Appl Intell (2012) 37:155–174
DOI 10.1007/s10489-011-0319-7

ORIGINAL PAPER

On evolutionary computing in multi-ship trajectory planning

Rafal Szlapczynski · Joanna Szlapczynska

Published online: 24 September 2011

Abstract The paper presents the updated version of Evolutionary Sets of Safe Ship Trajectories: a method which applies evolutionary algorithms and some of the assumptions of game theory to solving ship encounter situations. For given positions and motion parameters of the ships, the method finds a near optimal set of safe trajectories of all ships involved in an encounter. The method works in real time and the solutions must be returned within one minute, which enforces speeding up the optimization process. During the development of the method we have tested extensively various formulas for fitness function, problem-dedicated specialized operators as well as methods of selection. In the course of this research it turned out that some of the classic evolutionary mechanisms had to be modified for better performance, which included the order of some operations. The results of the adaptation process are presented here. The paper includes explicit description of all evolutionary mechanisms used and accentuates the research on improving the optimization process by adjusting evolutionary mechanisms to the problem.

Keywords Evolutionary algorithms · Ship collision avoidance · Decision support systems

R. Szlapczynski (✉)
Gdansk University of Technology, Narutowicza 11/12, Gdańsk, Poland
e-mail: rafal@pg.gda.pl

J. Szlapczynska
Gdynia Maritime University, Morska 81-87, Gdynia, Poland
e-mail: asiasz@am.gdynia.pl

1 Introduction

There are a number of methods of solving multi-ship encounter situations: they can be divided into deterministic and heuristic ones. Deterministic approach is based on differential games and has been proposed by Lisowski [16]. Its main limitation is high computational time for more complex scenarios of encounters. Of the heuristic ones the most successful and flexible is searching for a ship's trajectory by genetic or evolutionary algorithms. The method has been first proposed by Smierzchalski and Michalewicz [20] and since then similar approaches has been tried by other researchers: evolutionary computation (EC) may be applied for finding an optimal path [26, 31] and genetic algorithms (GA) are used for optimization of collision avoidance maneuvers [13, 28]. Other related approaches include trajectory optimization using genetic annealing algorithm [2] and ship collision avoidance route planning by ant colony algorithm [27]. Apart from these, automatic collision avoidance of ships using artificial potential field and speed vector [29] is also used, which is an adaptation of the Potential Field Method (PFM) widely used for navigating mobile robots [19]. Summaries of applying GA and EC to maritime collision avoidance and trajectory planning have been presented by Yang et al. [30] and Statheros et al. [21] among others.

In short, EC and GA approach to the problem use algorithms, which for a given set of pre-determined input trajectories find a solution that is optimal according to a given fitness function. However, their limitation is that they assume that targets' motion parameters do not change and if they do change, the own trajectory (i.e., the trajectory of the own ship) has to be recomputed. This limitation becomes a serious one on restricted waters. If a target's current course collides with a landmass or another target of a higher priority, there is no reason to assume that the target would

keep such a disastrous course until the crash occurs. Consequently, planning the own trajectory for the unchanged course of a target will be futile in the majority of such cases. Also, existing EC methods do not offer a full support to Vessel Traffic Service (VTS) operators, who might face the task of synchronizing trajectories of multiple ships while many of those ships are maneuvering.

Therefore, we propose a new approach, where, instead of finding the optimal own ship's trajectory for the unchanged courses and speeds of targets, a search is made for an optimal set of safe trajectories of all ships involved in an encounter. Our method is called *Evolutionary Sets of Safe Ship Trajectories* (ESoSST) and its earlier version has been presented in [25]. Here, it must be noted that optimizing a set of trajectories instead of a single trajectory drastically magnifies the search space. At the same time, working in the constrained maritime environment while trying to obey the COLREGS produces multiple constraints, which make it more difficult to find any acceptable solution. All these factors contribute to a non-typical optimization problem, often unsolvable by pure genetic algorithms in a given time that is strictly limited because of operating while the ships are approaching each other or the land. That is why the first version of the ESoSST method [25] was aimed at meeting all the critical requirements: it offered basic functionality of solving the defined problem and simplified supporting of international collision avoidance rules (aka COLREGS) [3, 5]. The returned results were usually sub-optimal and therefore having accomplished the major goals we focused our research on improving all the phases of the evolutionary process, which is addressed by this paper. First, the fitness function has been changed: the optimization criterion and existing penalties were modified and additional COLREGS-violation penalties were introduced. Then, most of the evolutionary mechanisms were extended or replaced with more advanced ones to improve the ESoSST method performance. New crossover operators have been designed, various selection algorithms have been tried and some of the previously used specialized operators have been slightly changed as well. Here we present a description and a discussion of the choices and modifications made in all the phases of the evolutionary cycle, as well as results of the simulation experiments that have been carried out to make these decisions.

The rest of the paper is organized as follows. In the next section the proposed approach to solving multi-ship encounters is compared with the own ship evolutionary approach. In Sect. 3 the task—finding sets of safe trajectories—is presented as an optimization problem. Then the fitness is formulated in Sect. 4. This is followed in Sect. 5 by a detailed description of the subsequent phases of the evolutionary cycle: initial population generation, reproduction, specialized operators, mutation and selection. In the same section also the modified evolutionary cycle is introduced. Section 6 includes visualization of an exemplary result returned by the ESoSST. Simulation experiments and discussion of their results is presented in Sect. 7. Finally we summarize our algorithm and conclude in Sect. 8.

2 Comparing two different evolutionary approaches

As stated in the introduction, our approach is to search for an optimal set of trajectories instead of searching for an optimal trajectory of the own ship. This approach is typical for collision avoidance methods based on games theory, but has not been tried before for evolutionary computing applied to marine navigation. Therefore, before we present the details of our ESoSST method, the practical difference of the two evolutionary approaches will be shown in this section. A question that is often asked when discussing a collision avoidance system is: what would happen if all of the ships involved in an encounter situation were using it? Below we present a scenario of an encounter of two ships in a narrow channel (the dotted areas surrounding the landmass form the safety isobath). The ship parameters are provided by Table 1.

The solution returned by the ESoSST method based on our approach is presented in Fig. 1. Because the method searches for an optimal set of trajectories, the results returned by the onboard systems of both ships would be the same for the same input data and settings or would differ slightly in case of minor differences in the input data of both ships. However, the tendency of their movement would be the same: both ships would perform maneuvers to their starboards (as recommended by COLREGS) and pass each other safely.

In case of the standard evolutionary approach, a system would assume that course of the other ship would not change. As a result, each ship would see the situation in a different way. Ship 1, assuming that Ship 2 keeps its course would maneuver to starboard, as shown in Fig. 2. However Ship 2, assuming that Ship 1 keeps its course, would keep close to its port side (Fig. 3) of the channel so as to pass Ship 1 starboard to starboard. Even though COLREGS generally recommend maneuvering to starboard in case of head on and crossing encounters, Ship 2 would not do it because it would see no threat of collision with Ship 1.

Table 1 Ship parameters for the presented scenario

	Initial position	Destination position	Speed [knots]
Ship 1	16° 20′ 31″ E 56° 35′ 03″ N	16° 32′ 17″ E 56° 50′ 09″ N	10.00
Ship 2	16° 34′ 14″ E 56° 49′ 36″ N	16° 16′ 36″ E 56° 34′ 31″ N	10.00

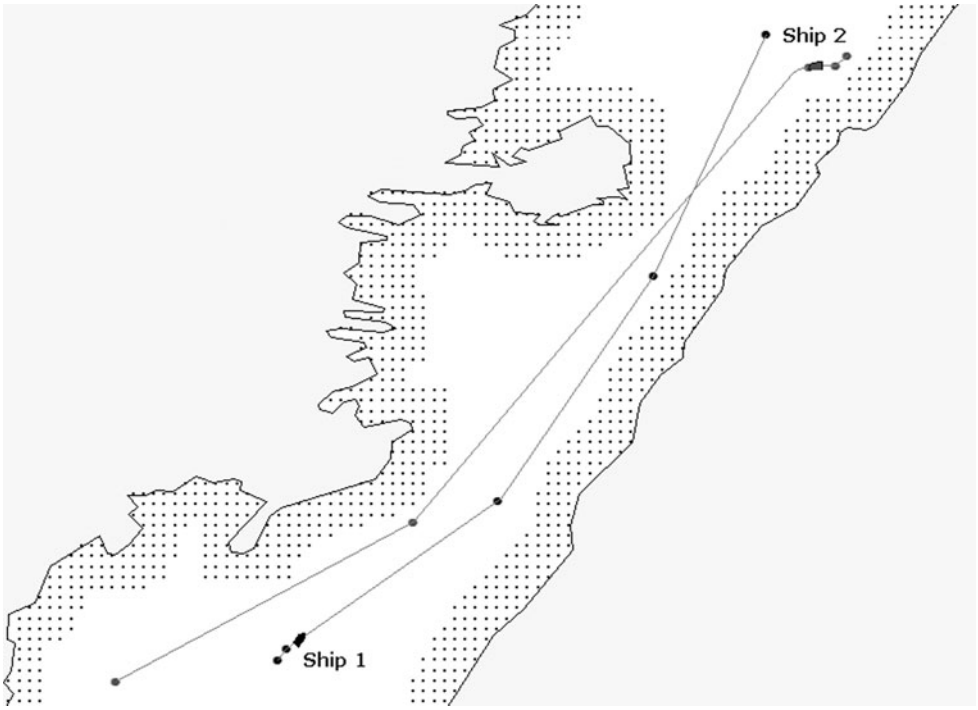

Fig. 1 The solution (a set of two trajectories) returned by the ESoSST method based on the proposed approach

Fig. 2 A trajectory planned for Ship 1 by the typical evolutionary collision-avoidance method

In result both ships would initially choose trajectories shown in Fig. 4, which they would later have to change, after detecting the maneuvers of the other ship. As has been illustrated in Fig. 4, the approach based on the optimization of a single trajectory fails to deliver a reasonable solution for some situations, even when the future maneuver of the other ship is obvious.

3 Optimization problem and its implications

We assume that the following data is given:

– stationary constraints (such as landmasses and other obstacles),
– positions, courses and speeds of all ships involved,

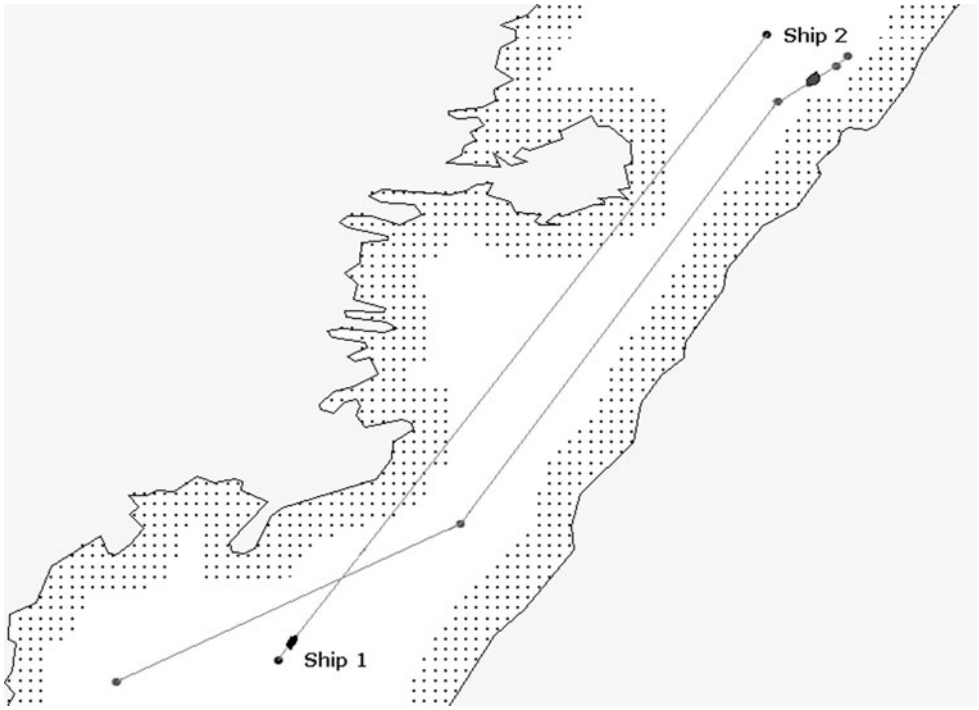

Fig. 3 A trajectory planned for Ship 2 by the typical evolutionary collision-avoidance method

Fig. 4 Potentially dangerous trajectories planned for the two ships by their systems with a single trajectory optimization method applied

– ship domains (a domain is an area around a ship that should be free from other ships, obstacles, etc.) and
– times necessary for accepting and executing the proposed maneuvers.

Ship positions and ship motion parameters (courses and speeds) are provided by ARPA (Automatic Radar Plotting Aid) and AIS (Automatic Identification System) systems. A ship domain can be determined based on the ship's length, its motion parameters and the type of water region. Since the shape of a domain is dependent on the water region, we have decided to use a ship domain model by Davis [6], which updated Goodwin's model [12], for open waters and to use a ship domain model by Coldwell [4], which updated Fuji's model [11], for restricted waters. As for the last parameter—

the necessary time, it is computed on the basis of navigational decision time and ship's maneuvering abilities. It must be noted that it is the navigational decision time (usually 3 to 6 minutes) that has the major impact on the total time between recommending a trajectory by the system and executing the first of the proposed maneuvers by the navigator. Therefore a 6-minute value of the necessary time has been assumed by default.

Knowing all the abovementioned parameters the goal is to find a set of trajectories that minimize the average way loss caused by maneuvering while fulfilling the following conditions:

- none of the stationary constraints (i.e. landmasses and safety isobaths) and ship domains are violated,
- course alteration should normally be between 15 and 60 degrees,
- speed alteration are not to be applied unless necessary (that is, when collision cannot be avoided by course alteration up to 60 degrees),
- a ship only maneuvers when she is obliged to and maneuvers to starboard are favored over maneuvers to port,
- a succession of small alterations of course and/or speed should be avoided (by default a new course has to be kept for at least 3 minutes).

The conditions are mostly either imposed by COLREGS [3, 5] and good marine practice or by the economics. In particular, course alterations are favored over speed changes during collision avoidance maneuvers (COLREGS, Rule 8 c) and speed reduction should only be applied when necessary (COLREGS, Rule 8 e).

Course changes smaller than 15 degrees might be misleading for the ARPA systems (problems with detection) and the course alterations larger than 60 degrees are inefficient. Also, ships should only maneuver when necessary, since each maneuver of a ship makes it harder to track its motion parameters for the other ships' ARPA systems. An additional computational constraint is that because of the optimization being done in real time (with the ships approaching each other and the obstacles), the solution should be returned within a short time specified by the operator of the system (by default, one minute is assumed). The following subsections provide details on detecting the constraint violations and its consequences.

3.1 Detecting static constraint violations (collisions with landmasses and safety isobath)

The ESoSST method uses a vector map of a given area. We have decided not to process a vector map for constraint violations detection, but to use it for generating a bitmap of an area. Although it is a time-intensive operation, fortunately, bitmaps can be generated offline and only once for each area. Then, when the method is running in real time, each bitmap cell which the trajectory of a ship traverses is read and checked for belonging to a landmass or a safety isobath. If a cell belongs to landmass or a safety isobath, a constraint violation is registered in the trajectory data structure. The information on constraint violation includes the percentage of a particular trajectory's segment, which crosses impassable cells. For a bitmap, whose detail level depends on the given vector map, the computational time of this algorithm would be proportional to the number of traversed cells. This approach is also flexible in terms of bathymetry checks: for a cell containing information on the water depth, it is easy to check whether it is passable or not for a particular ship. The computational time of detecting static constraint violations for a given scenario is proportional to the number of ships.

3.2 Detecting ship domain violations

The algorithm for detecting ship-to-ship collisions is as follows. Each ship's trajectory is checked against all other ships. For each pair of ships the start time and end time of each trajectory's segments are computed. If two segments of the two trajectories overlap in time they are checked for geometrical crossing. In case of a crossing the special collision risk measure—approach factor value [22] is computed. Then, if the approach factor value indicates collision, the type of an encounter (head-on, crossing or overtaking) is determined on the basis of the ships' courses and it is decided which ship is to give way (both ships in case of head-on). The collision is only registered for the give way ship and the information on the collision are stored in the trajectory data structure. The computational time of detecting ship domain violations for a given scenario is proportional to the number of potential ship-to-ship collisions and thus grows squarely with the number of ships.

3.3 Detecting COLREGS violations

The three most common types of COLREGS violations are as follows:

- a ship does not give way when it should,
- a ship gives way when it should not (making unexpected and misleading maneuvers),
- a ship maneuvers to port-board when it should maneuver to starboard.

Each of these three situations may happen on either open or restricted waters, which gives us a total of six cases to handle. The difficulty with deciding whether a ship has acted lawfully or not, lies in the nature of evolutionary algorithms as well as in the nature of the problem itself: COLREGS specify only the procedures for ship-to-ship encounters. When looking at a set of ship trajectories for a multi-target encounter, it is sometimes impossible to tell, what the

reason for a particular maneuver was: which ship was given way intentionally and which one benefited from it only as a side effect. Therefore the final COLREGS violations detection rules applied in the method are:

1. On open waters:
   a. if a ship is not obliged to give way to any other ship any maneuver it performs is registered as COLREGS violation,
   b. if a ship is obliged to give way and does not perform a maneuver it is registered as COLREGS violation,
   c. all maneuvers to port board are registered as COLREGS violations.
2. On restricted waters: here, as explained before, every trajectory node which is a part of a maneuver contains information on the reason why this particular node has been inserted or shifted: land or other stationary obstacle avoidance, target avoidance or accidental maneuver generated by evolutionary mechanisms. Based on this COLREGS violations are registered as follows:
   a. if a ship does not initially have to give way to any target and its first maneuver has reason other than static constraint violation avoidance it is registered as COLREGS violation,
   b. any maneuver to port board of reason other than static constraint violation avoidance is registered as COLREGS violation.

The computational time of detecting COLREGS violations for a given scenario is proportional to the number of ships.

3.4 High cost of evaluation and other consequences

Due to the facts presented in the sub-sections above the evaluation (which includes constraint violations detection) is the most time consuming phase of the evolutionary algorithm with the computational times of other stages being insignificant in comparison. Combined with the fact that the evolutionary process is executed in real time it seriously limits the number of generations we can afford. For the most complex scenarios including several ships of various dynamics, complex ship domain models and restricted waters with multiple obstacles only up to 200 generations can be processed for a population of 100 members, even if more generations could bring further rise in fitness function values. Thus, the obvious conclusion is that it is necessary to achieve as much progress as possible in each generation, which can be done by investing more computational time in other stages of the evolutionary process to make them more effective. Additionally, it means that balancing between two desirable goals: search intensity and search diversity [15] is especially difficult in our case.

The other practical implications of the problem are as follows. The constraints are hard to be met and vast majority of the individuals in early generations will be unacceptable, with the valid and safe sets of trajectories few and far between. What more, in many cases the offspring of nearly perfect parents will be unacceptable too. Also, even minor mutations can often bring disastrous effects completely spoiling previously high-valued individuals. All this, combined with strict time limits (by default—one minute) leads to the need for optimizing the evolutionary process. In the two following sections first the fitness function is formulated (Sect. 4), then the elements of the evolutionary cycle are described with focus on the evolutionary process optimization (Sect. 5).

4 Formulating fitness function

In EC all individuals (sets of trajectories) are evaluated by the specially designed fitness function, which should reflect optimization criteria and constraints [17]. In this section it is shown how this normalized fitness function is formulated. First the basic economy criterion is presented, then penalties for constraint violations. Penalizing constraint violations is a commonly used technique, but usually penalties are additive elements, either static or dynamic. In our case they are factors: this makes it easier to normalize the fitness function. It also means that the pressure on infeasible solutions automatically grows with the general growth of the fitness function values. This tendency is similar to dynamic or annealing penalties [18], where pressure on infeasible solutions is increased towards the end of the process (for later generations). It also must be noted that keeping the high resolution of penalties is crucial here. In [18] it is stated that "usually the penalty function is based on the distance of a solution from a feasible region or on the effort to repair the solution" and it is reasonable to apply this approach here. If the initial population consists of unacceptable individuals, we have to differ between them: assign higher fitness function values to those which are "promising" (and may be subjects to evolution) and lower to those that should simply be eliminated. For example, a trajectory crossing a landmass on 1% of its length shows promise (this crossing can possibly be eliminated by a specialized operator), but the one which crosses landmass on 50% of its length is probably useless and should be penalized much more severely. Also the collisions with ships are penalized less severely than those with land because they are less "certain". A collision with landmass is always valid, while collisions with other ships may be eliminated as a side effect of the future changes of those other ships' trajectories or future changes of the own trajectory due to avoiding collision with landmass.

4.1 Basic criterion—minimizing way loss

The basic criterion is the economic one—minimizing way losses of trajectories in a set. For each of the trajectories a trajectory economy factor tef_i is computed according to the formula (1).

$$tef_i = \left(\frac{l_i - \Delta l_{wi}}{l_i}\right), \tag{1}$$

where i: the index of the current ship, l_i: the total length of the i-th ship's trajectory [nautical miles], Δl_{wi}: the total way loss of the i-th ship's trajectory [nautical miles] computed as a difference between the trajectory length (l_i) and length of a segment joining trajectory's start point and endpoint.

As can be seen, the trajectory economy factor *tef* is always a number from a (0, 1] range.

4.2 Penalizing static constraint violation

After the trajectory economy factor has been computed, the static constraints are handled by introducing penalties for violating them. For each trajectory its static constraint factor scf_i is computed. The static constraints are always valid and their violations must be avoided at all cost, therefore penalties applied here are the most severe—hence the square in the formula (2).

$$scf_i = \left(\frac{l_i - l_{ci}}{l_i}\right)^2, \tag{2}$$

where l_{ci}: the total length of the parts of the i-th ship's trajectory, which violate stationary constraints [nautical miles].

The static constraint factor is a number from a [0, 1] range, where "1" value means no static constraint violation (no landmasses or other obstacles are crossed) and "0" value is for trajectories crossing landmasses on their whole length.

4.3 Penalizing collisions with other ships

Analogically to the static constraint factor, collision avoidance factor caf_i is computed to reflect the ship's collisions with all other privileged ships as shown by (3).

$$caf_i = \prod_{j=1, j\neq i}^{n} (\min(fmin_{i,j}, 1)), \tag{3}$$

where n: the number of ships, j: the index of a target ship, $fmin_{i,j}$: the approach factor value [22] for an encounter of ships i and j, if i-th ship is the privileged one, the potential collision is ignored and the approach factor value is equal to "1" by definition.

The collision avoidance factor is a number from a [0, 1] range, where "1" value means no ship domain violation and "0" means a crash with at least one of the targets.

4.4 Penalizing COLREGS violations

The COLREGS violations are secondary to static constraint violations and to collisions with other ships and therefore we have decided to penalize it moderately, to make sure that constraints from the previous two points are met first. COLREGS compliance factor ccf_i is computed according to the following formula (4).

$$ccf_i = 1 - \sum_{k=1}^{m} [p_k], \tag{4}$$

where m: the number of COLREGS violations registered for the current ship as described in Sect. 3.3, k: the index of a registered violation, p_k: penalty for the k-th of the registered COLREGS violation.

The penalty values for all registered COLREGS violations are configurable in the method and are set to 0.05 by default.

4.5 Fitness function value

Once all aforementioned factors have been computed, the fitness function value is calculated. We wanted the fitness function to be normalized, for convenience of further evolutionary operations, mostly for selection. When fitness function values are normalized, we do not need any additional operations on them and they can directly be used for random proportional and modified random proportional selection in the reproduction and succession phases of the evolutionary algorithm. We can also easily measure and see progress we make with each generation. However, normalized fitness function is harder to obtain, because we have to make sure that we keep the high resolution of evaluating the individuals, namely that we differ between various levels of penalties: stationary constraints, being more important than collision avoidance and collision avoidance being more important than COLREGS compliance.

Here, we succeeded in formulating a normalized fitness function, while keeping relatively high resolution of evaluation: minor stationary constraint violations are penalized similarly as major collisions with other ships and minor collisions with other ships are penalized similarly as multiple COLREGS violations. The final fitness function is as follows:

$$fitness = \sum_{i=1}^{n} \frac{fitness_{tr\,i}}{n}, \tag{5}$$

where:

$$fitness_{tr\,i} = tef_i \cdot scf_i \cdot caf_i \cdot ccf_i. \tag{6}$$

It must be noted here, that while fitness function values are normalized, a single trajectory fitness function ($fitness_{tr\,i}$)

value may be equal to 1.0 only for a stand-on ship in lack of obstacles on his way. The global fitness function value (*fitness*) of 1.0 is only possible when none of the ships maneuvers, that is when there are no encounters (situations, which are of no interest). The minimal assumed course alteration maneuver is 15 degrees, the minimal time for accepting and executing a maneuver—6 minutes. Thus, for a ship which was supposed to cover a distance of 12 nautical miles with a speed of 12 knots, but performed one minimal course alteration maneuver and kept the changed course for 3 minutes, the trajectory's fitness function value would be approximately 0.99. For more complex scenarios the maximum possible value of fitness function computed over all trajectories would be even smaller. Therefore, while the precise value cannot be determined analytically, it is reasonable to assume that for a randomly generated multi-ship encounter situation the maximum possible value of fitness function would be below 0.98, which can be considered a better practical reference value than 1.0.

5 Evolutionary algorithm

This section describes subsequent phases of the evolutionary cycle: generation of the initial population, reproduction, specialized operators, mutation and selection. It also introduces the modified evolutionary cycle. Some of the changes and choices that we have made were our inventions, dictated by the specifics of the problem, while others were based on reported strategies and techniques [18]. The former are collision avoidance operators, which are dedicated to particular encounter situations and are using the data on the collision type, degree and its time and place. The latter include non-uniform mutation and arithmetical crossover of nodes, adjusted to our problem. In general, as opposed to typical EC or GA, we use a hybrid approach of EC and operators which are either semi-deterministic or strongly based on problem-specific data [14]. In our case a solution is a set of trajectories, which are evaluated separately (though not independently) and we can benefit from this fact. For example, we use trajectory fitness values instead of generation number (a typical GA parameter) in case of mutation: we introduce a trajectory mutation probability—a probability of mutating a part of a solution—which depends on trajectory fitness value. The details on the particular evolutionary operations are given in the following subsections.

5.1 Generating the initial population: randomly generated trajectories or strong pre-processing?

The main question, regarding this phase of the evolutionary algorithm is as follows: is it better to invest computational time in strong pre-processing to gain strong initial population or rather opt for randomly generated initial population to save on computational time?

As has been said in the introduction, each individual (a population member) is a set of trajectories, each trajectory corresponding to one of the ships involved in an encounter. A trajectory is a sequence of nodes, each node containing the following data:

- geographical coordinates x and y,
- the speed between the current and the next node.

Typically, the initial population is generated randomly or by some very generic methods. We tried strong pre-processing approach first however, where the initial population contained three types of individuals:

- a set of original ship trajectories—segments joining the start and destination points,
- sets of safe trajectories determined by other methods,
- randomly modified versions of the first two types—sets of trajectories with additional nodes, or with some nodes moved from their original geographical positions.

The first type of individuals resulted in an immediate solution in case of no collisions, or in faster convergence in case of minor constraint violations. The second type provided sets of safe (though usually not optimal) trajectories. Depending on the type of water region, they were mostly generated by the method of planning a trajectory on raster grids [23], which enabled avoiding collisions with other ships as well as with stationary obstacles (for restricted waters) and by the method of planning a sequence of necessary maneuvers on open waters [24]. Both methods returned more useful results then plain randomly-generated trajectories, at the cost of consuming more computational time. In particular, the computational complexity of methods working on raster grids is always at least $O(N)$, where N is the number of points in a grid [1]. The third type of individuals (randomly modified individuals of the previous two types) was used to generate the majority of a diverse initial population and thus to ensure a vast searching space.

However, with the development of specialized collision-avoidance operators, it turned out that randomly generated initial population can bring equally good final results. Also, in some cases (restricted waters with multiple stationary constraints combined with multi-ship encounters) it is either impossible or too time-costly to find safe sets of trajectories deterministically, prior to the evolution. Therefore, we have completely abandoned previously used deterministic methods of generating the initial population in favor of spending this amount of computational time on additional generations of evolution and more refined problem-dedicated operators.

5.2 Reproduction: crossover of whole individuals, crossover of single trajectories and crossover of nodes

In the crossover phase pairs of individuals (parents) are crossed to generate new individuals (offspring). Three types of crossover operators have been designed and implemented:

(a) An offspring inherits whole trajectories from both parents and the higher-valued of the two possible trajectories is chosen.
(b) An offspring inherits whole trajectories from both parents and the choice of a particular trajectory (from the first or the second parent) is done randomly.
(c) Each of the trajectories of the offspring is a crossover of the appropriate trajectories of the parents.
(d) Each node of a trajectory is an arithmetical crossover of the nodes in the parents' trajectories.

The above listed crossover operators are shown in Fig. 5.

Of these operators, the first one (a)—inheriting the higher valued of the two possible trajectories—was designed to combine the best features of two parent individuals. Thus, it is the only operator which should statistically produce the offspring higher valued than the parents. Unfortunately, using this operator has to be preceded by the evaluation phase, which enforces applying the evolutionary scheme with doubled evaluation phase. Therefore, during experiments, described later in the paper, it will be tested whether its advantages compensate for the additional computational time, which results in a lesser number of possible generations.

As for the other operators, there is no guarantee or even high probability that offspring of two highly valued parents will be highly valued itself. For example, in case of random trajectory inheriting (b), the resulting trajectories may not fit to other trajectories (collisions between ships). Therefore, to make sure, that the best individuals will not be lost (the parents might be better fitted than their offspring), the overlapping populations are used. As a result, reproduction doubles the temporary population size.

5.3 Specialized operators

We have decided to differ between typical random mutation (the next subsection) and problem-dedicated specialized operators, described in this section. Specialized operators, responsible for more conscious improving of trajectories (as opposed to random mutation) can result in a faster convergence to a solution. Instead of mixed mutation approach, favored by some researchers [9], we made the choice of a particular specialized operator dependant on the current state instead of previous states.

The evolutionary operators, which have been used here, can be divided into three groups, with group 1 only applied for restricted waters. On restricted waters, the order of applying collision avoidance operators for collisions with landmasses and other ships is such that operators handling violations of stationary constraints precede operators handling violations of other ships' domains. The reason for this order is as follows: a violation of stationary constraint must always be handled, since it is disastrous regardless of other ships' behavior. However, violation of other ships' domains may be no longer valid after violations of stationary constraints have been handled, because the operators responsible for avoiding violations of stationary constraints may have changed the trajectories in such a way that previously detected ship-to-ship collisions would not occur.

The following operators have been used:

(1) Operators avoiding collisions with stationary obstacles (restricted waters only). If a segment of a trajectory crosses a landmass or other stationary obstacle the amount of time remaining to collision and time remaining to reaching the next node is checked. A succession of small alterations of course should be avoided (COLREGS, Rule 8 b). We also assume that a new course should be kept for at least 3 minutes to be "readily apparent" to other ships' ARPA systems and navigators. Therefore, a new node or a segment can only be inserted in such a way that 3 minute intervals between course changes are kept. Thus one of the five operators is chosen based on the following rules:
   (a) Segment insertion—if only there is enough time for three course alterations, a new segment is inserted.
   (b) Node insertion—if there is not enough time for a whole new segment (additional three course alterations), a single node is inserted.
   (c) First node shift—if there is not enough time for a node insertion (additional two course alterations) and the collision point is much closer to the first node of a segment, the first node is moved away from the collision point.
   (d) Second node shift—if there is not enough time for a node insertion (additional two course alterations) and the collision point is much closer to the second node of a segment, the second node is moved away from the collision point.
   (e) Segment shift—if there is not enough time for a node insertion (additional two course alterations) and the collision point is close to the middle of a segment, the whole segment is moved away from the collision point.

   The operators are shown in Fig. 6. The direction of a maneuver is here chosen deterministically (away from the collision point) and its size is chosen randomly from a range computed on the basis of the length of the part of the segment, which violates a constraint.

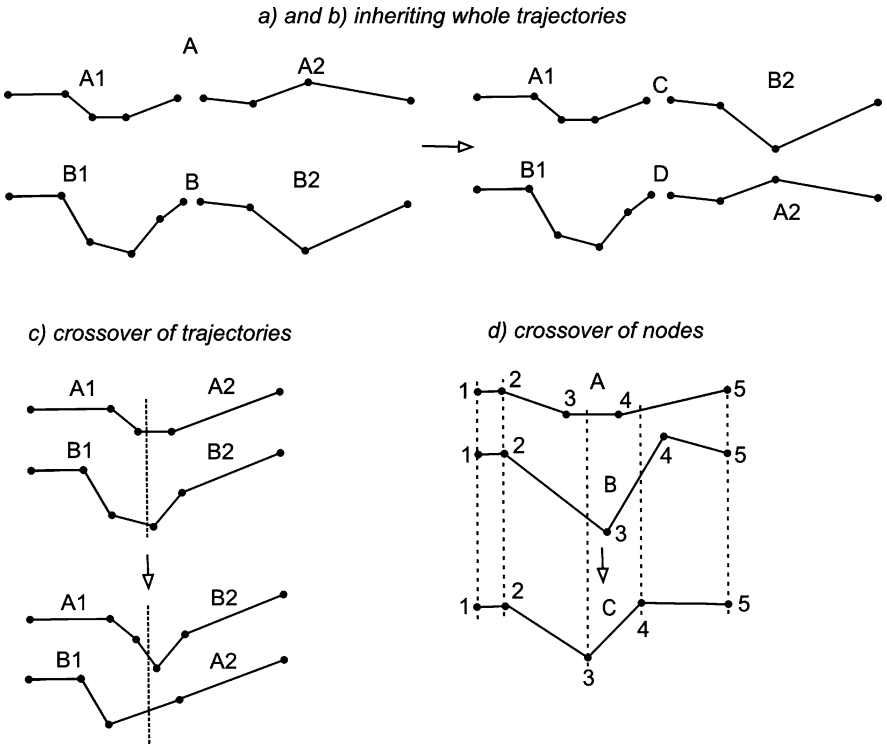

Fig. 5 Reproduction: inheriting whole trajectories (**a**) and (**b**), crossover of trajectories (**c**) and crossover of nodes (**d**)

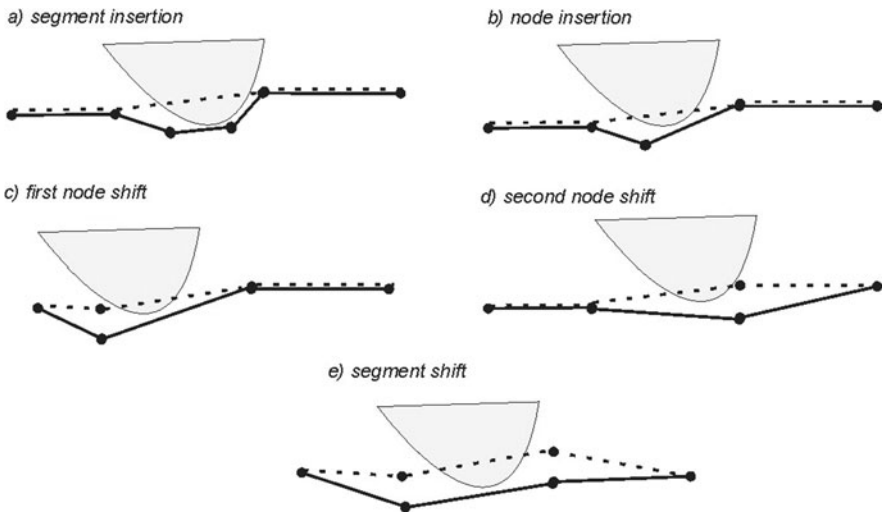

Fig. 6 Specialized operators: avoiding collisions with stationary obstacles

(2) Operators avoiding collisions with prioritized ships. Five types of these operators have been used, all operating on single trajectories and similar to those avoiding collisions with static obstacles. If a collision with a prioritized ship has been registered, one of five possible operators is selected depending on the values of a time remaining to a collision and a time remaining to reaching the next node, similarly as for group 1.

These operators are shown in Fig. 7.

(3) Validations and fixing. This group includes three operators, shown in Fig. 8.

(a) Node reduction—its purpose is to eliminate all the unnecessary nodes. If a segment, which bypasses a given node by joining its neighbors, is safe, the node is deleted. This procedure is repeated iteratively until there are no unnecessary nodes in a trajectory.

(b) Smoothing—if a course alteration is larger than 30 degrees, a node is replaced with a segment to smoothen the trajectory.

(c) Adjusting maneuvers—each trajectory of an individual is analyzed and in case of unacceptable maneuvers (such as slight course alterations), the nodes

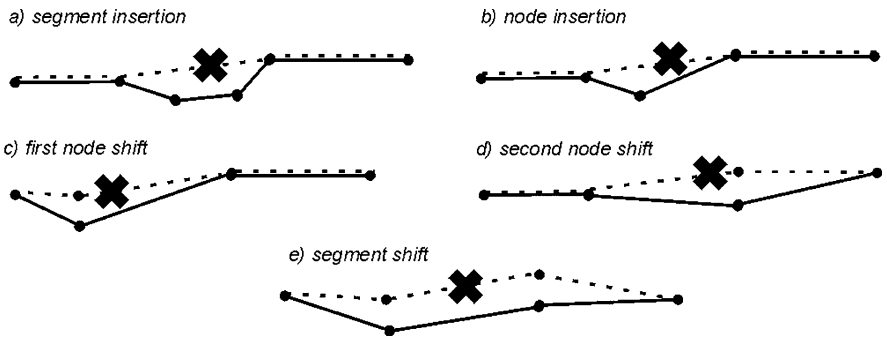

Fig. 7 Specialized operators: avoiding collisions with targets

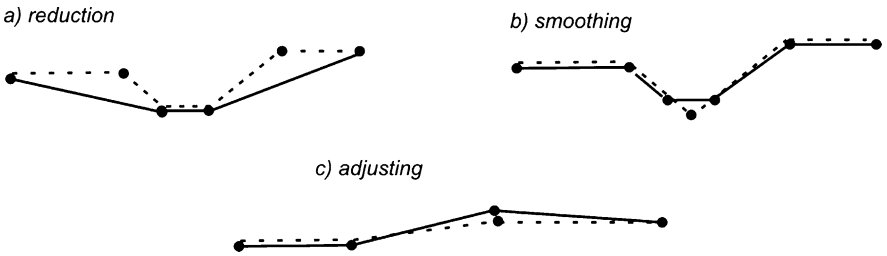

Fig. 8 Validations and fixing operators: node reduction (**a**), trajectory smoothing (**b**) and trajectory adjusting (**c**)

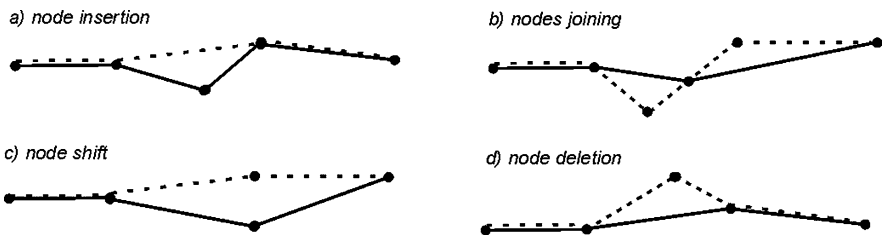

Fig. 9 Random mutation operators

being responsible are moved so as to round a maneuver up or down to an acceptable value.

In general, none of the operations described above guarantees success (avoiding the collision with a given target, avoiding a collision with an obstacle etc.), but they all are likely to do so and therefore are highly effective statistically, which is enough for evolutionary purposes.

We have decided, that all the above listed operators would be used whenever needed (fixing probability parameter set to 1), as opposed to mutation. There is no risk of spoiling, and thus losing, a high valued individual, because overlapping populations are used and the specialized operators work on individuals' copies, increasing the temporary population size.

5.4 Mutation

The mutation operations are applied to an individual's copy. Four types of these random operators have been used, all operating on single trajectories. These operators are:

(a) node insertion: a node is inserted randomly into the trajectory,
(b) node joining: two neighboring nodes are joined, the new node being the middle point of the segment joining them,
(c) node shift: a randomly selected node is moved (its polar coordinates are altered),
(d) node deletion: a randomly selected node is deleted.

They are shown in Fig. 9.

Since not all of the nodes can be subject to modifications, instead of traditional mutation probability typical for genetic algorithms [7, 32] we introduce the term of trajectory mutation probability. By trajectory mutation probability we mean the conditional probability of using any of the mutation operators on a trajectory, provided that no specialized operator has been used for this trajectory before in this generation. A trajectory mutation probability m_{tr} decreases with the increase of the trajectory fitness value $fitness_{tr\,i}$ (6), so as to mutate the worst trajectories of each individual first, without spoiling its best trajectories. It is computed according to the formula below:

$$m_{tr} = m_b \cdot (1 - fitness_{tr\,i}), \tag{7}$$

where m_b: basic mutation probability, a configuration parameter, usually set to a value from range $[0.05, 0.2]$, thus

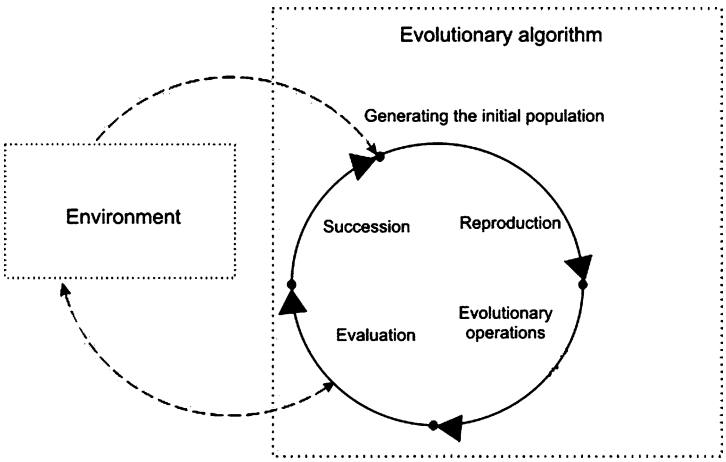

Fig. 10 Evolutionary algorithms—traditional scheme

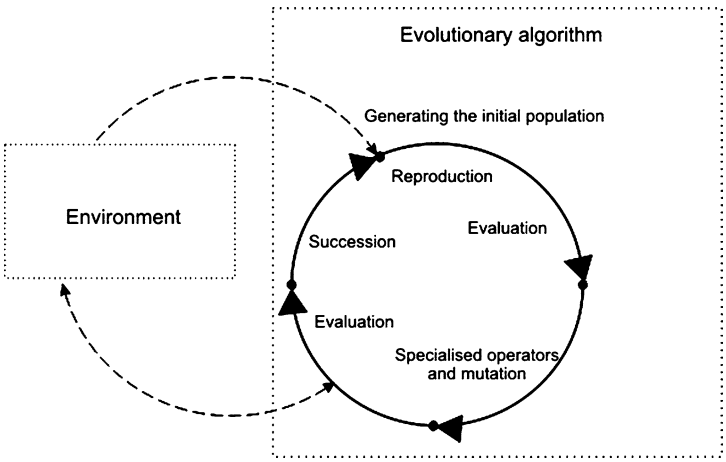

Fig. 11 Modified evolutionary algorithm—early version

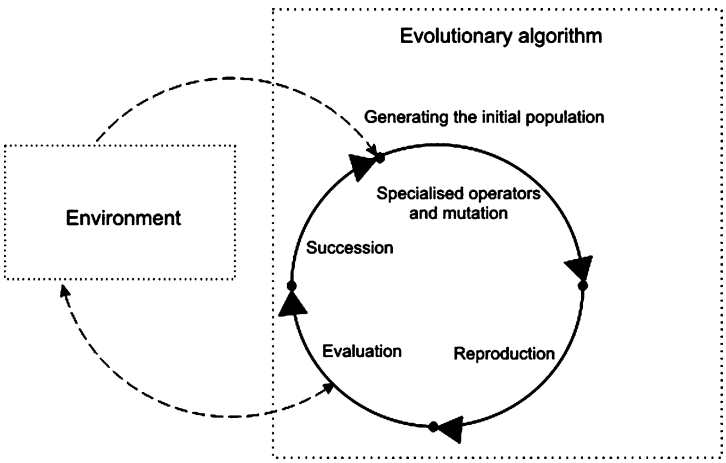

Fig. 12 Modified evolutionary algorithm—final version

larger than typically in genetic algorithms, where mutation is usually secondary to crossover [8].

In the early generations of the evolution all random operators: the node insertion, deletion, joining and shift are equally probable. In the later generations node shift dominates with its course alteration changes and distance changes decreasing with the number of generations. For node insertion and node shift instead of Cartesian coordinates x and y, the polar coordinates (course alteration and distance) are mutated in such a way that the new maneuvers are between 15 and 60 degrees. As a result, fruitless mutations (the ones leaving to invalid trajectories) are avoided for these two operators.

5.5 Selection: two different methods

The next phase of our interest is selection. We have decided to use various selection methods [10] for reproduction and succession for the following reasons:

1. In case of crossover: a low-valued individual may have one of its trajectories of a high value and therefore may be a much better parent than it is an individual. In early phase of evolution such individual might also be a better parent than one with all trajectories acceptable, but not any of them outstanding. An individual excelling in one aspect (one perfect trajectory) may be a better parent than another, having balanced trajectory values. Also, crossover is done directly after succession, when the majority of the weakest individuals (products of previous crossover) have been eliminated anyway, as opposed to succession when much larger population is a subject to selection and some of the individuals may have low fitness.
2. In case of succession: it is absolutely necessary to rely on fitness function value to progress, therefore, higher valued individuals must be favored.

5.6 A new scheme of the evolutionary process

The traditional evolutionary cycle is presented in Fig. 10. Unfortunately, it is not possible to incorporate specialized operators described in Sect. 5.3 directly into this cycle. These operators use the information returned by evaluation (fitness function values as well as the data on detected collisions of ships with other ships or with landmass) to improve the trajectories (eliminate some of the constraint violations, etc.). By doing this they raise the rate of progress per generation, which allows for a much lesser number of generations to obtain the same results. But using the evaluation data means that evaluation has to be performed prior to the specialized operations work and therefore the evolutionary algorithm has to be modified as shown in Fig. 11.

However, since the evaluation requires collision detection, it is the most time consuming phase of the cycle. In the modified evolutionary algorithm shown above it would be performed twice: once for a doubled population (after reproduction) and again after mutation, for population four times the size of the original one. Therefore, doubling the evaluation phase in a cycle increases the total computational time approximately 1.5 times (the extra evaluation after reproduction is done for a population half the size of the one after mutation). To shorten the process, we have decided to apply a radical change in the order of operations within the algorithm. The reproduction phase and specialized operations/mutation phase have changed places with each other and the evaluation is done only once for each cycle—directly preceding succession. The result is shown in Fig. 12. By applying this we have managed to combine the potentially higher rate of fitness function progress per generation with an unaffected computational time for each generation.

6 Visualisation of an exemplary result

Below we present an exemplary ESoSST method's result for a scenario of an encounter of 6 ships on restricted waters. The result (set of six trajectories) was obtained for a single evolutionary run (with parameters presented in Table 2). It illustrates how the solution avoids all of the penalties described in Sect. 4, while minimizing the way loss. Ship positions for the selected moments between the start and finish time of the ship movement (0%, 20%, 40%, 60%, 80% and 100%) are presented in Figs. 13–18 respectively.

To shorten analysis of the scenario presented in Figs. 13–18 (with ship positions given in Table 3) let us group the ships as follows:

– ship 3, ship 4 & ship 5, forming group 1, heading westbound,
– ship 2 and ship 6, forming group 2, heading eastbound,
– ship 1 heading southbound.

All ships from group 1 must avoid collision with an obstacle and do that by maneuvering to port which additionally results in their crossing astern of the ships from group 2. Of these ships, ship 5 performs the largest course alteration, to pass safely ahead of ship 4, which is on its starboard. Ships from group 2, which would normally give way to ships from group 1, benefit from the maneuvers of the ships from group 1. The only course changes of the ships from group 2 are due to landmass avoiding after passing ahead of the ships from group 1. Ship 1 has to avoid collision with the obstacle and with ships from group 2. The shortest way to do that is to alter its course to port, which results in crossing safely ahead of ships from group 2. The solution avoids all of the penalties described in Sect. 3. No ship domain or static constraint has been violated and the COLREGS violation penalties have not been applied either because the maneuvers to port have been done or to avoid collisions with landmass. Choosing the maneuvers to starboard by ship 1 or ships from group 1 would result in a much larger way loss and thus smaller fitness function value.

7 Simulation experiments and discussion of their results

For all simulation experiments presented in this section, a set of 100 test scenarios were used. 50 of the scenarios were encounter situations on open waters and the other 50—on

restricted waters. The number of ships ranges from 2 to 6. For each number of ships 10 scenarios have been generated covering all basic encounter types (head-on, overtaking and crossing with various combinations of courses). We have used a random generator, whose parameters included:

– water region type (open or restricted),
– the longitude/latitude frame,
– number of ships,
– encounter type,
– the option of generating a group of ships.

The generator works as follows. Motion parameters of the first ship are always generated randomly for the given frame (repeated, if start point or destination point are not on water). The initial positions of the subsequent ships are also generated randomly, but other parameters are computed automatically so as to make sure that ships will crash if none of them maneuvers. As for the encounter type parameter, it specifies the relation between the first two of the generated ships and limits the possible range of courses for the second ship. The option of generating a group of ships is used for larger total numbers of ships (5 or 6) to generate 2 or 3 ships of the same courses and close initial positions. For a ship, whose course collides with such a group of ships, it is harder to maneuver because usually such an encounter cannot be decomposed to a series of ship-to-ship encounters. A single, larger course change has to be applied then, which transfers to a larger way loss. In case of 5 or 6 ships, 2 out of 10 scenarios have included groups of ships.

Table 2 Parameters of the ESoSST utilized or obtaining the exemplary result

Generations	100
Population size	100
Basic mutation probability	0.05
Probability of applying a specialized operator in case of collision	1.00

Table 3 Restricted water complex scenario—ship positions & resulting fitness values

	Origin position	Destination position	V [kn]	Resulting trajectory fitness value	Resulting average fitness value
Ship 1	21° 29′ 58″ E	21° 39′ 13″ E	13.18	0.9137	
	59° 58′ 05″ N	59° 44′ 44″ N			
Ship 2	21° 25′ 45″ E	21° 43′ 24″ E	14.54	0.9909	
	59° 45′ 05″ N	59° 57′ 44″ N			
Ship 3	21° 51′ 33″ E	21° 17′ 38″ E	17.67	0.9139	
	59° 54′ 51″ N	59° 47′ 58″ N			0.9565
Ship 4	21° 45′ 43″ E	21° 23′ 26″ E	12.43	0.9004	
	59° 48′ 07″ N	59° 54′ 42″ N			
Ship 5	21° 42′ 05″ E	21° 27′ 15″ E	14.61	0.9374	
	59° 44′ 35″ N	59° 58′ 17″ N			
Ship 6	21° 19′ 24″ E	21° 44′ 04″ E	13.32	0.9893	
	59° 47′ 39″ N	59° 53′ 56″ N			

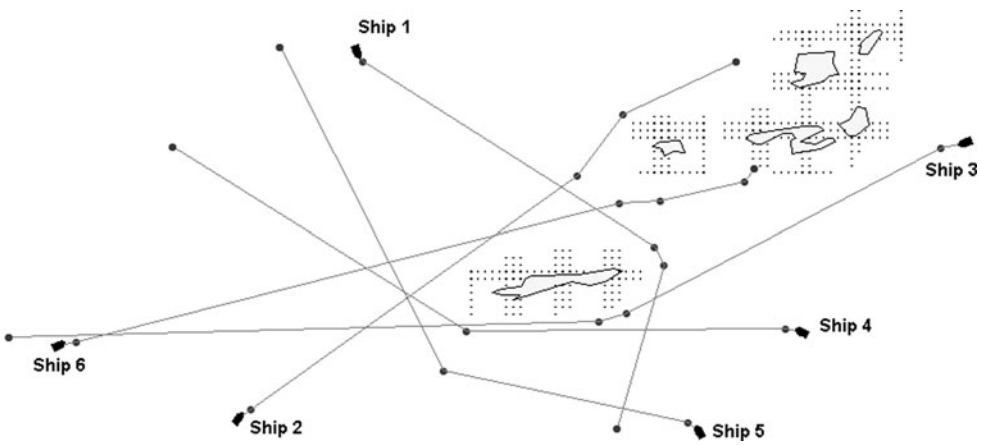

Fig. 13 Restricted water scenario—*dotted areas* depict shallow waters (initial positions)

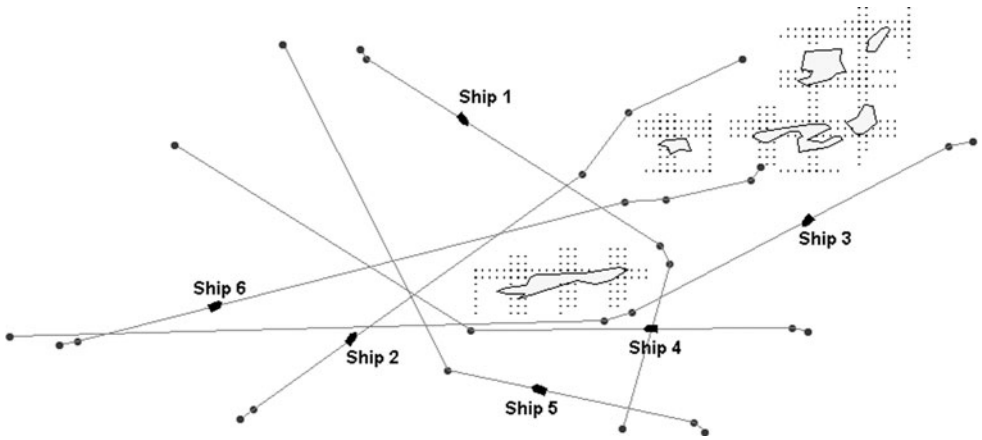

Fig. 14 Restricted water scenario—*dotted areas* depict shallow waters (positions after 20% of the animation time)

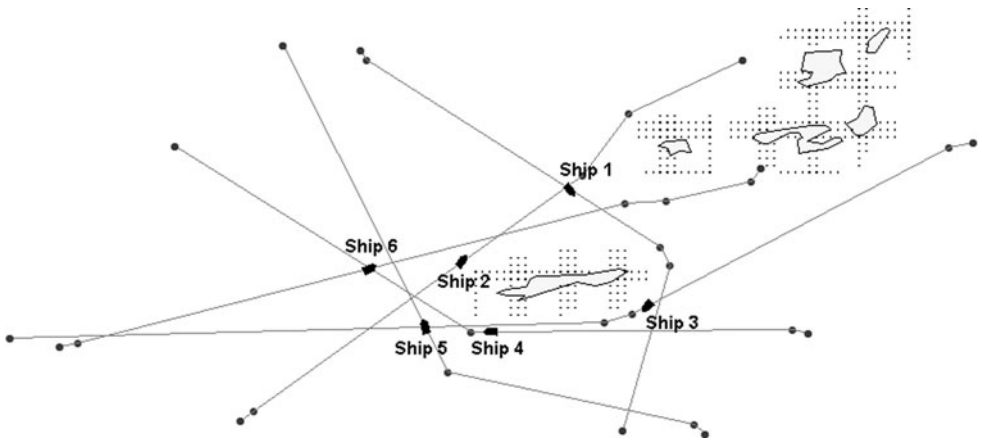

Fig. 15 Restricted water scenario—*dotted areas* depict shallow waters (positions after 40% of the animation time)

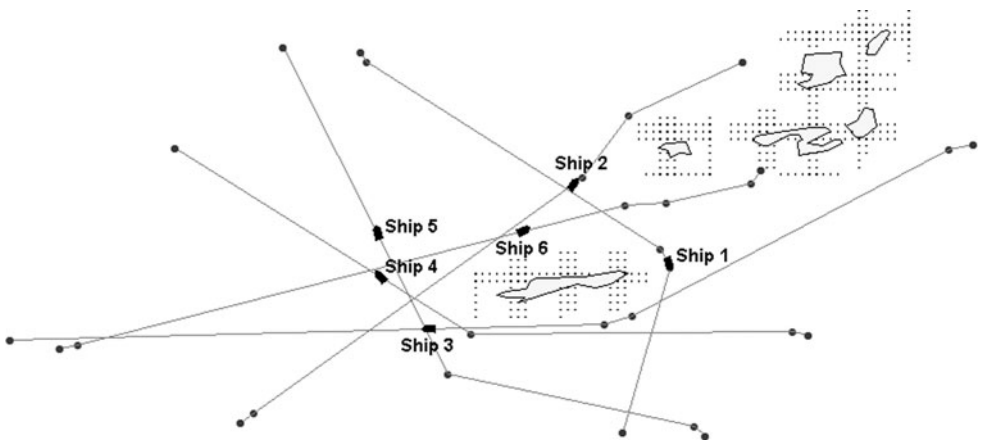

Fig. 16 Restricted water scenario—*dotted areas* depict shallow waters (positions after 60% of the animation time)

7.1 Comparing the performance of post-selection (succession) algorithms

The purpose of this test was choosing the best of the post-selection algorithms taken into account. The following types of selections have been tested here:

(a) Truncation: the highest valued individuals are selected.
(b) Random proportional with threshold: probability of being selected is proportional to fitness function value, but only the upper percentage of individuals can be selected (the lowest valued are eliminated).
(c) Modified random proportional with threshold: similar to random proportional with threshold but probability of being selected is proportional to scaled fitness function value (8) for increasing selective pressure.

$$scaled\_fitness_i = fitness_i - \min_{k=1..j}(fitness_{tr\,k}), \tag{8}$$

where j: number of all individuals.

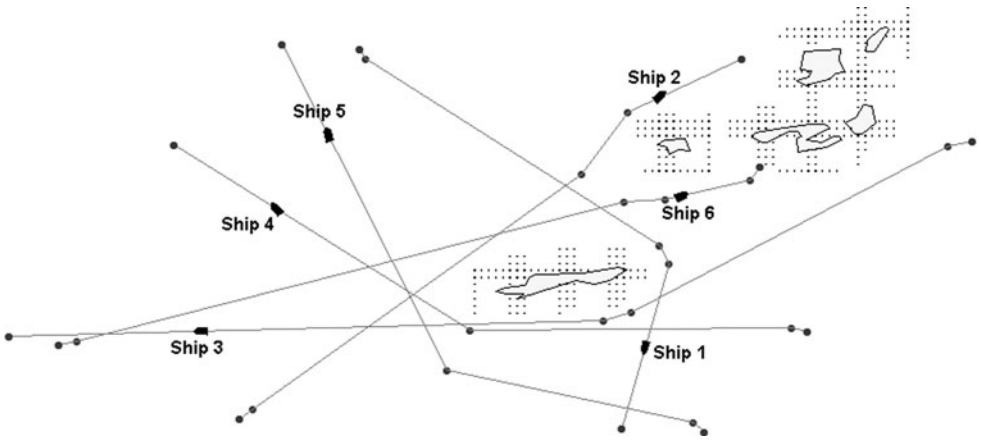

Fig. 17 Restricted water scenario—*dotted areas* depict shallow waters (positions after 80% of the animation time)

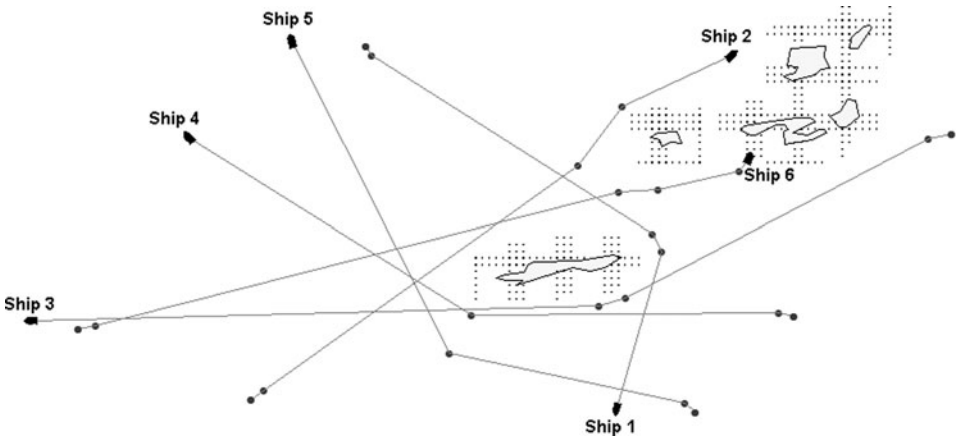

Fig. 18 Restricted water scenario—*dotted areas* depict shallow waters (destinations reached—100% of the animation time)

Additionally, various values of elite size—the number of highest-valued individuals which are automatically selected for succession—have been tested in case of random proportional and modified random proportional selection. The elite individuals can either be removed from the population after selecting them for the next generation or can be additionally included for proportional or modified proportional method (returned to the pool) to have a chance of being selected again. As for the threshold, its sizes of 50% and 100% have been tested. In case of threshold equal to 50%, only the higher valued half of the population is included for further proportional or modified proportional selection. In case of 100% all of the individuals take part (have a chance of being selected). The test parameters and the test results have been gathered in Tables 4–10.

For 100 generations (Tables 5–7) basic truncation selection turned out to be more effective than most variants of random proportional and modified random proportional selections. The reason is probably the superiority of fast convergence over diverse population for such a small number of generations. Very few variants of random selections obtained better results than truncation selection and all of them featured large elite sizes, which practically meant an approach very similar to truncation selection, with only a narrow margin left for real random selection. The best results were returned by random proportional selection with an elite consisting of 80 individuals and the remaining 20 individuals chosen randomly from whole population.

Table 4 Test parameters for simulation experiments

Test scenarios	100
Runs for each combination of scenario and selection method	10
Generations	100/200
Population size	100
Basic mutation probability	0.05
Probability of applying a specialised operator in case of collision	1.00

Table 5 Average fitness function value (truncation succession) for 100 generations

Average fitness function value	0.9725

For 200 generations (Tables 8–10) basic truncation selection was also competitive and the best results were reached by selections, whose elite sizes exceeded half of their pop-

Table 6 Average fitness function values (random proportional succession) for 100 generations

Elite size	Threshold			
	50% (only better half allowed)		100% (all allowed)	
	Elite returned		Elite returned	
	Yes	No	Yes	No
0	0.9386		0.8298	
5	0.9656	0.9652	0.9618	0.9625
10	0.9659	0.9685	0.9645	0.9667
20	0.9664	0.9700	0.9673	0.9699
40	0.9682	0.9717	0.9697	0.9703
60	0.9680	0.9722	0.9727	0.9728
80	0.9719	0.9730	0.9731	0.9730

Table 7 Average fitness function values (modified random proportional succession) for 100 generations

Elite size	Threshold			
	50% (only better half allowed)		100% (all allowed)	
	Elite returned		Elite returned	
	Yes	No	Yes	No
0	0.9436		0.8931	
5	0.9638	0.9665	0.9619	0.9632
10	0.9648	0.9678	0.9640	0.9665
20	0.9670	0.9698	0.9660	0.9676
40	0.9681	0.9710	0.9682	0.9688
60	0.9707	0.9725	0.9695	0.9699
80	0.9721	0.9728	0.9709	0.9713

Table 8 Average fitness function value (truncation succession) for 200 generations

Average fitness function value	0.9741

ulations. The highest fitness function value was again obtained for random proportional selection with elite of 80 individuals and the remaining 20 individuals chosen randomly from the whole population. This time however, the results indicate that with the growing number of generations the elite should not be returned to the population for random selection of the remaining part of the next generation.

Generally, this comparative simulation has shown that for this particular optimization problem, where operating in real time drastically limits the number of possible generations to about 100–200, the simplest truncation selection is highly competitive. It is also more flexible, since it does not need adjusting the values of parameters depending on the situation.

Table 9 Average fitness function values (random proportional succession) for 200 generations

Elite size	Threshold			
	50% (only better half allowed)		100% (all allowed)	
	Elite returned		Elite returned	
	Yes	No	Yes	No
0	0.9476		0.8290	
5	0.9677	0.9699	0.9673	0.9560
10	0.9696	0.9710	0.9701	0.9710
20	0.9706	0.9722	0.9710	0.9719
40	0.9718	0.9739	0.9716	0.9731
60	0.9725	0.9740	0.9727	0.9747
80	0.9730	0.9741	0.9737	0.9748

Table 10 Average fitness function values (modified random proportional succession) for 200 generations

Elite size	Threshold			
	50% (only better half allowed)		100% (all allowed)	
	Elite returned		Elite returned	
	Yes	No	Yes	No
0	0.9559		0.9201	
5	0.9702	0.9700	0.9652	0.9649
10	0.9706	0.9714	0.9671	0.9672
20	0.9717	0.9722	0.9700	0.9705
40	0.9722	0.9736	0.9709	0.9729
60	0.9727	0.9739	0.9719	0.9740
80	0.9730	0.9741	0.9725	0.9741

7.2 Comparing the performance of pre-selection (mating) algorithms

The purpose of this test was choosing the best of the pre-selection algorithms taken into account. The following types of selections have been tested:

(a) Threshold,
(b) Random proportional,
(c) Modified random proportional,
(d) Uniform—all individuals have the same chance of being selected as parents.

Similarly as in post-selection, various values of the elite size have been tested in case of uniform, random proportional and modified random proportional selections. The differences between results for various methods were insignificant and therefore it has been decided to apply uniform selection for pre-selection, because it enabled us to eliminate the additional evaluation directly preceding the crossover (scheme from Fig. 12 instead of Fig. 11).

Table 11 Average fitness function values for ESoSST method with and without specialized operators (100 generations for full evolutionary method with all operators)

The method	Number of generations	Average fitness function
Method with all operators	100	0.9725
Method without specialized operators (basic mutation only)	105	0.9585

Table 12 Average fitness function values for ESoSST method with and without specialized operators (200 generations for full evolutionary method with all operators)

The method	Number of generations	Average fitness function
Method with all operators	200	0.9741
Method without specialized operators (basic mutation only)	210	0.9618

7.3 Investigating the impact of specialized operators

The purpose of this test was deciding whether the benefits of specialized operators outweigh their computational cost and the necessity to change the evolutionary scheme (Fig. 12 instead of Fig. 10). Specialized, problem-dedicated operators only make sense here if they considerably improve ship trajectories. In Sect. 5.6 we showed how a new evolutionary scheme had to be introduced, to avoid doubling the evaluation phase. But, even without additional evaluation, an iteration of the cycle with the operators takes approximately 1.05 of the time spent on the same iteration without them. If running 100 generations with reversed order of reproduction and specialized operators/mutation consumes approximately the same time as running the 105 generations without specialized operations and the reproduction directly preceding mutation, it is worth checking which option returns better solutions. The results of such comparative experiments are given below, with the test parameters given previously in Table 4 (except that the number of generations was either 100/200 for the method with all operators and 105/210 for the method with basic mutation only). The numbers of generations have been set to such values that the total computational time is the same.

As can be seen, the method with specialized operators is superior to the one without them. Even with the number of generations doubled, the method with only the mutation operations cannot reach the results of its fully equipped rival (comparing Table 11 with Table 12). Therefore the traditional evolutionary scheme (Fig. 10) will be no longer taken into account and further experiments will focus on comparing evolutionary schemes from Figs. 11 and 12.

7.4 Comparing competitive evolutionary schemes

The purpose of this test was choosing the best of the competitive versions of the method, namely of the ones having different evolutionary schemes and different combinations of crossover operators applied. The basic test parameters have already been given in Table 4.

The test results for various combinations of method's settings are provided in Table 13. For each of the considered sets of crossover operators both evolutionary schemes (from Figs. 11 and 12) were tried. It is assumed, that the method would normally run for 100 generations, but the tests were also run for 200 generations to find out what further progress is possible. For both evolutionary schemes two sets of crossover operators were tried: basic one (b) and (c)—crossover of individuals and crossover of trajectories and extended one (additionally (d)—crossover of nodes). For evolutionary scheme with doubled evaluation phase also the special set of four crossover operators was tested—the extra operator was (a)—the one using information from evaluation to choose the better of the two trajectories from two parent sets. The last operator could not be tested for evolutionary scheme with single evaluation, because of the lack of up-to-date evaluation data between mutation and crossover phases.

The results led the authors to the following conclusions:

1. The differences in average fitness function values between the tested versions are insignificant, when compared to the differences obtained when testing the method with various fixing operators and mutation settings. When combined with generally high fitness function values, this suggests that the designed crossover operators (Sect. 5.2, (b), (c) and (d)) are effective enough and it is unlikely to improve the method's effectiveness by experimenting with new ones.
2. The extended set of crossover operators (a), (b), (c) and (d) brings only minor rise in fitness function and is not worth the additional computational time spent on the extra evaluation phase preceding crossover phase.
3. The set of three operators brings minor progress when compared with a set of two operators only, but it is obtained at no additional cost (the same computational time), so it might be considered an improvement over the basic set.
4. The experimental evolutionary scheme with mutation and fixing operators preceding crossover phase returned better average results for all combinations of crossover operators and maximum generation numbers. Thus, this scheme of evolutionary algorithm may be considered not only more efficient (saving on computational time due to

Table 13 Statistical test results for various versions of the method

Crossover operators used	Evolutionary scheme	Number of generations	Average fitness function values
Crossover operators (b) and (c)	Mutation and fixing operators preceding crossover, single evaluation phase (Fig. 12)	100	0.9756
		200	0.9764
Crossover operators (b), (c) and (d)		100	0.9768
		200	0.9773
Crossover operators (b) and (c)	Crossover preceding mutation and fixing operators, evaluation phase doubled (Fig. 11)	100	0.9743
		200	0.9749
Crossover operators (b), (c) and (d)		100	0.9754
		200	0.9762
Crossover operators (a), (b), (c) and (d)		100	0.9770
		200	0.9774

single valuation phase), but also more effective and generally better suited for the method.

5. As expected, the average fitness function values were always higher for 200 generations than for 100 generations but the difference was always below 0.1% of the average fitness function values. This shows that the progress is still possible with growing number of generations but the solutions returned for 100 generations are close enough to the optimal ones to be accepted and recommended by the system.
6. The favorable version of the method is the one with a set of three crossover operators and modified evolutionary scheme (Fig. 12).

8 Summary

In the paper a method of solving encounter situations—Evolutionary Sets of Safe Ship Trajectories (ESoSST)—has been presented. The method is a generalization of evolutionary trajectory finding. A set of trajectories of all the ships involved, instead of just the own trajectory, is determined. The method avoids violating ship domains and stationary constraints while obeying the COLREGS and minimizing total way loss computed over all trajectories. While developing the method it turned out that some evolutionary mechanisms had to be adjusted to the particular problem. For some of the optimization constraints gathering the data on their violations for evaluation purposes is time consuming (collisions with other ships and static obstacles), which heavily affects total computational time. When combined with the strict time limit for computations it results in limiting the number of possible generations and population size. This led us to designing specialized operators dedicated to the problem, which speed up the progress made in each generation. Using these operators enforced radical changes of the traditional evolutionary algorithm's scheme.

The full version of the method has been compared with the basic one by means of a series of simulation experiments. As expected, the results have shown superiority of the extended version over the one devoid of specialized operators, even when the latter was run for a much larger number of generations and consequently—much longer time. More surprising was the outcome of comparing chosen selection methods. It turned out that some very generic selection methods achieved better results than the more refined ones. However this can be explained by the fact that in our case the benefits of fast convergence to a solution (large progress per generation) outweigh its risks (probability of stopping at a local optimum due to a loss of diversity within population). Similarly to specialized operators different sets of crossover operators have been tested. This time though it turned out that a set including more conscious crossover (using the data returned by the additional evaluation phase) does not bring a rise in fitness function values sufficient to justify the greatly increased computational time. In general, the experiments resulted in choosing the best suited version of the ESoSST method: the one which relies on advanced specialized operators, while using a set of three specialized crossover operators and very generic selection methods: uniform for pre-selection and truncation for post-selection.

Two issues still need to be solved before verifying the ESoSST method effectiveness in the real environment (a VTS center). First, a more advanced model of ship dynamics has to be applied. The reason is that the current model, while sufficient for open waters, is not precise enough for restricted water regions, where distances between ships are smaller. Second, we need to extend the COLREGS-compliance to support additional Traffic Separation Schemes (TSS) rules, which are used to regulate the high density traffic at confined waterways, often supervised by VTS centers. The research on both aspects is currently in progress. Once it is completed, the extended ESoSST method will be evaluated with the assistance of VTS Gulf of Gdansk operators.

Acknowledgements We thank the Polish Ministry of Science and Higher Education for funding this research under grant no. N N516 186737.

References

1. Bayrak AG, Polat F (2010) Formation preserving path finding in 3-D terrains. Appl Intell. Published online: 27.11.2010. doi:10.1007/s10489-010-0265-9
2. Cheng X, Liu Z (2007) Trajectory optimization for ship navigation safety using genetic annealing algorithm. In: ICNC 2007. Third international conference on natural computation, vol 4, pp 385–392
3. Cockroft AN, Lameijer JNF (1993) A guide to collision avoidance rules. Butterworth-Heinemann Ltd, Stoneham
4. Coldwell TG (1983) Marine traffic behaviour in restricted waters. J Navig 36:431–444
5. COLREGS (1972) Convention on the international regulations for preventing collisions at sea
6. Davis PV, Dove MJ, Stockel CT (1982) A computer simulation of multi-ship encounters. J Navig 35:347–352
7. Deep K, Thakur M (2007) A new mutation operator for real coded genetic algorithms. Appl Math Comput 193:211–230
8. De Falco I, Della Cioppa A, Tarantino E (2002) Mutation-based genetic algorithm: performance evaluation. Appl Soft Comput 1:285–299
9. Dong H, He, Huang JH, Hou W (2007) Evolutionary programming using a mixed mutation strategy. Inf Sci 177:312–327
10. Eiben AE, Schoenauer M (2002) Evolutionary computing. Inf Process Lett 82:1–6
11. Fuji J, Tanaka K (1971) Traffic capacity. J Navig 24:543–552
12. Goodwin EM (1975) A statistical study of ship domains. J Navig 28:329–341
13. Ito M, Feifei Z, Yoshida N (1999) Collision avoidance control of ship with genetic algorithm. In: Proceedings of the 1999 IEEE international conference on control applications, vol 2, pp 1791–1796
14. Jozwiak L, Postula A (2002) Genetic engineering versus natural evolution: genetic algorithms with deterministic operators. J Syst Archit 48:99–112
15. Linhares A, Yanasse HH (2010) Search intensity versus search diversity: a false trade off? Appl Intell 32:279–291
16. Lisowski J (2005) Dynamic games methods in navigator decision support system for safety navigation. In: Proceedings of the European safety and reliability conference, vol 2, pp 1285–1292
17. Michalewicz Z, Fogel DB (2004) How to solve it: modern heuristics. Springer, Berlin
18. Michalewicz Z, Schoenauer M (1996) Evolutionary algorithms for constrained parameter optimization problems. J Evol Comput 4(1):1–32
19. Pradhan SK, Parhi DR, Panda AK, Behera RK (2006) Potential field method to navigate several mobile robots. Appl Intell 25:321–333
20. Smierzchalski R, Michalewicz Z (2000) Modeling of a ship trajectory in collision situations at sea by evolutionary algorithm. IEEE Trans Evol Comput 3(4):227–241
21. Statheros T, Howells G, McDonald-Maier K (2008) Autonomous ship collision avoidance navigation concepts, technologies and techniques. J Navig 61:129–142
22. Szlapczynski R (2006) A unified measure of collision risk derived from the concept of a ship domain. J Navig 59:477–490
23. Szlapczynski R (2006) A new method of ship routing on raster grids, with turn penalties and collision avoidance. J Navig 59:27–42
24. Szlapczynski R (2008) A new method of planning collision avoidance manoeuvres for multi target encounter situations. J Navig 61:307–321
25. Szlapczynski R (2011) Evolutionary sets of safe ship trajectories: a new approach to collision avoidance. J Navig 64:169–181
26. Tam CK, Bucknall R (2010) Path-planning algorithm for ships in close-range encounters. J Mar Sci Technol 15:395–407
27. Tsou MC, Hsueh CK (2010) The study of ship collision avoidance route planning by ant colony algorithm. J Mar Sci Technol 18(5):746–756
28. Tsou M-C, Kao S-L, Su C-M (2010) Decision support from genetic algorithms for ship collision avoidance route planning and alerts. J Navig 63:167–182
29. Xue Y, Lee BS, Han D (2009) Automatic collision avoidance of ships. Proc Inst Mech Eng, Part M J Eng Marit Environ, 33–46
30. Yang LL, Cao S-H, Li BZ (2006) A summary of studies on the automation of ship collision avoidance intelligence. J Jimei Univ (Nat Sci):2
31. Zeng X (2003) Evolution of the safe path for ship navigation. Appl Artif Intell 17:87–104
32. Zhao X, Gao XS, Hu ZC (2007) Evolutionary programming based on non-uniform mutation. Appl Math Comput 192:1–11

Rafal Szlapczynski M.Sc. Eng. in computer science, holds a Ph.D. in technical sciences since 2007. Currently works as an assistant professor at Gdansk University of Technology, Faculty of Ocean Engineering and Ship Technology. His scientific research focuses on application of various AI methods and heuristics to solving navigational problems, mostly in ship collision-avoidance. In 2009–2011 has been a project manager of research project for Polish Ministry of Science and Higher Education entitled "Evolutionary Sets of Safe Ship Trajectories in solving multiple ship collision situations".

Joanna Szlapczynska M.Sc. Eng. in computer science, holds a Ph.D. in technical sciences since 2009. Currently works as an assistant professor at Gdynia Maritime University, Faculty of Navigation. Her scientific research has been focused on multi-criteria optimisation of ship routes (weather routing) and application of AI methods to various navigational problems.