second model is a mess-models, which all processes equivalent and communicate with each other. In this model, a main (root) process acts as the manager of all other parallel processes.

The main process is responsible for the data initiation and partitioning, and then distribution to other processes. The execution results of each process are then sent back to the main process. This scheme will be repeated if necessary.

With this model, we have a computational model for parallel algorithm is shown in Figure 2 and Figure 3 below.

In parallel model 2, the process 0 is responsible for initiating the data and distributes it to the entire process, including for itself. Execution results of each process then broadcasted to all other processes and aggregated in each process. This scheme will be repeated if necessary.

Parallel models for CSA

Parallel CSA implemented using Visual C# 2010 Express 01014-169-2560017-70895 with MPI.NET Library on Microsoft® Visual Studio 2010 10.0.30319.1 RTMRel, and Microsoft® .NET Framework ver. 4.0.30319. Implementation is done on Toshiba Portégé Z935 notebook with Intel® Core™ i5 3317 1.7GHz CPU and 4GB RAM. This notebook runs on 64-bit edition of Microsoft® Windows™7 OS.

Application flow logic implementation is summarized as follows:

1. Take dataset; convert the dataset into a matrix of costs of the distance between the coordinates of each point.
2. Setup parameters: population size, selection size, cloning factor, mutation factor.
3. Setup random operations to create the initial population for some of the tour.
4. Setup operations evaluation to check the affinity, the cost of the tour in initial population.
5. A Selection operation to select a number of best populations, which is the tour with the minimum cost, based on selection parameter.
6. Cloning and hypermutation, tour duplication and mutation.
7. Replace the tour in population with the mutated tour.

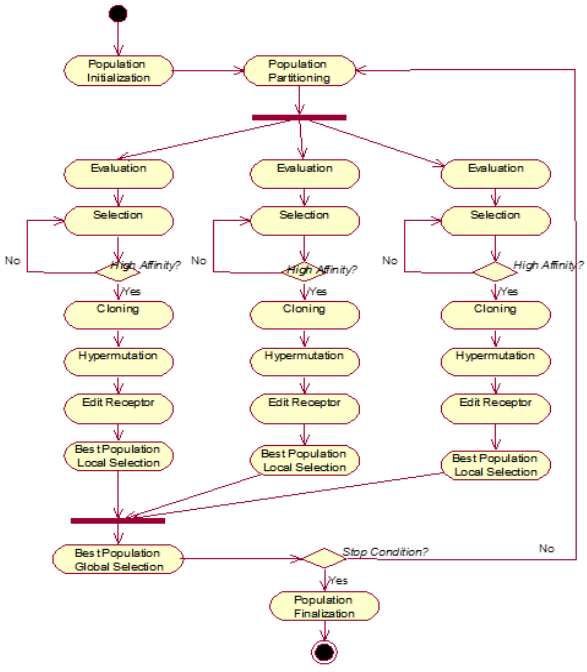

Fig. 2 Parallel Clonal Selection Algorithm Model 1

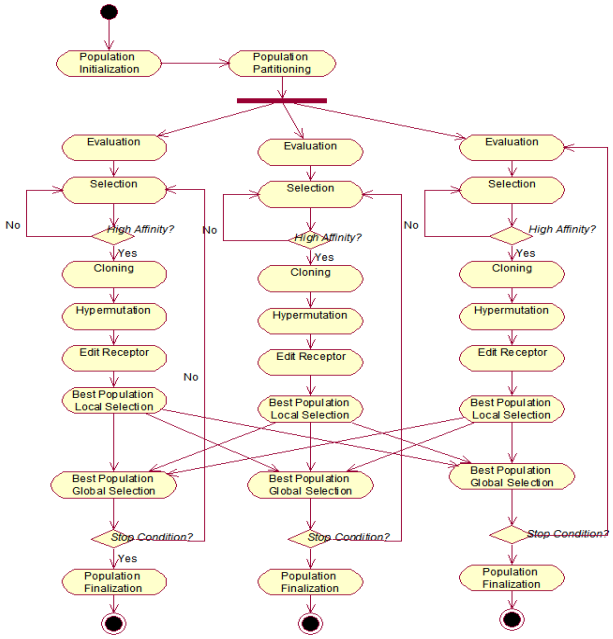

Fig. 3 Parallel Clonal Selection Algorithm Model 2

III. RESULTS AND DISCUSSION

Once the applications implemented, experiment conducted using datasets taken from TSPLib [16][21]. The experiment started using datasets with 52 vertices, i.e., Berlin52 dataset.

Experiment Design

CSA algorithm has 6 (six) parameters (Table 1). The main parameters observed are N (the number of population) and n (the size of selections). Additional parameter np (number of process) added for parallelized version of the algorithm. The algorithm execution affected by the number of generation (G). In this experiment, parameter G fixed to value of 1000. These three parameters N, n, and np will be conditioned on parallel computing models 1 and 2 and tested with the complexity issues represented by the number of nodes. The ultimate goal is to find execution time of each execution for speedup measurement. In addition, the tour will be observed to find the best value achieved with certain converging point for both models. In summary, the following Figure 5 is a design experiment for parallelization of the clonal selection algorithm. With:

1. N = initial population size (population Size),
2. n = number of selected antibodies for cloning (selection Size),
3. G = number of generations,
4. s = number of nodes as representations of the complexity of the problem,
5. p = number of parallel processes,
6. wes = sequential execution time,
7. wep = parallel execution time,
8. speedup = speed up achieved, i.e., wes/wep,
9. convergence = the number of generations where convergence achieved, equipped with the best tour's cost gained.

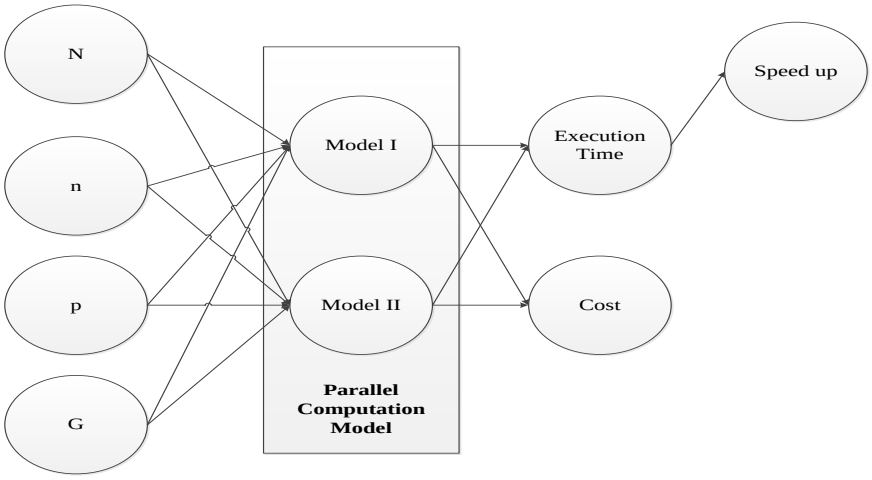

Fig. 4 Experiment design

Simulation execution arranged as follows:

Experiment 1: Preparation determination of the values of the parameters and the number of executions to be carried out:

1. Set G = 1000, the number of generations, as a condition of final execution.
2. Set N = {50, 100, 150, 200), n = {5, 10, 15, 20}, and p = {2, 4, 8, 12, 16}.
3. Dataset will be used in observation (Berlin52.tsp).
4. Two models of parallel computation (model 1 and model 2).
5. Reduction is done by means of a fixed N value with varying n value, followed by a fixed n value with varying N value.

Experiment 2: Execution of Parameter Observations: In this research, carried out the reduction of execution with the following scenarios:

1. Execution of parallel models 1 and 2 for 2 processors with 52 nodes dataset for fixed values of n and varying N.
2. Execution of parallel models 1 and 2 for 2 processors with 52 nodes dataset for fixed values of N and varying n.

Experiment 3: Execution of Performance Observations:

1. Execution for 4, 8, 16, processors with datasets of 52, with a fixed value of G, N, and n (value N and n obtained from the previous executions).
2. Execution of sequential models with datasets of 52, with a fixed value of G, N, and n.
3. Calculate the speedup, i.e., the ratio between execution time of sequential and parallel model.

Experiment Results

In this section, we will see the experiment result for Berlin52.tsp dataset. There are two results from experiments:

1. With selection size (n) = 10 and varying population size (N)
2. With selection size (n) is varied and population size = 50

The initial dataset consist of set of cities in Berlin as follows in Figure 5, and optimal tour obtained, from TSPLibs with its best cost 7542 (Figure 6)

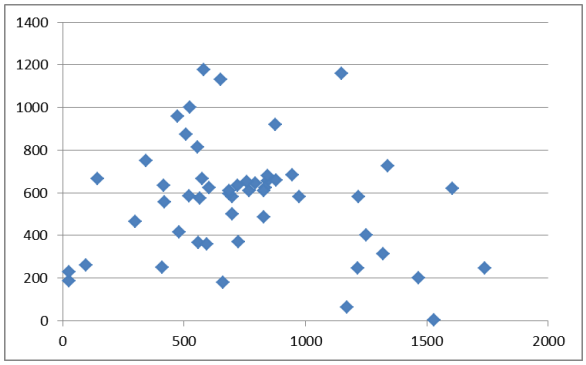

Fig. 5 Initial Tour of Berlin52 Dataset

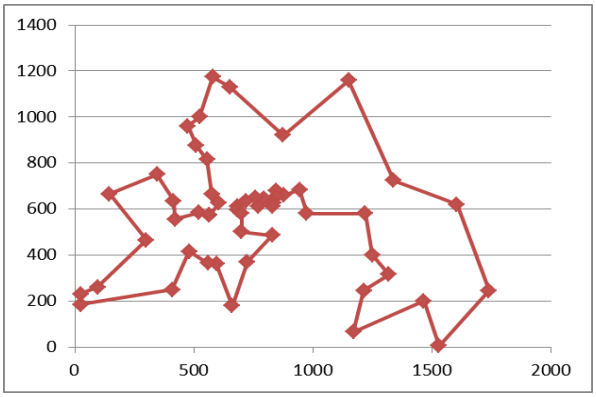

Fig. 6 Optimal Tour of Berlin52 dataset (from TSPLib)

Result 1: Parameter observations

Using 2 processors and varying n and N values, this Table 2 below shows the result. Best cost is 13006.27 (57.98% of optimum cost 7542). Its obtained using model 1 with N = 100 and n = 20. This result achieved in 2:38.3 minutes. Fastest execution time is 00:29.526 minutes. This result achieved for model 2 with N = 50 and n = 10. Cost obtained for this execution is 21542.37 (35.010% of optimum cost). Model 1 is superior in getting the best tour's cost, although obtained with longer execution time, compared with Model 2. However, Model 2 is superior in less execution time needed.

Table. 2 Clonal Selection Algorithm Parameters

Exp	Model	N	n	Best Tour		
				Cost	Generation	Execution Time
1	1	50	10	17841.85	16	00:46.595
2	1	100	10	18339.07	12	01:32.478
3	1	150	10	14149.32	39	02:19.637
4	1	200	10	14502.24	22	03:11.223
5	2	50	10	21542.37	302	00:29.526
6	2	100	10	21354.46	875	00:58.256
7	2	150	10	21127.29	624	01:32.607
8	2	200	10	21256.62	160	02:08.632
9	1	100	5	16860.56	16	01:07.820
10	1	100	10	18339.07	12	01:32.478
11	1	100	15	15006.29	24	02:03.038
12	1	100	20	13006.27	41	02:38.320
13	2	100	5	21925.75	540	00:53.636
14	2	100	10	21354.46	875	00:58.256
15	2	100	15	21756.79	508	01:14.785
16	2	100	20	21443.92	579	01:12.882

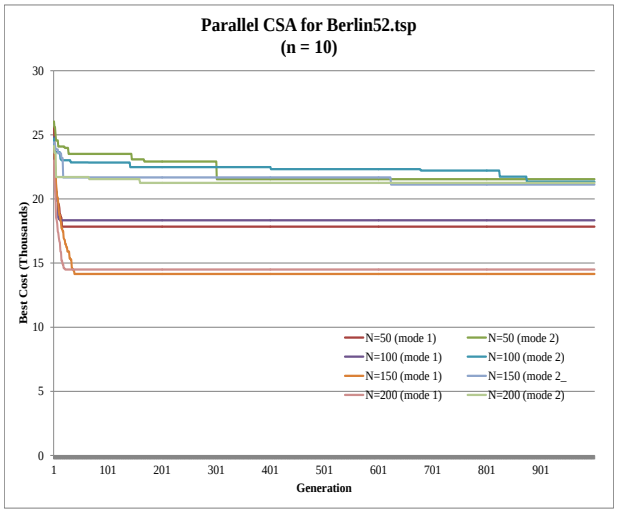

Fig. 7 CSA Parallelization results with varying N

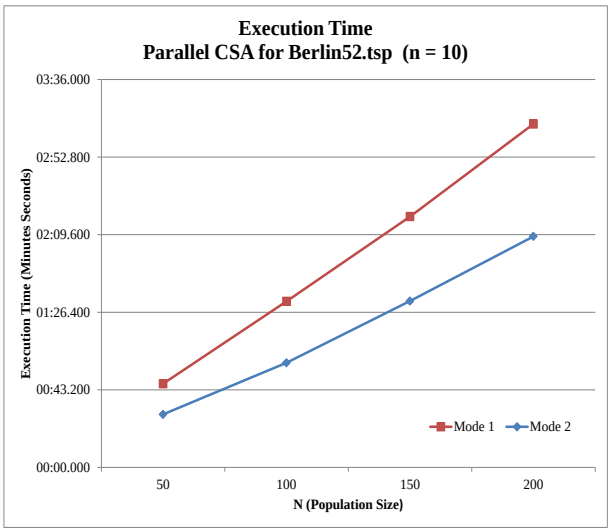

Fig. 8 Parallel CSA execution time with varying N

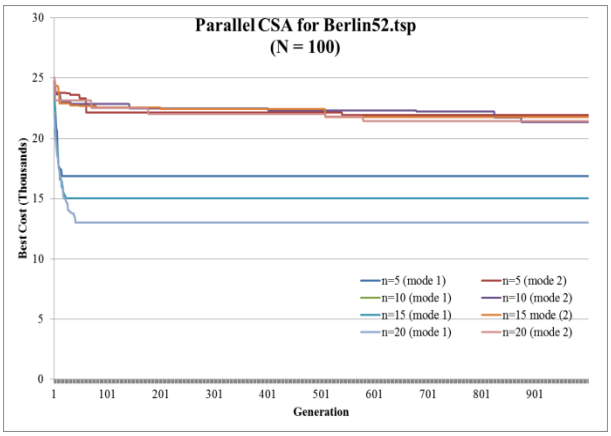

Fig. 9 CSA Parallelization results with Varying n

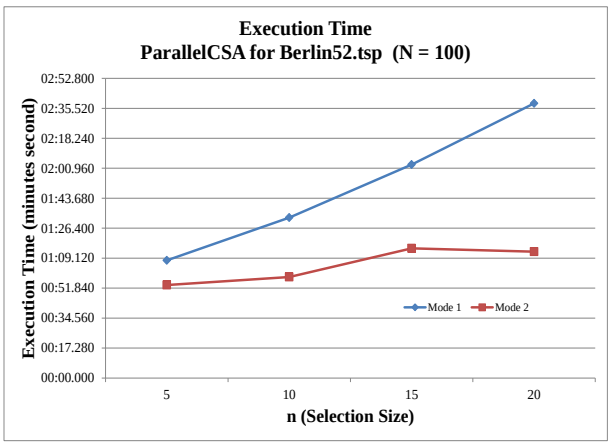

Fig. 10 CSA Parallelization results with varying N

Best tour obtained by:
12,51,26,50,11,32,10,27,25,46,15,35,47,0,22,1,6,28,23,4,42,8,9,40,44,7,18,33,36,37,34,43,39,38,19,49,14,5,45,21,16,2,31,48,17,30,41,29,20,24,3,13. (Figure 11)

Best tour obtained by:
10, 51, 13, 46, 27, 24, 5, 17, 2, 16, 29, 41, 1, 6, 20, 30, 22, 33, 4, 23, 14, 38, 47, 11, 50, 26, 3, 45, 15, 43, 21, 0, 48, 34, 35, 32, 8, 9, 44, 18, 36, 37, 42, 39, 31, 7, 40, 49, 19, 28, 25, 12 (Figure 12)

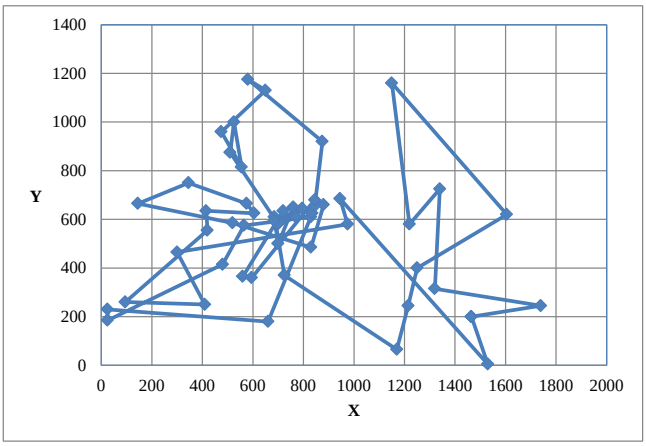

Fig. 11 Optimal Tour Obtained from Model I

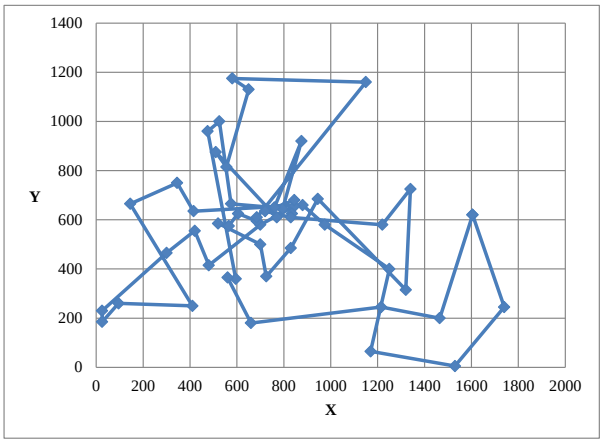

Fig. 12 Optimal Tour Obtained from Model II

Result 2: Performance observations

This section will explain the results of experiment observations of parallel computing performance for both datasets. Table 3 summarizes the speedup obtained for both parallel models for 2, 4, 8, 12, and 16 numbers of processes:

Table. 3 Speedup achieved for Berlin52.tsp

Model	Number of process	Execution Time		Speedup
		Parallel	Sequential	
1	2	00:15.222	00:05.000	0.328
1	4	01:20.460	00:05.000	0.062
1	8	05:49.277	00:05.000	0.014
1	12	11:43.518	00:05.000	0.007
1	16	22:07.369	00:05.000	0.004
2	2	00:26.073	00:05.000	0.192
2	4	01:05.440	00:05.000	0.076
2	8	01:53.942	00:05.000	0.044
2	12	01:36.411	00:05.000	0.052
2	16	01:49.936	00:05.000	0.045

Speedup obtained were less than one, which indicates that there is no significant improvement in computational performance. This is because computation done on single processor only. Best speedup achieved for two processes (p = 2). First model has better speedup compared to the second model. However, speedup significantly decreased as the number of process increased. With the optimal cost from TSPLib, observation result for 2, 4, 8, 12, and 16 number of process summarized in Table 4.

Table. 4 Best Costs comparison for Berlin52.tsp

Model	Number of process	Best Tour		% Optimal	
		Parallel	Sequensial	Parallel	Sequential
1	2	16503.03	15836.84	45.70%	47.62%
1	4	16772.5	15836.84	44.97%	47.62%
1	8	15630.46	15836.84	48.25%	47.62%
1	12	15550.46	15836.84	48.50%	47.62%
1	16	16427.38	15836.84	45.91%	47.62%
2	2	21980.04	15836.84	34.31%	47.62%
2	4	21655	15836.84	34.83%	47.62%
2	8	22439	15836.84	33.61%	47.62%
2	12	21638.3	15836.84	34.85%	47.62%
2	16	20754.7	15836.84	36.34%	47.62%

IV. CONCLUSIONS

Two parallel computation models for CSA have been produced in this research. Both models are prepared to be implemented in parallel environment system using Visual C# with MPI.NET library. Experiment design in the form of execution scenarios and TSP datasets, also equipped in this research. For both datasets, experiment results are consistent wherein:

1. Model 1 gave best cost, but need longer execution time compared to model 2.
2. Model 2 gave shorter execution speed
3. The best value of N for both models is 50 or 100
4. The best value of n for both models is 20 or 10

Best cost is 13006.27 (57.98% of optimum cost). Its obtained using model 1 with N = 100 and n = 20. This result achieved in 2:38.3 minutes. Fastest execution time is 00:29.526 minutes. This result achieved for model 2 with N = 50 and n = 10. Cost obtained for this execution is 21542.37 (35.010% of optimum cost).

Model 1 gave the best cost, with the price of longer execution time, compared to model 2. However, model 2 gave shortest execution time, with tour's cost worse than model 1.

Research continued to observe parallelism performance and scalability measurement of computational models, by applying the parallel computation model for the larger or more complex TSP problem sizes. In addition, experiment should be implemented in an environment for parallel computing development, e.g., a computer cluster.

ACKNOWLEDGEMENTS

This work was partially supported by the Universitas Pasundan and Ministry of Research Technology and Higher Education Republic of Indonesia as preliminary research for post-doctoral grant year 2018.

REFERENCES

1. R.P Lippmann, "An introduction to computing with neural nets," ASSP Magazine, IEEE, pp. 4 - 22, April 1987.
2. Marco Dorigo, "Ant colony optimization: a new meta-heuristic," in Congress on Evolutionary Computation, vol. 2, Washington, DC, 1999.
3. D Karabogak and Bastur, B, "On the performance of artificial bee colony (ABC) algorithm," Elsevier: Applied Soft Computing, vol. 8, no. 1, pp. 687–697, January 2008.
4. David E. Goldberg and John H. Holland, "Genetic Algorithms and Machine Learning," Machine Learning 3, pp. 95-99, 1988.
5. Jonathan Timmis and Leandro Nunes Castro, Approach, "Artificial Immune Systems: A New Computational," London: Springer Verlag, 2002.
6. Leandro N. de Castro and Fernando J. Von Zuben, "Artificial Immune Systems: Part I – Basic Theory and Applications," Departement of Computer Engineering and Industrial Automation, School of Electrical and Engineering, State University of Campinas Brazil, TR - DCA 01/99., 1999:1.
7. Leandro N. de Castro and Fernando J. Von Zuben, "Artificial Immune Systems: Part II - A Survey of Applications," Departement of Computer Engineering and Industrial Automation, School of Electrical and Engineering, State University of Campinas Brazil, 2000.
8. Donald Davendra, "Traveling Salesperson Problem: Theory and Application", Donald Davendra, Ed. Croatia: InTechWeb.Org, 2010.
9. Leandro N. de Castro and Fernando J. Von Zuben, "Learning and Optimization Using the Clonal Selection Principle," IEEE Transactions On Evolutionary Computation, vol. 6, no. 3, pp. 239-251, June 2002.
10. Gaber J Bakhouya M, "An Immune Inspired-based Optimization Algorithm : Application to the Traveling Salesman Problem," AMO - Advanced Modeling and Optimization, vol. 9, no. 1, pp. 105-116., 2007.
11. Andrew B. Watkins, "Exploiting Immunological Metaphors In The Development Of Serial, Parallel, And Distributed Learning Algorithms," Cantenbury - United Kingdom, Master Thesis 2005.
12. Jacek Dabrowski and Marek Kubale, "Computer Experiments with a Parallel Clonal Selection Algorithm for the Graph," in IEEE International Symposium on Parallel and Distributed Processing, Miami, FL, 2008, pp. 1-6.
13. Ayi Purbasari, Oerip S. Santoso, Rilla Mandala, Iping S. Supriana, "Data Partition and Communication on Parallel Heuristik Model Based on Clonal Selection Algorithm", TELKOMNIKA, Vol 13, No1, 2015, pp 193 – 201
14. Ayi Purbasari, "Clonal Selection Algorithm Parallelization with MPJExpress," 8th Computer Science & Electronic Engineering Conference (CEEC 2016) University of Essex, Essex, United Kingdom September 28-30th 2016
15. Ayi Purbasari, Oerip S. Santoso, Rila Mandala, Iping S. Supriana, "A New Approach to Solve Multiple Traveling Salesmen Problem by Clonal Selection Algorithm", International Journal of Applied Engineering Research (IJAER), Research India Publications, vol. 9, No. 21, pp.. 11005-11017
16. Ayi, Purbasari, "Using parallel clonal selection algorithm to solve multitravelling salesperson problem", in 6th International Workshop on Computer Science and Engineering, WCSE, Tokyo, 17-19 June 2016
17. Argonne National Lab Mathematics and Computer Science. The Message Passing Interface (MPI). [Online]. http://www.mcs.anl.gov/research/projects/mpi/
18. Blaise Barney. Introduction to Parallel Computing. [Online]. https://computing.llnl.gov/tutorials/parallel_comp/#Designing
19. Ian Foster. (1995) Designing and Building Parallel Programs. [Online]. http://www.mcs.anl.gov/~itf/dbpp/
20. Indiana University. MPI.NET: High-Performance C# Library for Message Passing. [Online]. http://www.osl.iu.edu/research/mpi.net/
21. TSPLIB. [Online]. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

BIBLIOGRAPHY OF AUTHORS

Dr. Ayi Purbasari received her Doctoral from the Insitut Teknologi Bandung (ITB), Bandung Indinesia. She is a lecturer at Informatics Departement, Universitas Pasundan since 2004 – present. Specializing in software engineering and also computation. She has published in Telkomnika journal and IJAER journal on Computational Intelligent. She has interest in parallel computation and nature-inspired algorithm. With Scopus ID 55001755600 with 8 documents and h-index 1

	Dr. Achmad Nizar Hidayanto is a lecturer at Faculty of Computer Science Universitas Indonesia, expertise in Computer Science, Business Management and Accounting, Decision Sciences, Social, Sciences, and Engineering. With Scopus ID 57205093001 with 135 documents and h-index 7.
	Dr. Arief Zulianto received her Doctoral from the Insitut Teknologi Bandung (ITB), Bandung Indinesia with dissertation High Performance Computing. He combined natural-based algorithm for high performance computing. He is a lecturer at Informatics Magister, Universitas Langlanguana since 2014 – present. Specializing in network and security / cybersecurity. With Scopus ID 55001755500 with 2 documents.