TRANSNAV
http://www.transnav.eu
the International Journal on Marine Navigation and Safety of Sea Transportation
Volume 14
Number 4
December 2020
DOI: 10.12716/1001.14.04.28

Cargo Vessel Route Rationalization with Chimerical Genetic Algorithm

A.L. Kuznetsov
Admiral Makarov State University of Maritime and Inland Shipping, Saint-Petersburg, Russia

G.B. Popov
"Maritime Construction and Technology" LLC, St. Petersburg, Russia

ABSTRACT: One of the most basic problems in logistics is the problem of route rationalization. Route rationalization may be based on different criterions, but at its core it always can be reduced to classic mathematical problems, such as travelling salesman problem (TSP). This study discusses methods, used to find approximate solutions for TSP and proposes authors modification of classic genetic algorithm (GA) for solving vessel's route rationalization problem. Test results and strategies for vessel's route rationalization are discussed. A number of conclusions on best strategies for route rationalization is carried out.

1 INTRODUCTION

Among many logistical problems, appearing during cargo transportation, there is always a form of route rationalization problem. Whether it is a land transportation by means of rail or road transport, or a maritime transportation, or even an air transportation, it necessary to find optimal route for moving cargo from point A to point B. Building a supply chain may involve solving this problem repeatedly for many iterations [1]. Thus, it is very important to have an instrument, supporting the decision making process, when it comes to route rationalization.

In maritime container cargo transportation, it is crucial to build economically stable closed vessel routes. Container vessels should be able to perform a round trip and return to the initial port, having delivered as many containers as possible, while also having spent least possible time to perform a trip and using shortest routes.

It is possible to reduce a route rationalization problem in its general form down to a classic travelling salesman problem (TSP). The criterion for solution optimality may be length of a route, time of cargo transportation by a certain route, amount of fuel consumed, etc. A classic TSP implies that a salesman (in this study's case – a vessel) has a task of travelling through a number of cities (sea ports), visiting each one only once. The problem is to find optimal route

2 METHODS AND MATERIALS

Theoretically it is possible to solve a route rationalization problem (or a TSP) by means of the most obvious brute force method. However, the complexity of the problem does not allow to do this in a sensible amount of time, due to faster than exponential (factorial) growth of possible outcomes with a polynomial growth of input. In fact, the travelling salesman problem is proven to be NP-hard [2]. This means, that there are no deterministic algorithms, capable of solving the problem in polynomial time. Thus, the only way to solve the problem is to use approximating algorithms, which allow to find at least a proximate solution in polynomial time.

There are many approaches to solving TSP using approximating algorithms. A number of researchers use particle swarm algorithms to solve TSP [3, 4]. Other researchers use dynamic programming [5, 6]. There is a vast plane of studies including the usage of evolutionary algorithms, specifically genetic algorithms [7, 8, 9]. It is also a common practice to use combinations of methods, such as a combination of a genetic algorithm with a dynamic programming [10].

This paper proposes the usage of a modified genetic algorithm. The core of the algorithm has already been proposed in a previous study by the authors [11]. This study considers improved version of the proposed algorithm and highlights some of the performance results of the algorithm.

Any classic genetic algorithm consists of the following steps:
1 initialization;
2 selection;
3 genetic operators (usually crossover and mutations);
4 termination.

Steps 2 and 3 are repeated until a certain condition is met or a certain amount of algorithm steps is performed, which calls the function of termination.

Optimality criterion for each solution is abstract distance between points (in this study's case sea ports). In terms of genetic algorithms theory optimality criterion is called a fitness function, and is calculated for each solution on each step of algorithm. Each sea port has its X and Y coordinates, which allows to calculate the distance between each two ports, using (1):

$$D_{i,j} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2} \tag{1}$$

This allows to build distance matrix, which includes distances between all pairs of sea ports.

A possible solution for a vessel route rationalization problem is a sequence of vessel's ports of call arranged in a certain order. The sequence should always be closed, i. e. the initial port should be the same as the final port, and there should not be any reoccurring ports. A classic crossover operator is not applicable to this problem, because of the way the genetic inheritance works. Parent solutions provide parts of their port sequences, but they do not consider, that these parts may repeat in children solutions. This is illustrated on Fig. 1, *a*.

The algorithm, suggested by authors, builds around splitting solutions in two parts: heads and tails. Each solution provides two children: one with the head of its parent and the other one with the tail of its parent. Missing parts of children are then generated randomly out of the number of missing ports. This ensures that no port in a solution sequence is repeated twice. The proposed algorithm is illustrated on Fig. 1, *b*.

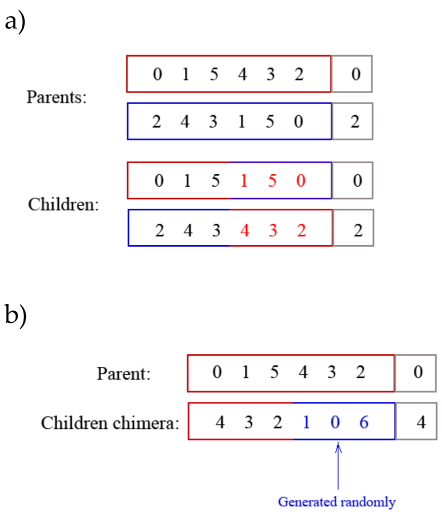
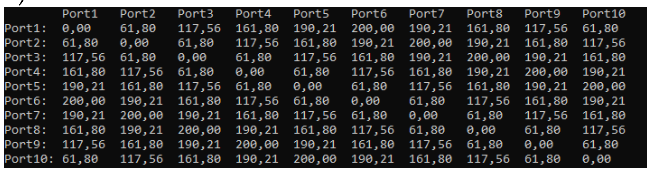


Figure 1. Port inheritance: a — classic GA; b — chimerical GA.

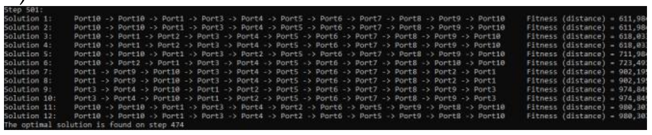
In order to keep the size of solution population constant the two resulting chimera children are compared based on their fitness functions. The best out of two solutions is selected to be carried to the next generation, the other one is destroyed.

In previous research the chimerical GA (hence the chimera children used in genetic operators stage of the algorithm) was prototyped using means of VBA and MS Excel. A more advanced model was built in AnyLogic 6. The current version of the algorithm is built in Visual Studio Community Edition 2019 as a console application written in C#. This allowed to build a more open source application, as it is not constrained by licenses of certain software.

The application is controlled by a number of console-typed commands, including commands for:
– generating ports and locating them (inputs are number of ports and X and Y coordinates);
– generating initial solution population (input is size of population);
– running chimerical GA (inputs are percent of mutations and number of algorithm steps (generations).

a)


```
         Port1   Port2   Port3   Port4   Port5   Port6   Port7   Port8   Port9   Port10
Port1:   0,00    61,80   117,56  161,80  190,21  200,00  190,21  161,80  117,56  61,80
Port2:   61,80   0,00    61,80   117,56  161,80  190,21  200,00  190,21  161,80  117,56
Port3:   117,56  61,80   0,00    61,80   117,56  161,80  190,21  200,00  190,21  161,80
Port4:   161,80  117,56  61,80   0,00    61,80   117,56  161,80  190,21  200,00  190,21
Port5:   190,21  161,80  117,56  61,80   0,00    61,80   117,56  161,80  190,21  200,00
Port6:   200,00  190,21  161,80  117,56  61,80   0,00    61,80   117,56  161,80  190,21
Port7:   190,21  200,00  190,21  161,80  117,56  61,80   0,00    61,80   117,56  161,80
Port8:   161,80  190,21  200,00  190,21  161,80  117,56  61,80   0,00    61,80   117,56
Port9:   117,56  161,80  190,21  200,00  190,21  161,80  117,56  61,80   0,00    61,80
Port10:  61,80   117,56  161,80  190,21  200,00  190,21  161,80  117,56  61,80   0,00
```

b)


```
Step 501:
Solution 1:     Port10 -> Port10 -> Port1 -> Port3 -> Port4 -> Port5 -> Port6 -> Port7 -> Port8 -> Port9 -> Port10     Fitness (distance) = 611,98
Solution 2:     Port10 -> Port10 -> Port1 -> Port3 -> Port4 -> Port5 -> Port6 -> Port7 -> Port8 -> Port9 -> Port10     Fitness (distance) = 611,98
Solution 3:     Port10 -> Port1 -> Port2 -> Port3 -> Port4 -> Port5 -> Port6 -> Port7 -> Port8 -> Port9 -> Port10      Fitness (distance) = 618,03
Solution 4:     Port10 -> Port1 -> Port2 -> Port3 -> Port4 -> Port5 -> Port6 -> Port7 -> Port8 -> Port9 -> Port10      Fitness (distance) = 618,03
Solution 5:     Port10 -> Port10 -> Port1 -> Port3 -> Port2 -> Port5 -> Port6 -> Port7 -> Port8 -> Port9 -> Port10     Fitness (distance) = 711,98
Solution 6:     Port10 -> Port2 -> Port1 -> Port3 -> Port4 -> Port5 -> Port6 -> Port7 -> Port8 -> Port10 -> Port10     Fitness (distance) = 723,49
Solution 7:     Port1 -> Port9 -> Port10 -> Port3 -> Port4 -> Port5 -> Port6 -> Port7 -> Port8 -> Port2 -> Port1        Fitness (distance) = 902,19
Solution 8:     Port1 -> Port9 -> Port10 -> Port3 -> Port4 -> Port5 -> Port6 -> Port7 -> Port8 -> Port2 -> Port1        Fitness (distance) = 902,19
Solution 9:     Port3 -> Port4 -> Port10 -> Port1 -> Port2 -> Port5 -> Port6 -> Port7 -> Port8 -> Port9 -> Port3        Fitness (distance) = 974,84
Solution 10:    Port3 -> Port4 -> Port10 -> Port1 -> Port2 -> Port5 -> Port6 -> Port7 -> Port8 -> Port9 -> Port3        Fitness (distance) = 974,84
Solution 11:    Port10 -> Port10 -> Port1 -> Port3 -> Port4 -> Port2 -> Port6 -> Port5 -> Port9 -> Port8 -> Port10     Fitness (distance) = 980,30
Solution 12:    Port10 -> Port10 -> Port1 -> Port3 -> Port4 -> Port2 -> Port6 -> Port5 -> Port9 -> Port8 -> Port10     Fitness (distance) = 980,30
The optimal solution is found on step 474
```

Figure 2. Console view examples: a — distance matrix output example; b — chimera GA step example.

In test mode the optimal fitness function is known beforehand, as all the ports are situated in a shape, closest to a circle. The resulting polygon's perimeter is the optimal solution. Termination condition is met when a solution's a fitness function equals to optimal fitness function.

Examples of a distance matrix for 10 ports and of GA step output for 12 ports are presented on Fig. 2.

3 RESULTS

Several strategies were tested with the algorithm. Table 1 and Fig 3. provides results for comparing two particular strategies. Strategy 1 considers constant size of solution population and a 15% chance of mutations on each step of the algorithm. Strategy 2 considers making size of population equal to the number of sea ports, while chance of mutations equals to 10%.

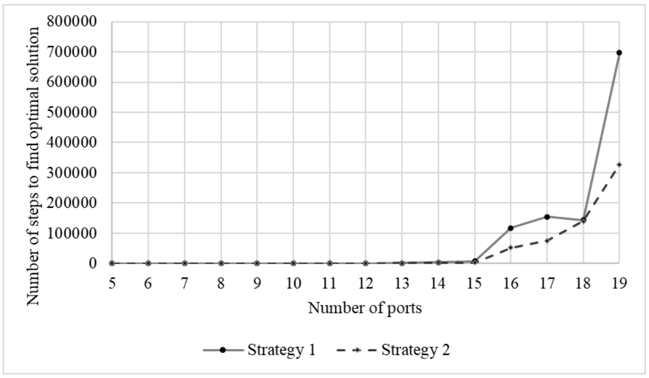
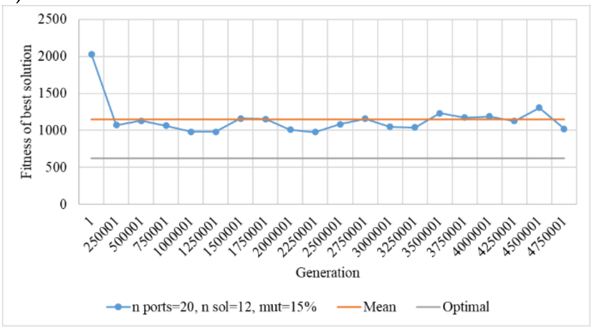


Figure 3. Comparison of strategies for running chimerical GA

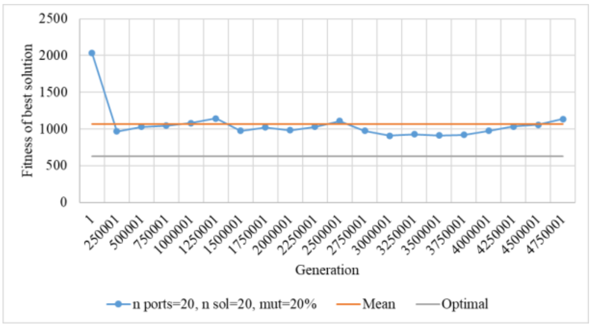
The table and figure above show that the speed of the algorithm (in steps, or generations) grows with the size of population, which is expected. However, this effect may not be the same, when comparing real time of algorithm running. Due to larger population in strategy 2, each step takes longer to calculate which leads to longer time delays in between algorithm steps. Moreover, strategy 1 (with constant 12 solutions in population) is more favorable for lower numbers of sea ports (12 and lower). This leads to strategy 3, which would suggest, that it is necessary to pick such sizes of populations, which are approximately twice the amount of sea ports. This is tested with 20 ports rationalization problem.

Results of testing strategy 3 are presented on Fig. 4 and Fig. 5. Observations were performed for each 250000th step.

a)



b)



c)

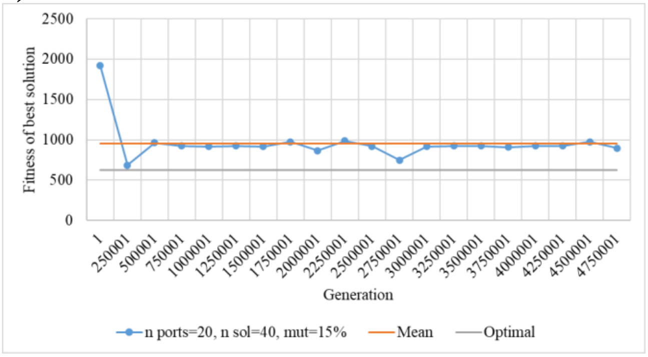


Figure 4. Strategy comparison for chimerical GA for 20 ports: a — 12 solutions, 15% mutations, b — 20 solutions, 20% mutations, c — 40 solutions, 15% mutations.

Table 1. Strategy comparison for running chimerical GA

N of ports	Strategy 1 Size of population	% mutations	Steps (generations) to find optimal solution	Strategy 2 Size of population	% mutations	Steps (generations) to find optimal solution
5	12	15%	2	5	10%	1
6	12	15%	2	6	10%	15
7	12	15%	3	7	10%	37
8	12	15%	6	8	10%	69
9	12	15%	11	9	10%	83
10	12	15%	37	10	10%	23
11	12	15%	163	11	10%	166
12	12	15%	199	12	10%	291
13	12	15%	1343	13	10%	1457
14	12	15%	4003	14	10%	3071
15	12	15%	7089	15	10%	2599
16	12	15%	117448	16	10%	52255
17	12	15%	155089	17	10%	75748
18	12	15%	143562	18	10%	139670
19	12	15%	696619	19	10%	327347

It is seen from the figures, that in all three cases the algorithm fluctuates around some mean value, finding solutions with higher and lower fitness function. However, those mean values are different and the larger the size of population, the closer the mean value becomes to the optimal solution. Moreover strategy 3 allowed to find a considerably close solution to the optimal one with fitness function of 686 miles, while the optimal fitness function is 625 miles. Best found solutions are shown on Fig. 5.

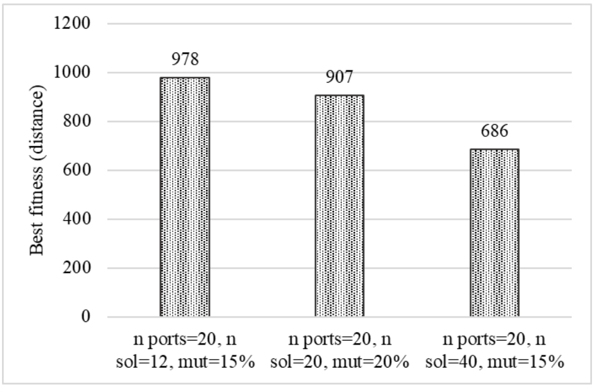


Figure 5. Best solutions, found for 20 ports with chimerical GA

Besides the fact, that fitness function of the best solution lowers with the rise of population size, the amount of solutions having lower fitness functions increases. The number of observations with sub 1000 fitness functions is 3, 9 and 19 out of 20 for strategy 1, 2 and 3 accordingly.

4 DISCUSSION

Results of this research advance further the results of authors' previous research on chimerical genetic algorithm. It is shown, that there are many possible strategies to be used for running chimerical GA. It is as important to pick the right strategy as it is important to lay down the validity of the model, based on genetic algorithms.

The dependency on the size of population is shown in the results section above. The size of solution population should grow with growth of number of sea ports for a vessel to call. The size of the population should be around twice the number of ports for reliable results to appear. However, the rate of performance of the model (in real-time seconds) should also be a consideration as well as number of steps needed to find optimal solution. Having too many initial solutions may result in significant delays in performance of the model.

Results show that proposed strategy 3 (population size is twice the number of sea ports) is preferable not only because it was the closest to optimal solution with 20 ports, but also because it is more stable. The fact that nearly every observation (19 of 20) has the best solution with fitness function of lower than 1000, while other strategies have significantly less such observations, speaks in its favor.

5 CONCLUSION

A number of conclusions is made:

1 Genetic algorithms are a common method, used for solving TSP, which means, that they are applicable for solving vessel's route rationalization problem.
2 A modification of a genetic algorithm is suggested with a modified crossover operator. The modified version of the algorithm operates with splitting a solution in two halves (heads and tails) and generating the missing half randomly. The modified algorithm is improved in this paper and is implemented with use of C# programming language.
3 Tests are performed on the proposed version of the chimerical GA, which show the robustness and validity of the algorithm.
4 Different strategies of using the algorithm are discussed. It is found, that the preferable strategy involves generating solution population with size twice the number of sea ports.

REFERENCES

[1] Koberg, Esteban, and Annachiara Longoni "A systematic review of sustainable supply chain management in global supply chains." *Journal of cleaner production* 207 (2019): 1084–1098.

[2] Orponen, P. and H. Mannila "On approximation preserving reductions: Complete problems and robust measures'", *Technical Report, Department of Computer Science, University of Helsinki* (1987): 28 p.

[3] Qian, Hao and Tao Su "Hybrid algorithm based on max and min ant system and particle swarm optimization for solving TSP problem." *33rd Youth Academic Annual Conference of Chinese Association of Automation (YAC). IEEE* (2018): 683-687.

[4] Jedrzejowicz, Piotr and Izabela Wierzbowska "Parallelized Swarm Intelligence Approach for Solving TSP and JSSP Problems." *Algorithms* 13.6 (2020): 142.

[5] Bouman, Paul, Niels Agatz and Marie Schmidt "Dynamic programming approaches for the traveling salesman problem with drone." *Networks* 72.4 (2018): 528-542.

[6] Salii, Yaroslav. "Revisiting dynamic programming for precedence-constrained traveling salesman problem and its time-dependent generalization." *European Journal of Operational Research* 272.1 (2019): 32-42.

[7] George, Tintu and T. Amudha. "Genetic Algorithm Based Multi-objective Optimization Framework to Solve Traveling Salesman Problem." *Advances in Computing and Intelligent Systems* (2020): 141-151.

[8] Juwairiah, Juwairiah, et al. "Genetic Algorithm for Optimizing Traveling Salesman Problems with Time Windows (TSP-TW)." *International Journal of Artificial Intelligence & Robotics (IJAIR)* 1.1 (2019): 1-8.

[9] Akter, Shamima, et al. "Genetic Algorithm with Updated Multipoint Crossover Technique and its Application to TSP." 2020 IEEE Region 10 Symposium (TENSYMP). IEEE, 2020.

[10] Allaoua, Hemmak. "Combination of Genetic Algorithm with Dynamic Programming for Solving TSP." *Int. J. Advance Soft Compu. Appl* 9.2 (2017): 31-44.

[11] Kuznetsov, Aleksandr L., Aleksandr V. Kirichenko, and German B. Popov. "Chimerical genetic algorithm for sea route rationalization." *Vestnik Gosudarstvennogo universiteta morskogo i rechnogo flota imeni admirala S.O. Makarova* 9.3 (2017): 456–467. DOI: 10.21821/2309-5180-2017-9-3-456-467.