

Multidisciplinary
SCIENTIFIC JOURNAL
OF MARITIME RESEARCH



University of Rijeka
FACULTY OF MARITIME STUDIES

Multidisciplinarni
znanstveni časopis
POMORSTVO

<https://doi.org/10.31217/p.36.2.9>

Pseudorandom number generator influence on the genetic algorithm performance to minimize maritime cargo delivery route length

Vadim V. Romanuke, Andriy Y. Romanov, Mykola O. Malaksiano

Odessa National Maritime University, Department of Technical Cybernetics and Information Technologies, 65029, Odessa, Mechnikova str. 34, Ukraine, e-mail: romanukevadimv@gmail.com; andreygorogogo@gmail.com; malax@ukr.net

ABSTRACT

We consider a problem of minimizing the maritime cargo delivery route length to reduce the delivery cost. In our model, the cost is equivalent to the sum of tour lengths of feeders used for the delivery to cover the route. Formulated as a multiple traveling salesman problem, we solve it with a genetic algorithm. The algorithm performance is dramatically influenced by the stream of pseudorandom numbers used for randomly generating the starting population and accomplishing random mutations. As the number of ports increases from 10 to 80, the route length variation intensifies from 3.5% to 22.5% on average. However, we increase the route length minimization accuracy by re-running the algorithm to solve the same problem until closely the best solution is obtained. The number of reruns is about 3 to 6 for up to 20 ports. For more than 20 ports the required number of algorithm reruns abruptly increases from 28 reruns for 30 ports to about 51 reruns within the range of 40 to 80 ports.

ARTICLE INFO

Original scientific paper
Received 24 August 2022
Accepted 22 November 2022

Key words:

Maritime cargo delivery
Feeder
Route length
Genetic algorithm
Pseudorandom number generator state

1 Introduction

In a rough estimation, only about one fifth of all goods are not transported by water. The amount of maritime cargo has been dramatically growing since early 1980s, and the maritime transportation is the basis of the world trading and commerce. Whereas it was only 0.1 billion metric tons in 1980, about 1.85 billion metric tons were shipped all over the world in 2020 [1].

The world fleet of containers has significantly expanded. The deadweight tonnage of container carriers since 1980 has increased from 11 up to 275 million metric tons [1]. In 2020, the global commercial shipping fleet grew by 3%, reaching 99800 ships of 100 gross tons and above [2]. By January 2021, the capacity was equivalent to 2.13 billion deadweight tons [2]. The ships delivered were mostly bulk carriers (42.77% or 913032 deadweight tons), followed by oil tankers (29.00% or 619148 deadweight tons) and container ships (13.20% or 281784 deadweight tons) [2]. Other types of ships – gas carriers, chemical tankers, offshore supplies, ferries and passenger ships delivered

about 243922 deadweight tons or 11.43% of total deliveries [2].

Compared to other types of transportation, the maritime transportation is environmentally friendly [3]. Besides, it has lesser expenses and a higher reliability allowing transporting any cargo, without limitations and restrictions. Typically the maritime transportation service prices 4 – 5 times less than of road freight and 12 – 16 times less than of air freight [4].

Ship cargo containers provide faultless protection to goods. As a result, once departed, it is a physical barrier against atmospheric conditions, temperature variations, fire, theft, and other negative impacts throughout handling. As for greenhouse effect, shipping represents 2.6% of overall emissions [5]. Comparing to airplanes, which emit 500 grams of CO₂ per metric ton of freight per kilometer of transportation, container ships emit only 10 to 40 grams of CO₂ per kilometer [6]. With much larger and heavier shiploads, shipping by sea tends to be the more cost-efficient route [7].

Despite the maritime cargo delivery is slower than other transportation types and it depends on weather conditions, the delivery speed can be increased by routing the most efficient tours for ship feeders accomplishing the delivery. The efficient tour implies its minimally possible length for a scheduled delivery. The length is expressed in units of either distance or time. The length of a delivery route consisting of efficient tours is minimized as well. The route length minimization is a transportation optimization problem equivalent to the traveling salesman problem [8, 9]. In the case of maritime cargo delivery, the traveling salesman problem solves the task of routing efficient tours of feeders. Those tours optimize the delivery cost.

The traveling salesman problem is an NP-hard problem in combinatorial optimization, whose exact solution usually takes too long to be obtained because exact algorithm perform reasonably fast only for small-sized problems [10]. Speaking formally, the exact algorithm has $N!$ permutations (as versions of the route) that need to be checked and see which one is the shortest. It is a brute-force search, and its time complexity is $O(N!)$ [11]. Because of the factorial of the number of ports, this exact algorithm becomes impractical even for only 12 ports (as $12! = 479001600$, it is difficult to run through nearly half a billion possible routes in a reasonable computational time). There are many improvements for it achieving computational times which are less than $O(N!)$ – branch and bound, linear programming, and dynamic programming techniques [12]. However, it has not been determined yet whether an exact algorithm exists that could run in time $O(1.9999^N)$ [12].

In order to achieve lower computational complexity, heuristic algorithms are widely used. Heuristic algorithms perform far much faster producing approximated solutions and saving computational resources [13, 14]. This is equivalent to saving time and budget. Depending on the heuristic algorithm, its complexity can be represented as $O(N^3)$, $O(N^2)$, $O(N \ln N)$ or even $O(N)$ [15]. One of the best heuristics is the genetic algorithm allowing to quickly find a delivery route with the minimal number of feeders tours, by which the route length is practically close to the minimum or just coincides with it [16, 17, 18]. To minimize the maritime cargo delivery route length, where multiple feeders are used, the genetic algorithm mainly requires such input parameters as follows: a map of ports to be visited en route, a number of feeders, a population size, and a set of rules to accomplish mutations. In detail, the map of ports is a set of two-coordinate port positions. The number of feeders defines the maximal number of separate tours by which the cargo can be delivered. The population size is the number of randomly generated tours to be processed by the algorithm.

2 Motivation and goal

An important part in the genetic algorithm is the pseudorandom number generator. Most genetic algorithm steps require random numbers to generate an initial population, to break a population into tours, and to perform genetic operations in order to create new tours [19]. Various pseudorandom number generators can show different results depending on the domain [20]. Minor experiments show no direct correlation between improved quality of the generator and improved performance of the algorithm [21]. In fact, higher-quality pseudorandom number generators (in terms of computational and statistical performance) in some cases worsen the genetic algorithm performance [20, 22]. This means that certain statistical conclusions on how the pseudorandom number generator influences the genetic algorithm performance are possible only after a long series of experiments. For instance, consider breaking a population into tours that is supposed to be random. If this randomness changes, does it lead to the change of the solution, or the way the algorithm converges? As of August 2022, it is an open question.

Issuing from the fact that tours of feeders are randomly generated by breaking the set of non-hub ports, the primary goal is to ascertain whether the respective pseudorandom number generator influences the genetic algorithm performance. If it does influence, the eventual goal is to increase the genetic algorithm performance by using the generator in the best way. The performance increment is implied as a further minimization of the route length. For achieving the goal, the following eight tasks are to be fulfilled:

1. To denote variables in a maritime cargo delivery model. The main assumptions used in the model are to be mentioned.
2. To formalize constraints in minimizing the maritime cargo delivery route length by a multiple traveling salesman problem.
3. To formalize an objective to be minimized.
4. To explain the place of the pseudorandom number generator in the genetic algorithm used to efficiently minimize the maritime cargo delivery route length.
5. To obtain statistics on how the algorithm performance depends on the pseudorandom number generator.
6. Based on the statistics, to make a decision on increasing the genetic algorithm performance.
7. To discuss the practical applicability and significance of the suggested decision.
8. Based on the results obtained and impartially discussed, to conclude on the contribution to the field of genetic algorithms used, in particular, to optimize maritime cargo delivery. A way of possible extension of the research and a spin-off will be outlined.

3 Variables in a maritime cargo delivery model

The following variables are used in a simplified maritime cargo delivery model: N is a number of ports, p_{k1} and p_{k2} are the horizontal and vertical components of the position of port k , and M_{\max} is a number of feeders available to accomplish the delivery. Every feeder starts its tour off port 1 and ends up by returning to that port. The port is called the hub.

We assume that if a feeder must go from port k to port j , without additional stops, then the feeder accomplishes it by a straight line with some constant speed. Due to port positions are naturally flat (no ship ascends or descends), the distance covered by the feeder from port k directly to port j (or in the opposite direction) is

$$\rho(k, j) = \sqrt{(p_{k1} - p_{j1})^2 + (p_{k2} - p_{j2})^2} = \rho(j, k) \tag{1}$$

by $k = \overline{1, N}$ and $j = \overline{k + 1, N}$.

It is clear that these $\frac{N(N-1)}{2}$ distances

$$\left\{ \left\{ \rho(k, j) \right\}_{k=1}^N \right\}_{j=k+1}^N \tag{2}$$

are nonzero. We assume that distances (2) are linearly mapped into durations of the maritime cargo delivery. The durations can be subsequently mapped into the respective delivery costs that should be minimized.

If feeder m visits either port j after port k or port k after port j (the direction here does not matter), then this fact is flagged as $x_{kjm} = 1$. To avoid surplus flagging, $x_{jkm} = 0$ if $x_{kjm} = 1$ and $x_{kjm} = 0$ if $x_{jkm} = 1$ for non-two-port tours (a two-port tour is when a feeder departs from the hub towards a port and then returns back to the hub). In other words, $x_{kjm} = 1$ if feeder m departs from port k and then directly arrives to port j ; otherwise $x_{kjm} = 0$. If feeder m does not visit port j after port k nor port k after port j , then $x_{kjm} = 0$. This, however, does not imply that ports k and j are not included into the tour of feeder m . Note that for a two-port tour, when feeder m departs from the hub towards port k and then returns back to the hub, we have $x_{1km} = x_{k1m} = 1$.

4 Constraints

Usually, there are at least two available feeders. Hence, $M_{\max} \in \mathbb{N} \setminus \{1\}$. An additional aim is to enable as less feeders as possible. We denote a current number of feeders by M , so

$$M \leq M_{\max} \tag{3}$$

Each flag

$$x_{kjm} \in \{0, 1\} \text{ by } k = \overline{1, N} \text{ and } j = \overline{1, N} \text{ and } m = \overline{1, M}. \tag{4}$$

Each of M feeders only once departs from the hub that is controlled with an equality

$$\sum_{j=2}^N \sum_{m=1}^M x_{1jm} = M. \tag{5}$$

Each of M feeders only once arrives at the hub that is controlled with a symmetric equality

$$\sum_{k=2}^N \sum_{m=1}^M x_{k1m} = M. \tag{6}$$

Another two constraints control the similar depart-and-arrive logic. Only one feeder can depart from port k , being not the hub, towards only one following port (which can be the hub) that is controlled with an equality

$$\sum_{j=1}^N \sum_{m=1}^M x_{kjm} = 1 \quad \forall k = \overline{2, N}. \tag{7}$$

Symmetrically, only one feeder can arrive at port j , being not the hub, from only one port (which can be the hub) that is controlled with an equality

$$\sum_{k=1}^N \sum_{m=1}^M x_{kjm} = 1 \quad \forall j = \overline{2, N}. \tag{8}$$

We eliminate any subtour of a feeder by imposing the following requirement: for every tour

$$T_m = \left\{ 1, \left\{ q_i^{(m)} \right\}_{i=2}^{A_m} \right\} \subset \left\{ \overline{1, N} \right\} \tag{9}$$

of feeder m we require that

$$\sum_{k \in Q_m} \sum_{j \in Q_m \setminus \{k\}} x_{kjm} \leq |Q_m| - 1 \quad \forall Q_m \subset T_m \subset \left\{ \overline{1, N} \right\} \tag{10}$$

by $2 \leq |Q_m| < A_m$ and $\forall m = \overline{1, M}$.

Constraint (10) for (9) ensures that every feeder has a tour as a closed loop: it departs from the hub and arrives at it. Owing to this constraint, a feasible route of delivering maritime cargo is of closed loops only, where every loop is a feeder tour starting off the hub and ending up by returning to the hub.

Obviously, every feeder has a limit of distance it can cover without fuel refill. We denote this limit by d_{\max} . Besides, enabling a feeder into delivery is also expensive. Therefore, a lower limit of the distance the feeder should cover must exist. We denote this lower limit by d_{\min} . So, in addition to constraints (3) - (10), inequalities

$$\sum_{k=1}^N \sum_{j=1}^N \rho(k, j) \cdot x_{kjm} \geq d_{\min} \quad \forall m = \overline{1, M} \tag{11}$$

and

$$\sum_{k=1}^N \sum_{j=1}^N \rho(k, j) \cdot x_{kjm} \leq d_{\max} \quad \forall m = \overline{1, M} \tag{12}$$

constrain the tour of every feeder. Thus, a feasible feeder tour length lies between d_{\min} and d_{\max} .

5 Objective to be minimized

To optimize the maritime cargo delivery, the sum of all the tours of the feeders is to be minimized. The respective objective function

$$\rho_{\Sigma} \left(N, M, \left\{ \left\{ \left\{ x_{kjm} \right\}_{k=1}^N \right\}_{j=1}^N \right\}_{m=1}^M, d_{\min}, d_{\max} \right) = \sum_{k=1}^N \sum_{j=1}^N \sum_{m=1}^M x_{kjm} \cdot \rho(k, j) \tag{13}$$

is to be minimized subject to flags (4) and constraints (3), (5) – (12) by (1) and $\rho(k, k) = 0 \quad \forall k = \overline{1, N}$. The minimization is implied to be done over binary variables (4) along with trying to minimize the total number of feeders used in the tours. That is, the minimization goal is to find such

$$M^* \in \{ \overline{1, M_{\max}} \} \tag{14}$$

and

$$x_{kjm}^* \in \{0, 1\} \text{ for } k = \overline{1, N} \text{ and } j = \overline{1, N} \text{ by } m = \overline{1, M^*} \tag{15}$$

at which

$$\begin{aligned} & \sum_{k=1}^N \sum_{j=1}^N \sum_{m=1}^{M^*} x_{kjm}^* \cdot \rho(k, j) = \\ & = \rho_{\Sigma} \left(N, M^*, \left\{ \left\{ \left\{ x_{kjm}^* \right\}_{k=1}^N \right\}_{j=1}^N \right\}_{m=1}^{M^*}, d_{\min}, d_{\max} \right) = \\ & = \min_{\substack{\{ \{ \{ x_{kjm}^* \}_{k=1}^N \}_{j=1}^N \}_{m=1}^{M^*} \\ m=1, M, M=1, M_{\max}}} \sum_{k=1}^N \sum_{j=1}^N \sum_{m=1}^M x_{kjm} \cdot \rho(k, j). \end{aligned} \tag{16}$$

The solution given formally as

$$\left\{ \left\{ \left\{ x_{kjm}^* \right\}_{k=1}^N \right\}_{j=1}^N \right\}_{m=1}^{M^*} \tag{17}$$

allows to build a set of M^* the most rational tours of M^* feeders. Sum (16) of these tours is the length of the shortest route to deliver maritime cargo and return to the hub. For simplicity, we presume this length to be equivalent to the delivery costs.

The solution to problem (16) may be not unique. There may be two shortest routes whose lengths are equal. Besides, one of those routes can be covered with a lesser number of feeders. Then the route covered by the lesser number of feeders is accepted as the unique solution. If both the shortest routes are covered by the same number of feeders, an additional criterion to select a route should be formulated. Usually, it is a criterion of maximizing the minimal tour length. This is done to equalize the lengths of feeders tours. In this way, as the minimal tour length is maximized, the maximal tour length is minimized.

6 Genetic algorithm and pseudorandom number generator

One of the best genetic algorithms presented in [18] solves problem (16) subject to constraints (3) – (8), (10) – (12) in reasonable computational time. It takes roughly 2.7 to 3.9 seconds to find the approximately shortest route for 20 ports on a 3 MHz processor core. A problem with 50 ports is solved in 9 to 22 seconds. Amazingly enough, the same problem is solved in various time spans after re-running the algorithm. Moreover, the resulting outcome as the route and its length vary as well.

Denote by H_m the number of ports which feeder m should visit after starting off the hub before returning to the hub (the hub is not counted in this number). The succession of non-hub ports which feeder m should visit is a vector

$$\mathbf{F}_m = [f_h^{(m)}]_{1 \times H_m} \tag{18}$$

where

$$f_h^{(m)} \in \{ \overline{2, N} \} \quad \forall h = \overline{1, H_m} \text{ and } \forall m = \overline{1, M}.$$

Vector (18) is a tour of feeder m (by convention, the hub can be not counted). The set of all non-hub ports is

$$\{ \overline{2, M} \}. \tag{19}$$

Tours

$$\{ \mathbf{F}_m \}_{m=1}^M \tag{20}$$

of all feeders constitute a route of the delivery (the hub is conventionally not mentioned but is factually kept). This means that

$$\bigcup_{m=1}^M \{ f_h^{(m)} \}_{h=1}^{H_m} = \{ \overline{2, N} \}. \tag{21}$$

Before the genetic algorithm runs into the very first iteration, tours (20) are randomly generated by breaking the set of non-hub ports (19). This random generation depends on the pseudorandom number generator state. In particular, a stream of pseudorandom numbers are initialized with a seed which is an integer between 0 and $2^{32} - 1$. We are using the Mersenne Twister generator henceforward [23].

Each feeder has a series of pseudorandom tours called chromosomes. Altogether M series of all the feeders constitute a population. Each element of the population is a route of the delivery using M feeders represented as M respective chromosomes. For every route of the population, the following routine is executed during an iteration of the algorithm. First, the distance to the port following the hub is calculated as

$$d_m = \rho(1, f_1^{(m)}). \tag{22}$$

Then, the remaining distances except the last one are accumulated into d_m :

$$d_m^{(obs)} = d_m, \tag{23}$$

$$d_m = d_m^{(obs)} + \rho(f_k^{(m)}, f_{k+1}^{(m)}) \text{ for } k = \overline{1, H_m - 1}.$$

Finally, the distance of returning to the hub is:

$$d_m^{(obs)} = d_m, \quad d_m = d_m^{(obs)} + \rho(f_{H_m}^{(m)}, 1). \tag{24}$$

To improve selectivity of the best feeder tours, tours which violate conditions (11) or (12) are expunged. Thus, if $d_m > d_{max}$ then a current accumulated distance d_m after (24) is increased exactly by the excess:

$$d_m^{(obs)} = d_m, \tag{25}$$

$$d_m = d_m^{(obs)} + d_m^{(obs)} - d_{max} = 2d_m^{(obs)} - d_{max}.$$

If $d_m < d_{min}$ then a current accumulated distance d_m after (24) is increased exactly by the shortage:

$$d_m^{(obs)} = d_m, \quad d_m = d_m^{(obs)} + d_{min} - d_m^{(obs)} = d_{min}. \tag{26}$$

Finally, sum

$$\begin{aligned} \tilde{\rho}_\Sigma(N, M, \{F_m\}_{m=1}^M, d_{min}, d_{max}) &= \sum_{m=1}^M d_m \geq \\ &\geq \rho_\Sigma\left(N, M^*, \left\{ \left\{ \left\{ x_{kjm}^* \right\}_{k=1}^N \right\}_{j=1}^N \right\}_{m=1}^{M^*}, d_{min}, d_{max} \right) \end{aligned} \tag{27}$$

is calculated and minimized over the population.

A new population is generated based on four forms of chromosome mutations: flip, swap, slide, and crossover [16, 17, 18]. The flip operator swaps a random sequence of ports inside a chromosome. Going into details, a chromosome

$$F_r = [f_h^{(r)}]_{1 \times H_r} \text{ by } r \in \overline{1, M} \tag{28}$$

is taken, within which the sequence

$$\{f_h^{(r)}\}_{h=h_1+1}^{h_2} \subset \{f_h^{(r)}\}_{h=1}^{H_r} \tag{29}$$

is extracted and flipped:

$$F_r^{(obs)} = [f_h^{(r)(obs)}]_{1 \times H_r} = F_r,$$

$$F_r = [f_h^{(r)}]_{1 \times H_r} \text{ by } f_h^{(r)} = f_h^{(r)(obs)} \quad \forall h = \overline{1, h_1} \text{ and} \tag{30}$$

$$f_h^{(r)} = f_{h_2+h_1-h+1}^{(r)(obs)} \quad \forall h = \overline{h_1+1, h_2} \text{ and}$$

$$f_h^{(r)} = f_h^{(r)(obs)} \quad \forall h = \overline{h_2+1, H_r}.$$

The flip operator returns then an updated vector (28) after (30). Indices h_1 and h_2 are pseudorandom integers produced by the same pseudorandom number generator that breaks the set of non-hub ports (19).

The swap operator randomly chooses the same-index-and-length sequence of ports from two chromosomes and interchange them. The pseudorandom number generator produces pseudorandom integers h_3 and h_4 , whereupon within two chromosomes (28) and

$$F_q = [f_h^{(q)}]_{1 \times H_q} \text{ by } q \in \overline{1, M} \tag{31}$$

for $r \neq q$ sequences

$$\{f_h^{(r)}\}_{h=h_3+1}^{h_4} \subset \{f_h^{(r)}\}_{h=1}^{H_r} \tag{32}$$

and

$$\{f_h^{(q)}\}_{h=h_3+1}^{h_4} \subset \{f_h^{(q)}\}_{h=1}^{H_q} \tag{33}$$

are interchanged:

$$F_r^{(obs)} = [f_h^{(r)(obs)}]_{1 \times H_r} = F_r,$$

$$F_r = [f_h^{(r)}]_{1 \times H_r} \text{ by } f_h^{(r)} = f_h^{(r)(obs)} \quad \forall h = \overline{1, h_3} \text{ and}$$

$$f_h^{(r)} = f_h^{(q)} \quad \forall h = \overline{h_3+1, h_4} \text{ and} \tag{34}$$

$$f_h^{(r)} = f_h^{(r)(obs)} \quad \forall h = \overline{h_4+1, H_r}$$

and

$$F_q^{(obs)} = [f_h^{(q)(obs)}]_{1 \times H_q} = F_q,$$

$$F_q = [f_h^{(q)}]_{1 \times H_q} \text{ by } f_h^{(q)} = f_h^{(q)(obs)} \quad \forall h = \overline{1, h_3} \text{ and}$$

$$f_h^{(q)} = f_h^{(r)(obs)} \quad \forall h = \overline{h_3+1, h_4} \text{ and} \tag{35}$$

$$f_h^{(q)} = f_h^{(q)(obs)} \quad \forall h = \overline{h_4+1, H_q}.$$

The swap operator returns then updated vectors (28) and (31) after (34) and (35), respectively.

The slide operator moves the last port from each chromosome to the beginning of another one. This is the only mutation operator that does not invoke the pseudorandom number generator. The crossover operator takes two chromosomes (28) and (31) for $r \neq q$, whereupon they are either interchanged or merged. This is done with using a merging probability P_{merge} given at the input of the genetic algorithm. If $\theta \leq P_{merge}$, where θ is a random value drawn (produced by the generator) from the standard uniform distribution on the open interval (0; 1), then chromosomes (28) and (31) as tours of two different feeders r and q are merged into a single tour:

$$F_{r \cup q}^* = \{f_h^{(r)}\}_{h=1}^{H_r} \cup \{f_h^{(q)}\}_{h=1}^{H_q} \subseteq \overline{2, N}. \tag{36}$$

This allows to decrease the number of feeders used to deliver maritime cargo. Otherwise, if $\theta > P_{\text{merge}}$ then each chromosome is cut in two random parts. For this, integers h_r and h_q are produced by the pseudorandom number generator to leave h_r first ports in chromosome (28) and to leave h_q first ports in chromosome (31). Then the remaining parts are interchanged:

$$\mathbf{F}_r^* = \left[f_h^{(r)*} \right]_{1 \times (h_r + H_q - h_q)} = \left\{ \left\{ f_h^{(r)} \right\}_{h=1}^{h_r}, \left\{ f_h^{(q)} \right\}_{h=h_q+1}^{H_q} \right\} \quad (37)$$

and

$$\mathbf{F}_q^* = \left[f_h^{(q)*} \right]_{1 \times (h_q + H_r - h_r)} = \left\{ \left\{ f_h^{(q)} \right\}_{h=1}^{h_q}, \left\{ f_h^{(r)} \right\}_{h=h_r+1}^{H_r} \right\}. \quad (38)$$

If $M = 2$ then

$$\mathbf{F}_{r \cup q}^* = \mathbf{F}_{1 \cup 2}^* = \{ \overline{2}, N \}$$

and so the merged two chromosomes constitute a route of the delivery (apart from the hub). If the merging is done by $M \geq 3$ then

$$\mathbf{F}_{r \cup q}^* \subset \{ \overline{2}, N \} \text{ and } \mathbf{F}_{r \cup q}^* \neq \{ \overline{2}, N \}$$

and single tour (36) is a part of the route.

7 Pseudorandom number generator influence

It is quite obvious that as generating the starting population and accomplishing the flip, swap, crossover mutations depend on the pseudorandom number generator state, the genetic algorithm iterations depend on the state. Nevertheless, it does not mean that the algorithm output, i. e. the approximately shortest route and its length after the last iteration, depends as deeply on the state as the first iteration does. While the algorithm is initialized with different states, its convergence might be so strong that the final result would be the same, whichever the state is. In such a case, the influence of the pseudorandom number generator on the genetic algorithm performance might be only in the convergence speed.

To ascertain whether the pseudorandom number generator state influences the route returned by the algorithm and its convergence speed, we denote by

$$\tilde{\rho}_\Sigma^*(N; s, u) = \tilde{\rho}_\Sigma(N, M^*, \{ \mathbf{F}_m \}_{m=1}^M, d_{\min}, d_{\max}; s, u) \quad (39)$$

the shortest route length found by state s and stream u of pseudorandom numbers in $I^*(N; s, u)$ iterations. Integer s is not a seed (an integer between 0 and $2^{32} - 1$) but is a sequential number of the state initializing a stream of pseudorandom numbers to be used for randomly generating the starting population and accomplishing the random flip, swap, crossover mutations. Integer u is a sequential

number of a sub-stream taken sequentially from the stream. Thus, we run the algorithm for every

$$N \in \{ 10, 15, \{ 10 + 10n \}_{n=1}^7 \} \quad (40)$$

and $s = \overline{1, 200}$ (i. e., it is 10 to 80 ports). The test for pair $\{N, s\}$ is repeated for 100 times ($u = \overline{1, 100}$) to obtain reliable and stable statistical data. This means that, once initialized with state s , the algorithm is run for 100 times, where every next run is initialized and supported by continuing the stream of pseudorandom numbers (by state s). The percentage

$$g(N; s) = 100 \cdot \frac{\max_{u=1, 100} \tilde{\rho}_\Sigma^*(N; s, u) - \min_{u=1, 100} \tilde{\rho}_\Sigma^*(N; s, u)}{\min_{u=1, 100} \tilde{\rho}_\Sigma^*(N; s, u)} \quad (41)$$

will show how badly the algorithm output varies at a given number of ports and state. We study such statistics of (41) as its minimal, maximal, average, and standard deviation values, respectively:

$$g_{\min}(N) = \min_{s=1, 200} g(N; s), \quad (42)$$

$$g_{\max}(N) = \max_{s=1, 200} g(N; s), \quad (43)$$

$$\bar{g}(N) = \frac{1}{200} \cdot \sum_{s=1}^{200} g(N; s), \quad (44)$$

$$\sigma_g(N) = \sqrt{\frac{1}{200} \cdot \sum_{s=1}^{200} [g(N; s) - \bar{g}(N)]^2}. \quad (45)$$

The percentage

$$v(N; s) = 100 \cdot \frac{\max_{u=1, 100} I^*(N; s, u) - \min_{u=1, 100} I^*(N; s, u)}{\min_{u=1, 100} I^*(N; s, u)} \quad (46)$$

will show how badly the algorithm convergence speed varies at a given number of ports and state. We similarly study minimal, maximal, average, and standard deviation values of (46), respectively:

$$v_{\min}(N) = \min_{s=1, 200} v(N; s), \quad (47)$$

$$v_{\max}(N) = \max_{s=1, 200} v(N; s), \quad (48)$$

$$\bar{v}(N) = \frac{1}{200} \cdot \sum_{s=1}^{200} v(N; s), \quad (49)$$

$$\sigma_v(N) = \sqrt{\frac{1}{200} \cdot \sum_{s=1}^{200} [v(N; s) - \bar{v}(N)]^2}. \quad (50)$$

At a given N and s every port position components $\{p_{k1}, p_{k2}\}$ are produced by the pseudorandom number generator:

$$p_{k1} = 100 \cdot \theta_1 \text{ and } p_{k2} = 100 \cdot \theta_2, k = \overline{1, N} \tag{51}$$

where θ_1 and θ_2 are pseudorandom numbers independently drawn from the standard uniform distribution on the open interval $(0; 1)$. The maximal number of iterations is 5000, whereas the algorithm early stop condition is used, by which (a run of) the algorithm is stopped if the currently found shortest route length does not change for 500 iterations (a one 10-th of the maximal number of iterations). The remaining parameters are: the tour length should lie between

$$d_{\min} = 250 \text{ and } d_{\max} = 500 \tag{52}$$

the merging probability is $P_{\text{merge}} = 0.05$, and the (maximal) number of available feeders is

$$\begin{aligned} M_{\max} &= \zeta \left(\frac{2N}{250N(N-1)} \cdot \sum_{k=1}^N \sum_{j=k+1}^N \rho(k, j) \right) = \\ &= \zeta \left(\frac{1}{125 \cdot (N-1)} \cdot \sum_{k=1}^N \sum_{j=k+1}^N \rho(k, j) \right). \end{aligned} \tag{53}$$

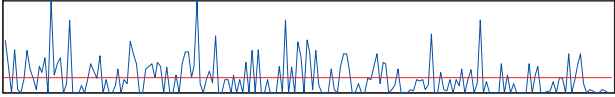
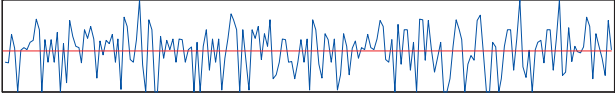
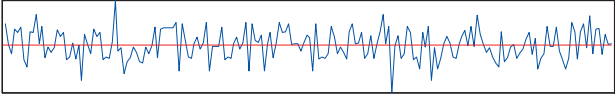
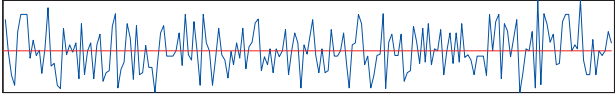
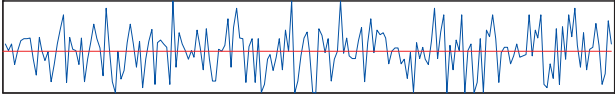
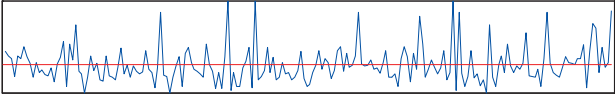
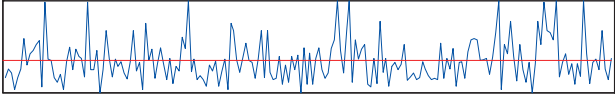
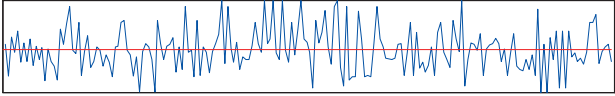
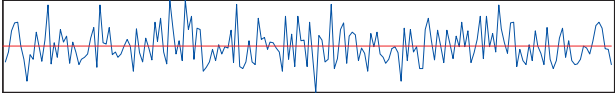
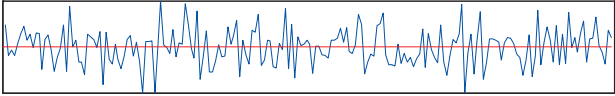
where function $\zeta(x)$ rounds number x to the nearest integer towards infinity. The difference between constraints (11) and (12) by (52) is particularly tight, where number (53) is the mean distance multiplied by the number of ports and divided by the minimal tour length (which is 250). This is intentionally done to study nearly the worst-case scenario. According to (53), the number of available feeders is between 2 and 5 when the number of ports is between 10 and 20. As the number of ports increases, M_{\max} is increased as well. It is 16 to 18 feeders for 80 ports.

The statistical data of how the algorithm output varies versus the pseudorandom number generator state are presented in Table 1. The minimum of the length variation percentage is 0 for up to 20 ports. As the number of ports

increases, the minimum increases. The maximum of the length variation percentage is increasing also, but it seemingly has peaks at 30 to 40 ports. The average has the same pattern. The standard deviation does not have a distinct pattern. Its minimum is at 80 ports, and maximum is at 40 ports. The plots of percentage (41) versus the pseudorandom number generator state in Table 1, where the horizontal line is the average by (44), show that the percentage stochasticity resembles a noise. The statistical data are reliable and stable: the overall averages in Table 1 does not change much as we narrow the number of pseudorandom number generator states (from 200) down to 100.

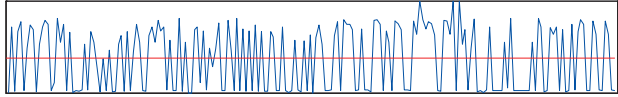
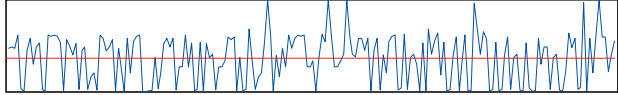
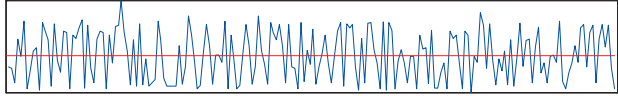
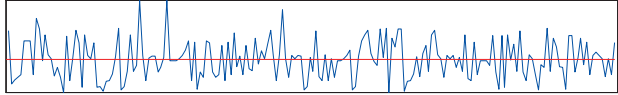
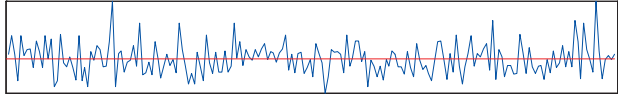
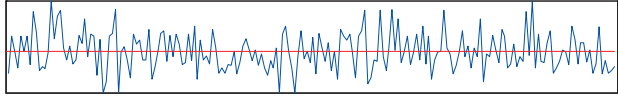
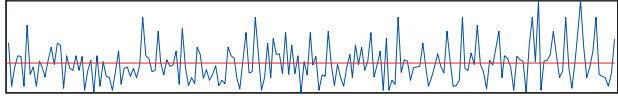
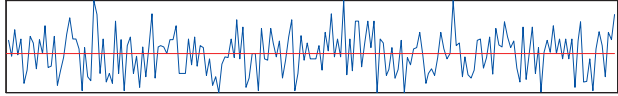
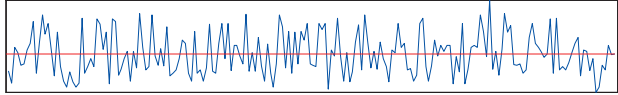
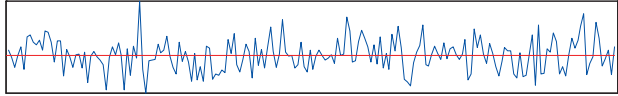
The statistical data of how the algorithm convergence speed varies versus the pseudorandom number generator state are presented in Table 2. The minimum of the convergence speed variation percentage is at 10 ports. As the number of ports increases, the minimum abruptly increases reaching its maximum at 70 to 80 ports. The maximum of the convergence speed variation percentage does not have the same pattern, but it is huge enough not dropping below 200% within the range of 30 to 80 ports. The average of the convergence speed variation percentage is increasing less abruptly, and it has a peak at 40 ports. Nevertheless, the average is huge enough also: it does not drop below 100% within the range of 30 to 80 ports, and it is more than 135% within the range of 40 to 80 ports. The standard deviation does not have a distinct pattern except for a non-stable trend to decrease. Its minimum is at 80 ports, and maximum is at 10 to 30 ports. The plots of percentage (46) versus the pseudorandom number generator state in Table 2, where the horizontal line is the average by (49), show that the percentage stochasticity resembles a noise similar to that in the plots in Table 1. The only peculiarity is that the noise-like plot for 10 ports is cut from below at some pseudorandom number generator states. Amazingly enough, the algorithm convergence speed statistical data are reliable and stable as well: the overall averages in Table 2 does not change much as we narrow the number of pseudorandom number generator states (from 200) down to 100 or even less.

Table 1 Statistics of the algorithm output variation by percentage (41)

		Statistics of (41)				Plot of percentage (41) versus the pseudorandom number generator state
		Minimum by (42)	Maximum by (43)	Average by (44)	Standard deviation by (45)	
N	10	0	21.2568	3.3866	4.1971	
	15	0	22.1689	9.8544	5.332	
	20	0	25.9913	13.4554	4.1506	
	30	10.1196	36.2552	21.9453	6.0325	
	40	9.5097	36.4187	21.6446	6.2765	
	50	9.0205	31.1044	15.7445	4.3032	
	60	10.3096	30.8677	17.4724	4.6683	
	70	11.1106	30.4186	20.0754	4.4597	
	80	13.1976	31.4215	22.3306	3.3899	
Overall average		7.0297	29.5448	16.2121	4.7567	

Source: Authors

Table 2 Statistics of the algorithm convergence speed variation by percentage (46)

		Statistics of (46)				Plot of percentage (46) versus the pseudorandom number generator state
		Minimum by (47)	Maximum by (48)	Average by (49)	Standard deviation by (50)	
N	10	2.1956	126.1297	48.7671	41.7183	
	15	9.1954	154.2146	61.4868	37.0163	
	20	13.7112	128.5714	59.8877	30.0487	
	30	33.28	232.5321	104.3324	40.9385	
	40	92.2194	282.684	162.6262	32.1158	
	50	91.967	209.9783	144.8646	23.9795	
	60	91.965	234.1051	137.0457	29.3096	
	70	98.8651	203.0236	142.8845	23.5172	
	80	101.2514	202.1148	143.11	22.6291	
Overall average		59.4056	197.0393	111.6672	31.2526	

Source: Authors

Table 3 The average number of algorithm reruns to make inequality (55) true

	μ	0	1	2	3	4	5	7.5	10
N	10	2.315	2.195	2.02	1.945	1.935	1.46	1.005	1
	15	2.035	1.56	1.37	1.19	1.145	1.07	1.02	1
	20	5.265	3.235	2.475	2.105	1.81	1.72	1.215	1.045
	30	27.42	10.265	5.4	3.88	2.59	2.215	1.66	1.18
	40	42.185	25.595	18.97	13.195	11.415	9.475	7.01	5.195
	50	44.605	20.505	11.01	7.23	5.98	4.975	3.665	1.515
	60	49.54	24.92	11.945	5.7	4.465	3.63	2.99	2.54
	70	50.445	25.58	13.745	7.35	5.27	3.52	1.65	1.26
	80	47.485	32.09	16.45	9.065	5.09	3.495	1.845	1.205

Source: Authors

As we see, the algorithm performance does deeply depend on the pseudorandom number generator. The algorithm convergence speed is influenced far deeper than the factual output (the shortest route length). However, the minimum of the shortest route length equal to

$$\tilde{\rho}_{\Sigma}^{**}(N; s) = \min_{u=1, 100} \tilde{\rho}_{\Sigma}^*(N; s, u) \tag{54}$$

for each pair $\{N, s\}$ is unlikely to be reached fast, after a few starting reruns of the algorithm. This is shown as follows. For each pair $\{N, s\}$ we calculate the average number of algorithm reruns taken to make inequality

$$\tilde{\rho}_{\Sigma}^*(N; s, u) \leq \left(1 + \frac{\mu}{100}\right) \cdot \tilde{\rho}_{\Sigma}^{**}(N; s) \tag{55}$$

true (checked over $u = \overline{1, 100}$) for the first time for a factor $\mu \in \{0, 1, 3, 4, 5, 7.5, 10\}$.

Factor $\mu = 0$ implies the average number of algorithm reruns to reach minimum (54) itself. Factor $\mu = 1$ implies the average number for reaching the shortest route length within 1% tolerance with respect to minimum (54) (i. e., the length will not exceed 1% over the minimum). Factor $\mu = 10$ implies the average number by which the length will not exceed 10% over the minimum. The distribution of the average number versus the number of ports and factor (56) is presented in Table 3. Obviously, it takes less algorithm reruns to reach the shortest route length differing from minimum (54) by a greater deviation. Besides, it takes a greater number of reruns as the number of ports increases. To reach minimum $\tilde{\rho}_{\Sigma}^{**}(10; s)$ (in other words, to solve a route length minimization problem for 10 ports in the best way), the algorithm should be re-run for 3 times (the average number ceiling). Roughly the same concerns the problem with 15 ports. To reach minimum $\tilde{\rho}_{\Sigma}^{**}(20; s)$, it expectedly will take 5-6 reruns. For more than 20 ports the required number of algorithm reruns abruptly increases: it is about 28 reruns for 30 ports and 42 to 51 reruns within the range of 40 to 80 ports. The minimum with 1% tolerance is reached faster – it is approximately

twice as faster for 50 to 70 ports, although it still takes over 30 reruns to reach length $1.01 \cdot \tilde{\rho}_{\Sigma}^{**}(80; s)$. Meanwhile, we notice that the statistics for 40 ports apparently has a computational artifact – the average number of algorithm reruns remains too big even at 4% to 10% tolerance. The artifact is noticeable at $\mu = 3$, though. It must be clear that the peak at 40 ports mentioned above for both Tables 1 and 2 is related to this artifact.

The genetic algorithm performance is indeed increased by re-running the genetic algorithm in accordance with Table 3. For 10 to 20 ports, the algorithm reruns can be easily parallelized. Ignoring this recommendation leads to losses in the accuracy – see the average percentage by (44) in Table 1. An example of the potential loss for 10 ports, where a single feeder is used, is presented in Figure 1, where the difference between the best and worst solutions is 21.2568% (this is the maximum seen in Table 1). In maritime cargo delivery, this is a huge loss (because the delivery cost unreasonably increases by more than one fifth).

A far more demonstrative example, for 30 ports, is presented in Figure 2. The same problem is solved by the same algorithm by two different pseudorandom number generator states, and the difference between two solutions is 36.2552% (this is the maximum seen in Table 1). This is an extremely huge loss – the delivery cost unreasonably increases by more than one third. Moreover, the worst route is covered by two feeders, whereas the best route is covered by a single feeder. An interesting fact is that the worst route is found in 835 iterations, and the best route can be found in either 1056 or 740 iterations. Consequently, a pseudorandom number generator state may exist (initializing a stream of pseudorandom numbers) such by which both the algorithm accuracy and its convergence speed are better than for other states.

The statistical data also have shown that in most cases the worst route is not obtained by the worst convergence speed (the maximal number of iterations), and the best route is not obtained by the best convergence speed (the minimal number of iterations). Admittedly, a few exclusions from this experience exist. Thus, an instance with 20

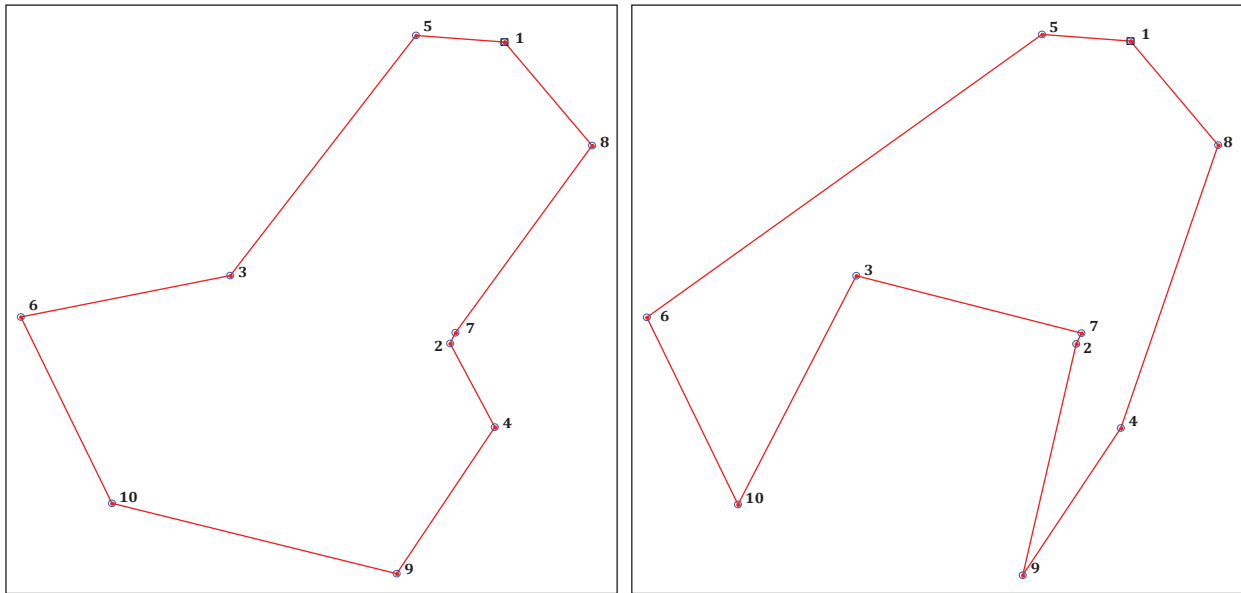


Figure 1 The best (on the left, the route length is 292.2456) and worst (on the right, the route length is 354.3676) solutions (out of 100 solutions) of an instance with 10 ports

Source: Authors

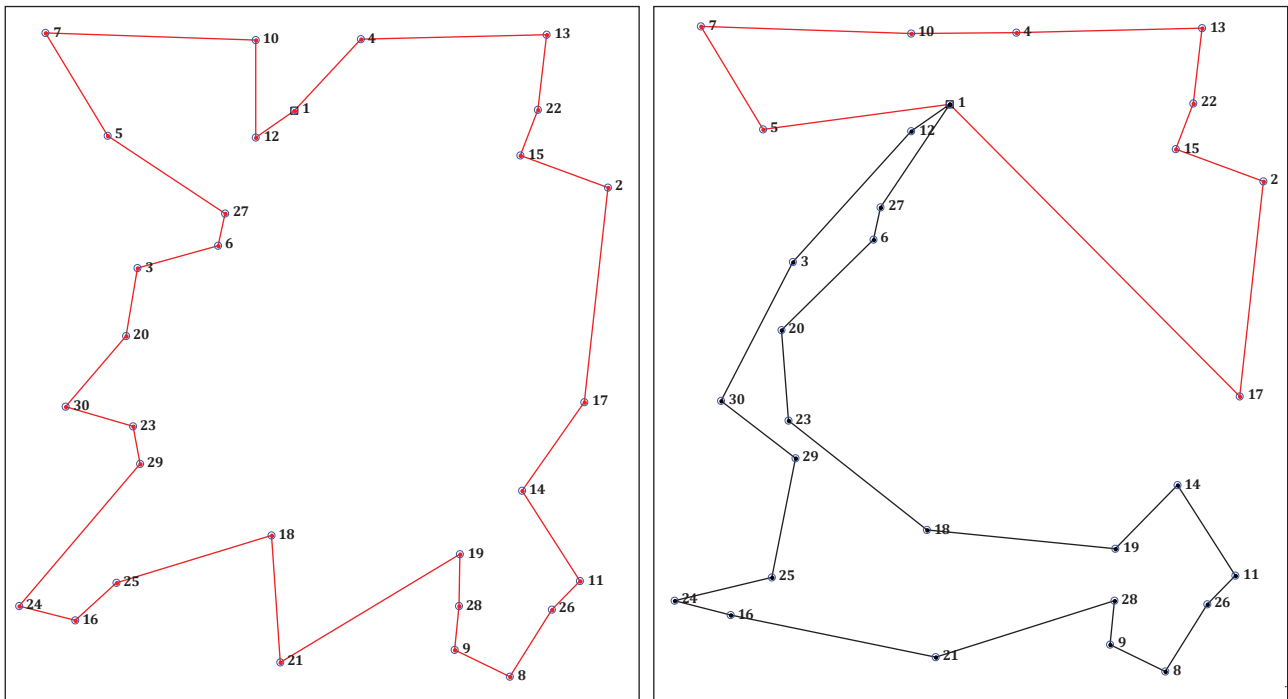


Figure 2 The best (on the left, the route length is 482.0141 covered by a single feeder) and worst (on the right, the route length is 638.5203 covered by two feeders whose tour lengths are 266.6763 and 371.844) solutions (out of 100 solutions) of an instance with 30 ports

Source: Authors

ports (in a rerun out of 100 reruns) and an instance with 50 ports (in a rerun out of 100 reruns) have been solved the worst (out of 200 instances each) by the worst convergence speed. The best route has been obtained the fastest in 11 instances: 2 ones with 10 ports, 3 ones with 15 ports,

and 6 instances with 30 ports. The pattern “the best route converges the longest” is slightly more likely than the previous two patterns: 5 instances with 20 ports, 16 instances with 50 ports, 2 instances with 60 ports, and 8 instances with 70 ports have had the shortest routes obtained in the

maximal number of iterations. Finally, the pattern “the worst route converges the fastest” is significantly likely than the previous one: there are 8, 12, 11, 10, 9, 8, 12, 13 instances (4% to 6.5% of 200 instances) with 15 to 80 ports whose longest routes have been obtained in the minimal number of iterations.

8 Practical applicability and significance

Our approach justified above consists in solving the same route length minimization problem for a definite number of times and then selecting the solution whose route length is minimal. The number of algorithm reruns can be taken from Table 3 as the average number ceiling depending on the desired accuracy. Our recommendation for the accuracy is factor $\mu \in [0; 3]$. The solving can be either parallelized or fulfilled sequentially. Therefore, it is an easy-applicable procedure.

This approach is very significant for maritime cargo delivery. In addition to the examples in Figures 1 and 2 showing the potential loss if the algorithm is not re-run, Figure 3 presents another staggering example for an instance with 80 ports. The best route, whose length is 757.2124, is covered by two feeders whose tour lengths are 266.6763 and 371.844. The worst route, whose length is 995.1403, is covered by three feeders whose tour lengths are 373.6182, 371.7376, and 249.7845. So, one of those three feeders will not cover the minimal tour length (which is 250). The best route, that obviously can be found only by re-running the algorithm, thus spares here a feed-

er and reduces expenses for maritime cargo delivery by 31.4215%. This is a tremendously significant saving. It is worth noting that the best route has been found in 2687 iterations, whereas the worst one has taken 2179 iterations. The patterns “the best route converges the longest” and “the worst route converges the fastest” are true here.

Some variation may exist inside the algorithm itself. Its inner parameters are adjustable. In expunging tours with conditions (11) or (12), much higher penalties might have been used. For instance, it is

$$d_m^{(obs)} = d_m, d_m = d_m^{(obs)} + 100 \cdot (d_m^{(obs)} - d_{max}) = 101d_m^{(obs)} - 100d_{max} \tag{57}$$

instead of (25), and it is

$$d_m^{(obs)} = d_m, d_m = d_m^{(obs)} + 100 \cdot (d_{min} - d_m^{(obs)}) = 100d_{min} - 99d_m^{(obs)} \tag{58}$$

instead of (26). However, the higher penalty does not imply the better selectivity. The respective counterexample is presented in Figure 4, where the port position components are generated as

$$p_{k1} = 50 \cdot \theta_1 \text{ and } p_{k2} = 50 \cdot \theta_2, k = \overline{1, 15}.$$

Even this almost trivial route length minimization problem clearly confirms that re-running the genetic algorithm, whichever its inner and input parameters are, is necessary

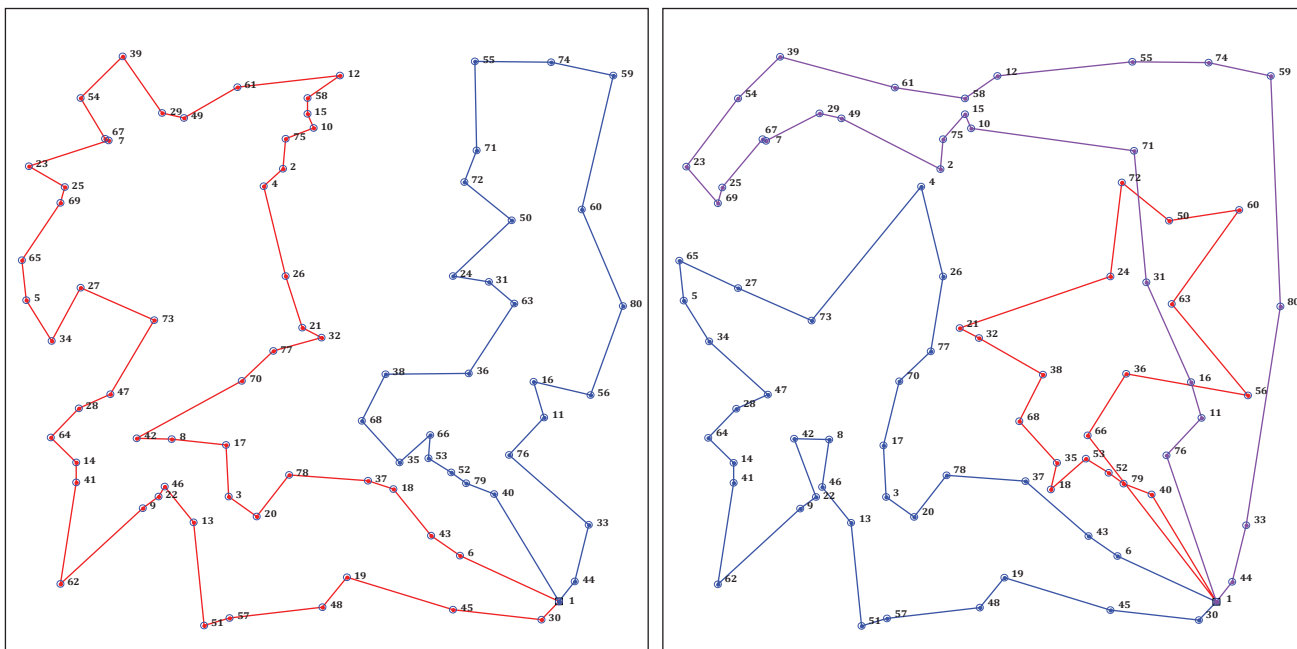


Figure 3 The best route (left) covered by two feeders and the worst route (right) covered by three feeders, where one of the three tours violates the lower limit constraint (11)

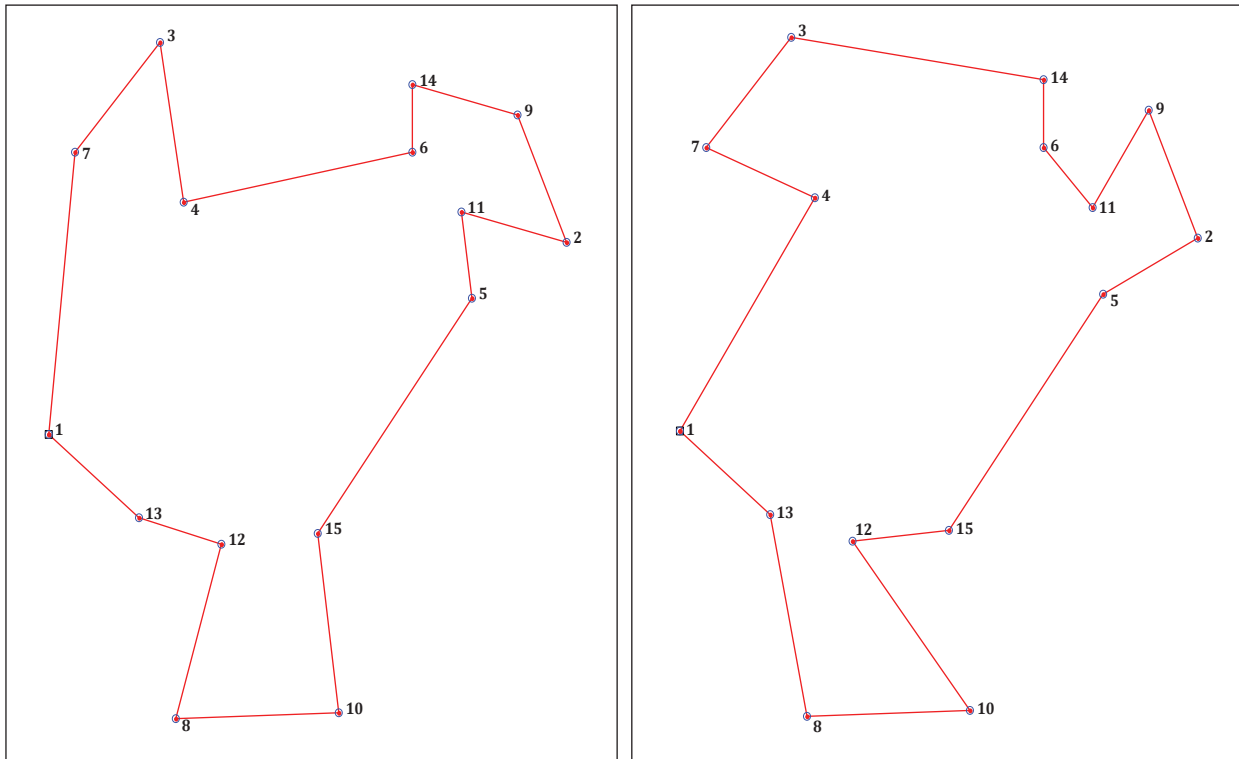


Figure 4 The route (left) of length $\tilde{\rho}_{\Sigma}^{**}(15; s, 1) = 183.4649$ returned by penalties in (25), (26), and the route (right) of length $\tilde{\rho}_{\Sigma}^{**}(15; s, 2) = 187.5093$ returned by penalties in (57), (58)

Source: Authors

for assuredly obtaining the route length practically close to the minimal length of the delivery route.

The algorithm does converge but the convergence result is not a single route or routes of the same length. Some variation of the convergence result exists. It depends on the stream of pseudorandom numbers. Re-running the algorithm reveals the core of the variation by trying a series of different streams.

It is necessary to remind that we have tested the pseudorandom number generator influence by studying nearly the worst-case scenario. In our scenario, the feeder tour length is tightly constrained from 250 to 500. In other scenarios, where the tour length constraints are looser, the algorithm performance is instable also and requires re-running to obtain an acceptable solution. Despite our model of delivering cargo is very simple, the constraints make it practically applicable.

9 Conclusion

We have ascertained that the pseudorandom number generator dramatically influences the genetic algorithm performance. Both the algorithm accuracy and its convergence speed are influenced deeply depending on the stream of pseudorandom numbers used for randomly generating the starting population and accomplishing random mutations. As the number of ports increases from 10 to

80, the route length variation intensifies from 3.5% to 22.5% on average. The algorithm accuracy is increased by re-running the algorithm to solve the same problem until closely the best solution is obtained. The number of reruns is taken from Table 3 as the average number ceiling. As the number of ports increases, the number of reruns should be taken greater depending on the desired accuracy. For instance, to definitely increase the genetic algorithm performance in solving route length minimization problems with 40 ports and more, the algorithm should be re-run for no less than 51 times, whereupon the solution whose route length is minimal is selected (out of those 51 solutions or more). Expenses for maritime cargo delivery are thus dramatically reduced. Therefore, our approach is a substantial scientific and valuable practical contribution to the field of genetic algorithms used, in particular, to optimize maritime cargo delivery.

In our set-up of testing the pseudorandom number generator influence, we generated port position components by the standard normal distribution. This made the set of ports look like it is a center-based cluster rather than, e. g., a crescent or curvilinear coast stretched in one direction. A way of possible extension of our research is to consider other models of generating port position components and study whether the pseudorandom number generator influence remains the same. Although we suppose that the influence does not depend on the pattern of how ports are scattered

with respect to the hub, such a study would be a generalization of the presented study. The latter may have a spin-off research of selecting the best pseudorandom number generator type: along with the Mersenne Twister generator we used in the presented study, other generators may be used [24, 25, 26] differing in the pseudoindependence level of their stream numbers [22, 23].

Funding: The research presented in the manuscript did not receive any external funding.

Authors Contribution: Conceptualization – V. V. Romanuke, A. Y. Romanov, M. O. Malaksiano; **methodology** – V. V. Romanuke, A. Y. Romanov, M. O. Malaksiano; **data collection and processing** – V. V. Romanuke; **writing** – V. V. Romanuke, A. Y. Romanov; **review and editing** – V. V. Romanuke, A. Y. Romanov, M. O. Malaksiano; **supervision** – V. V. Romanuke, M. O. Malaksiano.

References

- [1] "Container shipping – statistics & facts." [Online]. Available: <https://www.statista.com/topics/1367/container-shipment>. [Accessed: July 25, 2022].
- [2] Review of Maritime Transport 2021, *United Nations Conference on Trade and Development*. Geneva: United Nations, 2021. [Online]. Available: https://unctad.org/system/files/official-document/rmt2021summary_en.pdf. [Accessed: July 25, 2022].
- [3] T. R. Walker, O. Adebambo, M. C. Del Aguila Feijoo, E. Elhaimer, T. Hossain, S. J. Edwards, C. E. Morrison, J. Romo, N. Sharma, S. Taylor, and S. Zomorodi, "Chapter 27 – Environmental Effects of Marine Transportation", in *World Seas: an Environmental Evaluation*. Cambridge, Massachusetts, USA: Academic Press, 2019, pp. 505 – 530. DOI: <https://doi.org/10.1016/B978-0-12-805052-1.00030-9>.
- [4] "Air Freight: a market study with implications for landlocked countries." [Online]. Available: <https://www.worldbank.org/en/topic/transport/publication/air-freight-study>. [Accessed: July 27, 2022].
- [5] "Advantages of Maritime transport." [Online]. Available: <https://blueoceanmag.com/advantages-of-maritime-shipment>. [Accessed: July 27, 2022].
- [6] "Sea vs road vs air freight – which one is right for you?" [Online]. Available: <https://blog.intekfreight-logistics.com/air-freight-vs-ocean-freight-carbon-footprint-environmental-impact>. [Accessed: July 28, 2022].
- [7] "Air freight vs ocean freight carbon footprint & environmental impact." [Online]. Available: <https://gwtimepex.co.uk/sea-vs-road-vs-air-freight>. [Accessed: July 28, 2022].
- [8] C. Archetti, L. Peirano, and M. G. Speranza, "Optimization in multimodal freight transportation problems: A Survey", *European Journal of Operational Research*, vol. 299, iss. 1, 2022, pp. 1 – 20. DOI: <https://doi.org/10.1016/j.ejor.2021.07.031>.
- [9] P. A. Miranda, C. A. Blazquez, C. Obreque, J. Maturana-Ross, and G. Gutierrez-Jarpa, "The bi-objective insular traveling salesman problem with maritime and ground transportation costs", *European Journal of Operational Research*, vol. 271, iss. 3, 2018, pp. 1014–1036. DOI: <https://doi.org/10.1016/j.ejor.2018.05.009>.
- [10] D.-Z. Du and P. M. Pardalos, *Handbook of Combinatorial Optimization*. New York, NY, USA: Springer, 1998, 2406 p. DOI: <https://doi.org/10.1007/978-1-4613-0303-9>.
- [11] B. Golden, L. Bodin, T. Doyle, and W. Stewart, Jr., "Approximate traveling salesman algorithms", *Operations Research*, vol. 28, no. 3, 1980, pp. 633–846. DOI: <https://doi.org/10.1287/opre.28.3.694>.
- [12] "An evaluation of the traveling salesman problem." [Online]. Available: <https://scholarworks.calstate.edu/downloads/xg94hr81q>. [Accessed: July 29, 2022].
- [13] A. Hertz and M. Widmer, "Guidelines for the use of metaheuristics in combinatorial optimization", *European Journal of Operational Research*, vol. 151, iss. 2, 2003, pp. 247–252. DOI: [https://doi.org/10.1016/S0377-2217\(02\)00823-8](https://doi.org/10.1016/S0377-2217(02)00823-8).
- [14] A. Colomi, M. Dorigo, F. Maffioli, V. Maniezzo, G. Righini, and M. Trubian, "Heuristics from nature for hard combinatorial optimization problems", *International Transactions in Operational Research*, vol. 3, iss. 1, 1996, pp. 1–21. DOI: [https://doi.org/10.1016/0969-6016\(96\)00004-4](https://doi.org/10.1016/0969-6016(96)00004-4).
- [15] "Time complexity analysis of the genetic algorithm clustering method." [Online]. Available: https://www.researchgate.net/publication/262403214_Time_complexity_analysis_of_the_genetic_algorithm_clustering_method. [Accessed: July 29, 2022].
- [16] L. D. Chambers, *The Practical Handbook of Genetic Algorithms*. Chapman and Hall/CRC, 2000, 544 p.
- [17] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*. John Wiley & Sons, 2003, 253 p. DOI: <https://doi.org/10.1002/0471671746>.
- [18] A. Király and J. Abonyi, "Redesign of the supply of mobile mechanics based on a novel genetic optimization algorithm using Google Maps API", *Engineering Applications of Artificial Intelligence*, vol. 38, 2015, pp. 122–130. DOI: <https://doi.org/10.1016/j.engappai.2014.10.015>.
- [19] Goldberg, D. E. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading: Addison-Wesley, 1989, pp. 10–18.
- [20] "How random generator quality impacts genetic algorithm performance." [Online]. Available: <https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.153.1330&rep=rep1&type=pdf>. [Accessed: July 29, 2022].
- [21] "On random numbers and the performance of genetic algorithms." [Online]. Available: https://www.researchgate.net/publication/2491570_On_Random_Numbers_and_the_Performance_of_Genetic_Algorithms. [Accessed: July 30, 2022].
- [22] R. Kneusel, *Random Numbers and Computers*. Springer International Publishing, 2018, 260 p. DOI: <https://doi.org/10.1007/978-3-319-77697-2>.
- [23] M. Matsumoto and T. Nishimura, "Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator", *ACM Transactions on Modeling and Computer Simulation*, vol. 8, iss. 1, 1998, pp. 3–30. DOI: <https://doi.org/10.1145/272991.272995>.
- [24] D. Blackman and S. Vigna, "Scrambled Linear Pseudorandom Generators", 2018. arXiv:1805.01407 [cs.DS]
- [25] S. Harase and T. Kimoto, "Implementing 64-bit Maximally Equidistributed F2-Linear Generators with Mersenne Prime Period", *ACM Transactions on Mathematical Software*, vol. 44, iss. 3, 2018, Article No. 30, pp. 1–30. DOI: <https://doi.org/10.1145/3159444>.
- [26] B. Widynski, "Squares: A Fast Counter-Based RNG", 2020. arXiv:2004.06278 [cs.DS].