# Knowledge Sharing in Cooperative Compact Genetic Algorithm

Orakanya Gateratanakul, Prabhas Chongstitvatana
Department of Computer Engineering
Chulalongkorn University
Bangkok, Thailand
orakanyag12@gmail.com, prabhas.c@chula.ac.th

***Abstract*—In this study, we present three methods of sharing knowledge between cooperative compact genetic algorithms. The methods exploit the effect of the worse solutions of the two-cooperative compact genetic algorithms to the search space which can prevent premature convergence. The benefit also encourages exploring other areas in solution space which enhance the opportunity to discover the better solutions. The proposed algorithm has a simple structure requires much less execution time than the non-sharing compact genetic algorithm.**

***Keywords: compact genetic algorithms; parallelization of the evolutionary algorithm; parallel genetic algorithm***

## I. INTRODUCTION

In the present, there are classes of problems that are difficult to solve. The time required to solve these problems increase very quickly as the size of the problems grows. As a result, the evolutionary algorithms have been designed and developed. The genetic algorithm (GA) [1], [2] is an algorithm that is widely used to solve these problems.

In this research, the compact genetic algorithm (cGA) is focused because it is operationally equivalent to the simple genetic algorithm with uniform crossover, while it requires less memory than the simple genetic algorithm. Under the compact genetic algorithm, its population is represented by a probabilistic vector [3]. Despite constant memory throughout a run, the long execution time is required in order to obtain good solutions. Consequently, the parallelization is used in order to reduce the execution time. There are many kinds of parallelization proposed. The experiment shows that migration is an effective method to increase performance on parallelized genetic algorithms [4], [5], [6] Migrating policy and migration parameter must be meticulously defined for the best performance. Thus, the cooperation of the two compact genetic algorithms is investigated further to study how to share the knowledge between them. Most of the previous parallelization of the genetic algorithm, in migration step, always share the better solution and reject the worse one (e.g. [2], [4], and [7]). However, for cGAs sharing their probabilistic vectors, these methods may lead to getting stuck in local optima, so the new proposed cooperative compact genetic algorithm uses the worse results together with the better results among nodes. Finally, the method is put to test its ability to solve some real-world problems.

The subsequent sections start with a basic description of the compact genetic algorithm. In section III, the migration of the parallel genetic algorithm is described. Then, the next section introduces and explains the new cooperative algorithms, followed by the experiment of the algorithm to evaluate the performance and the discussion of its results in section IV. The last section shows the conclusion and the future work.

## II. COMPACT GENETIC ALGORITHM

The compact genetic algorithm is introduced by Harik, Lobo, and Goldberg [3] and proved that it is equivalent to the simple genetic algorithm with uniform crossover. The main idea is to search for some good solutions in solution space by imitating natural evolvement. Basically, its process starts by representing the whole population by a probabilistic vector. Size of the vector indicates the number of alleles required to represent solutions. However, in some cases, the size of the vector depends on the encoding of problems. Actually, there are many ways to initial values in the vector such as Edge length(EL) method for TSP that uses probabilities computed from the lengths of possible neighbors [8]; yet they are commonly set to 0.5 (Uniform distribution). After assigning values to the vector, individuals are selected based on the vector. Then, each individual is evaluated its quality by a fitness function which is specific to each problem. The final process is to update the probabilistic vector following the high potential solution according to the quality measured in the latest step. If the best solutions are not found or the results do not meet the conditions such as the amount of executed fitness functions, these steps are repeated. From the figure 1, it illustrates the pseudocode of the compact genetic algorithm and the fundamental parameters and the main steps mentioned above.

```
1) initialize probability vector
    for i = 1 to l do p[i] = 0.5

2) generate two individuals from the vector
    a = generate(p), b = generate(p)

3) let them compete
    winner, loser = evaluate(a,b)

4) update the probability vector towards the better one
    for i = 1 to l do
        if winner[i] != 1 then p[i] = p[i]+1/n
        else p[i] = p[i]-1/n

5) p represents the final solution
    compact GA parameters
    n: population size, l: chromosome length
```

Figure 1. Pseudocode of the compact genetic algorithm [10]

## III. THE DIFFERENCES OF MIGRATION IN PARALLEL GAS AND PARALLEL CGAS

The migration is a method to share information among nodes in a parallel system. For GA and cGA, the migration encourages the propagation of high-quality solutions. Although the GA and the cGA exploit the benefit of the migration for the same purposed, there are differences of migration parameters and topology because of the different structure of algorithms.

### *A. The migration in the parallel genetic algorithms*

The migration parallel GA is a way to prevent premature convergence at local optima of unsatisfied solutions. The high-quality results are disseminated throughout parallel GAs and replace low-quality individuals to improve the poor evolvement getting trapped in local optima. However, the difficulty is how to achieve a balance between convergence of solutions and diversity from independent evolvement. The short interval time of migration and the vast number of migrating individuals can destroy the balance. Because there are parameters for migration which have dramatic effects on performance such as migration size, migration topology, migration frequency and migration strategy, the parameters must be delicately determined.

### *B. The migration in the parallel compact genetic algorithms*

The parallelization of the compact genetic algorithm has an alternative migration method by using probabilistic vectors instead of individuals [9], [10]. In detailed, the impact of migrating the probabilistic vectors on search space is more significant than migrating individuals due to the fact that the whole population is distributed while transferring individuals is limited by the migration cost. However, there are the same problems as the parallel GA, the balance between convergence and diversity. One way of migration is to share solely probabilistic vectors possessing high-quality solutions of the generation or share all probabilistic vectors.

## IV. THE COOPERATIVE ALGORITHMS

The three cooperative algorithms are inspired by the effect of probabilistic vector owning low-scored individuals on search space. The algorithms enhance this concept to prevent premature convergence and increase opportunities to search in other areas. The algorithm consists of one master node and two cooperative compact GAs. The two-cooperative compact GAs independently execute to preserve diversity and frequently migrate their probabilistic vectors to the master node. The master node has functions to update the global probabilistic vector by defining and applying weights to control the impact of the shared probabilistic vectors. It then broadcasts the updated global vector to the two cGAs.

### *A. Master node algorithm*

For the first algorithm, after discovering new best solutions, the difference calculated from a probabilistic vector possessing the lower-scored individual in the generation before and after being updated gains more weight. Then, the weight of the probabilistic vector possessing the higher-scored individual in the generation slightly increases to provide some opportunities for searching in high-scored areas. However, chances for exploring in high-scored areas are limited. When the number of fitness function counted after finding the current best individual reaches a maximum, the search space will be moved to an area between itself and the initialized search space as shown at the step 3 in the figure 2. The algorithm exploits the worse solutions in this direction. On the other hand, the second algorithm is similar but puts more weight on the better probabilistic matrix in order to search in high-quality space before using the lower-scored individuals to explore other spaces. Finally, the third algorithm randomly selects one element of the two probabilistic matrices, so this algorithm does not have the maximum.

```
1) initialize probability vector, assign C constant.
2) broadcast the vector to cGAs.
3) calculate received knowledge. Weight depends on NFs and scores.
    if NF is high until reach C, go to 5) step
    if the score is higher and NF is low, get low weight
    else If the score is lower and NF is low, get high weight
    else If the score is higher and NF is high, get high weight
    else get low weight
4) update the global probability vector and back to 1).
5) the global probability vector is updated to a proportion of itself and the first initialized global probability and go back to 1) step.
```

Figure 2. Pseudocode of the master node of the first proposed algorithm

### B. *Parallel cGAs algorithm (slave node)*

Slave nodes operate as normal cGA. Selection, evaluation by fitness function and update of local probabilistic vectors occur in these nodes. However, probabilistic vectors are assigned by the master node. For the first and the second methods, the slave nodes also remember the number of executed fitness function after exploring the current best individuals called NF shown in the figure 3 to predict being caught in local optima.

NF: the number of executed fitness functions after finding the latest best solution.
1) assign received values to probability vector.
2) generate individuals from the vector.
3) evaluate the individuals.
4) update the vector and NF.
5) if reaching the time interval of migration:
Send the count, the score of the best solution in the generation and the difference of vector before and after being updated to master node.
6) if the best score in cGA is the global solution, then finish the program.
Else back to 1)

Figure 3. Pseudocode of the slave node

## V. EXPERIMENT

For performance evaluation of the proposed algorithms. Four migration methods are compared with the same migration frequency and a non-cooperative method. The experiment was controlled and run on one computer to avoid the network issue. Traveling salesman problems which is a deceptive problem [11] were used to qualify the algorithm. The TSP itself has a large number of deceptive local optima. 11 cities, 15 cities, and 17 cities with five instances (Prob. 1-5) for each of them were used in the experiment. A probabilistic matrix represents the probabilities of each path from one city to other cities is used to encode the solution. Each cGA had a probabilistic matrix initially assigned by values computed from the lengths of possible neighbor cities. Local heuristic, 2-opt [12], was exploited to reduce execution time. To be fair, the interval time of migration is fixed at 80 and all processes irrelevant to migration are the same for all tests.

### A. *Evaluation*

For each migration method, the experiment was run ten times for all instances and measured the quality using the average number of executed fitness functions of each instance. The limitation of the total number of fitness function is 500000 iterations, 250000 iterations for each slave node. Each group of instances possesses a complicated trap problem having similar and high distances among cities. As a result, the experiment sometimes did not attain the global solutions. In the case of reaching the limitation but not discovering global results, the evaluation considered the best solution acquired.

There are four migration methods and one non-cooperative cGAs executed to effectively exhibit the performance of algorithms. The main difference between the methods is the scheme to update the global probabilistic matrix. The first migration method is to randomly choose one element of the two probabilistic matrices. Adapted from a concept of the migration in genetic algorithm [2], [4] and [7], another method chooses one probabilistic matrix maintaining the better solution of the generation at the interval time. Weight is used in the third and the fourth approaches to managing the proportion between two probabilistic matrices. However, the third method puts more weight on the better probabilistic matrix, while the fourth method emphasizes on the worse matrix. Finally, the non-cooperative cGAs is two cGAs independently operating

### B. *Results and discussion*

TABLE I. THE NUMBER OF EXECUTED FITNESS FUNCTION COMPARISON BETWEEN THE PROPOSED ALGORITHM AND OTHER 4 SIMILAR MIGRATION METHODS

| | | **Migration methods** | | | | |
|---|---|---|---|---|---|---|
| | | ***Random selection(Rand)*** | ***Best selection(BST)*** | ***More weight on better scores(WB)*** | ***More weight on worse scores(WW)*** | ***Non-cooperation (Non Co)*** |
| 11 cities Prob. 1 | Best | 2 | 3 | 2 | 3 | 2 |
| | Worse | 28 | 24 | 34 | 43 | 114 |
| | Avg. | 14.9 | 17.6 | 11.8 | 14.3 | 26.2 |
| 11 cities Prob. 2 | Best | 18 | 34 | 91 | 8 | 19 |
| | Worse | 2102 | 1770 | 1833 | 2958 | 29422 |
| | Avg. | 879 | 686.2 | 782.4 | 657.3 | 5458.5 |
| 11 cities Prob. 3 | Best | 30 | 12 | 16 | 3 | 5 |
| | Worse | 625 | 665 | 723 | 803 | 8606 |
| | Avg. | 289.2 | 256.1 | 375.7 | 298 | 2415 |
| 11 cities Prob. 4 | Best | 12 | 8 | 11 | 5 | 2 |
| | Worse | 94 | 97 | 110 | 63 | 571 |
| | Avg. | 50.1 | 42.4 | 52.9 | 32.6 | 120.4 |
| 11 cities Prob. 5 | Best | 30 | 2 | 3 | 2 | 25 |
| | Worse | 269 | 338 | 372 | 221 | 3320 |
| | Avg. | 98.6 | 116.3 | 101.2 | 97 | 602.4 |
| 15 cities Prob. 1 | Best | 866 | 112 | 162 | 123 | 13 |
| | Worse | 19642 | 15302 | 14153 | 10924 | 117623 |
| | Avg. | 8657.1 | 7741.2 | 6435.8 | 3772.6 | 36996.4 |
| 15 | Best | 3 | 5 | 3 | 2 | 4 |

| | | Migration methods | | | | |
|---|---|---|---|---|---|---|
| | | ***Random selection(Rand)*** | ***Best selection(BST)*** | ***More weight on better scores(WB)*** | ***More weight on worse scores(WW)*** | ***Non-cooperation (NonCo)*** |
| cities Prob. 2 | Worse | 118 | 86 | 63 | 113 | 27678 |
| | Avg. | 23.4 | 30.6 | 32.3 | 25.8 | 5635.6 |
| 15 cities Prob. 3 | Best | 12 | 15 | 28 | 18 | 3 |
| | Worse | 1069 | 361 | 381 | 593 | 86838 |
| | Avg. | 178.7 | 166.5 | 179.1 | 157.8 | 12083 |
| 15 cities Prob. 4 | Best | 131 | 93 | 16 | 5 | 3 |
| | Worse | 4164 | 4258 | 914 | 3787 | 203992 |
| | Avg. | 1000 | 1172.6 | 500 | 932.4 | 64592.1 |
| 15 cities Prob. 5 | Best | 2 | 2 | 2 | 2 | 2 |
| | Worse | 4 | 4 | 4 | 3 | 487 |
| | Avg. | 2.8 | 2.7 | 5 | 2.7 | 53.2 |
| 17 cities Prob. 1 | Best | 27 | 39 | 248 | 97 | 5 |
| | Worse | 2276 | 1479 | 3939 | 2958 | 404744 |
| | Avg. | 1092.2 | 632.2 | 1087.5 | 1104.2 | 44415.2 |
| 17 cities Prob. 2 | Best | 347 | 974 | 15 | 149 | 1423 |
| | Worse | 2960 | 19184 | 12544 | 7475 | 436126 |
| | Avg. | 1855 | 4408.5 | 3284.4 | 2737.3 | 54712.4 |
| 17 cities Prob. 3 | Best | 6342 | 24176 | 299 | 10437 | - |
| | Worse | 206631 | 333331 | 231185 | 206347 | - |
| | Avg. | 103348.1 | 115416.6 | 91192.9 | 77455.9 | - |
| 17 cities Prob. 4 | Best | 213 | 110 | 715 | 78 | 89 |
| | Worse | 2497 | 5214 | 4490 | 2561 | 215773 |
| | Avg. | 1238.4 | 1852.8 | 1936.5 | 1030.8 | 22187.4 |
| 17 cities Prob. 5 | Best | 16538 | 1542 | 1332 | 2269 | - |
| | Worse | 125292 | 145044 | 147238 | 141917 | - |
| | Avg. | 41045 | 58394.9 | 40264.1 | 32803.8 | - |

From the table I, the number of executed fitness function is presented the best, worse and average values. WW, WB and Rand present the first, second and third proposed algorithms, sequentially. The results show that WW has excellent performance more than 10 times and almost 2 times greater than non-cooperative method (NonCo) and pure selection based on the best individual of the generation (BST) in the case of difficult instances [13], respectively.

For simple instances [13], these two methods, BST and NonCo, usually take the first and the second places of the lowest iterations of the "Best" values recorded for each test in the table since the algorithms tend to quickly convergence. In contrast, in complicated problems, premature convergence leads to being trapped in local optima. For example, from the table I, the third and fifth instances of 17 cities group are difficult problems [12]. Most final solutions of NonCo method are local optima with about 99.3 percent accuracy, the average accuracy of the solutions.

The performance of Rand and WB is slightly better than migration algorithm using solely better probabilistic matrices. However, because the differences in these results are not significant and some process of the compact genetic algorithm is based on probability, it is difficult to assure which algorithm among Rand, BST and WB is the most effective indeed. At least, the results prove that low-quality solutions can be exploited to advance the performance.

## VI. Conclusion and Further Study

Sharing knowledge in the cooperative compact genetic algorithm is presented in this paper. The proposed cooperative compact genetic algorithms are built on the benefits of using low-quality solutions to prevent premature convergence and expand search space. The results indicate the average performance of the proposed algorithms better than non-cooperative algorithm and migration method using solely the higher-scored solution. In the future, we plan to scale out the algorithm to be able to use in scalable parallelization.


## Acknowledgment

This work is supported by scholarship of the computer engineering department, Chulalongkorn University, 2018.


## References


[1] J. H. Holland, Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. Cambridge, MA: MIT Press, 2010.

[2] E. Cantù-Paz, *Efficient and accurate parallel genetic algorithms*. Boston, MA: Kluwer Academic Publishers, 2001.

[3] G. R. Harik, F. G. Lobo and D. E. Goldberg, "The compact genetic algorithm," 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360), Anchorage, AK, 1998, pp. 523-528.

[4] L. Wang, A. A. Maciejewski, H. J. Siegel and V. P. Roychowdhury, "A comparative study of five parallel genetic algorithms using the traveling salesman problem," *Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, Orlando, FL, 1998, pp. 345-349.

[5] M. Nowostawski and R. Poli, "Parallel genetic algorithm taxonomy," 1999 Third International Conference on Knowledge-Based Intelligent Information Engineering Systems. Proceedings (Cat. No.99TH8410).

[6] D. Whitley, T. Starkweather, and K. Mathias, *Optimization using distributed genetic algorithms*. Fort Collins, CO: Colorado State University, Dept. of Computer Science, 1991.

[7] F. J. Marin, O. Trelles-Salazar, and F. Sandoval, "Genetic Algorithms on LAN-message passing architectures using PVM: Application to

the Routing problem," *Parallel Problem Solving from Nature — PPSN III Lecture Notes in Computer Science*, pp. 534–543, 1994.

[8] R. Baraglia, J. I. Hidalgo and R. Perego, "A hybrid heuristic for the traveling salesman problem," in *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 613-622, Dec 2001.

[9] F. G. Lobo, C. F. Lima, and H. Mártires, "An Architecture for Massive Parallelization of the Compact Genetic Algorithm," *Genetic and Evolutionary Computation – GECCO 2004 Lecture Notes in Computer Science*, pp. 412–413, 2004.

[10] M. Pelikan, D.E. Goldberg, F. Lobo, "A survey of optimization by building and using probabilistic models", *American Control Conference 2000. Proceedings of the 2000*, vol. 5, pp. 3289-3293 vol.5, 2000, ISSN 0743-1619.

[11] D. E. Goldberg, "Simple genetic algorithm and the minimal deceptive problem," in Genetic Algorithms and Simulated Annealing, L. Davis, editor, San Mateo, CA: Morgan Kaufmann, 1987, pages 74-88.

[12] G. A. Croes, "A method for solving traveling salesman problems," *Oper. Res.*, vol. 6, no. 6, pp. 791–812, 1958

[13] K. Smith-Miles, J. V. Hemert, and X. Y. Lim, "Understanding TSP Difficulty by Learning from Evolved Instances," *Lecture Notes in Computer Science Learning and Intelligent Optimization*, pp. 266–280, 2010.

.

- **The below form will not be published, but it will help us to understand your paper better**

# Authors' background

| Name | Email | Position (Prof , Assoc. Prof. etc.) | Research Field | Homepage URL |
|---|---|---|---|---|
| Orakanya Gateratanakul | orakanyag12@gmail.com | Graduate student | Optimization | |
| Prabhas Chongstitvatana | prabhas.c@chula.ac.th | Professor | Optimization and quantum computing | https://www.cp.eng.chula.ac.th/~piak/ |
| | | | | |
| | | | | |