Modeling of Ship Trajectory in Collision Situations by an Evolutionary Algorithm

Roman Śmierzchalski, Zbigniew Michalewicz, IEEE senior member

Abstract— **For a given circumstance (i.e., a collision situation at sea) a decision support system for navigation should help the operator to choose a proper maneuver, teach them good habits, and enhance their general intuition on how to behave in similar situations in the future. By taking into account certain boundaries of the maneuvering region along with information on navigation obstacles and other moving ships, the problem of avoiding collisions is reduced to a dynamic optimization task with static and dynamic constraints. This paper presents experiments with a modified version of the Evolutionary Planner/Navigator (EP/N). Its new version, ϑEP/N++, is a major component of a such decision support system. This new extension of EP/N computes a safe-optimum path of a ship in given static and dynamic environments. A safe trajectory of the ship in a collision situation is determined on the basis of this algorithm. The introduction of a time parameter, the variable speed of the ship, and time-varying constraints representing movable ships, are the main features of the new system. Sample results of ship trajectories obtained for typical navigation situations are presented.**

Keywords— **Evolutionary algorithms, path planning, collision avoidance, dynamic environment, Evolutionary Planner/Navigator.**

I. Introduction

FINDING a safe, anti-collision maneuver is traditionally executed by drawing radar plots based on the observed echoes of the moving objects. Newly-built ships are equipped with specialized radar anti-collision systems, Automatic Radar Plotting Aids (ARPA), which facilitate the navigator's work considerably. The International Maritime Organization (IMO) has worked out a timetable for the installation of the ARPA systems on ships built since 1984 [5]. Functions executed by the ARPA system automate the activities connected with tracking the objects and provide a graphical presentation of the navigational situation. The ability to process data and display the navigational situation on the radar screen allows the navigator to make reasonable decisions about which maneuver to take. On the basis of the information obtained from the ARPA system (as well as navigator's seamanship and intuition), the final decision on how to act in order to avoid the collision must still be made individually by the navigator.

Apart from adequate preparation for its operation, proper use of the anti-collision system also requires additional algorithms that help the responsible navigator to make correct decisions. Recent tendencies in the automation of ship navigation have led to automatic calculations of the anti-collision maneuver, along with simultaneous quantitative assessment of the risk of collision based on the obtained data [3], [7], [8], [9]. An extension of the conventional anti-collision system ARPA is often desired. Without interfering with the anti-collision system itself, such an extension would both calculate an anti-collision maneuver and display this maneuver clearly to the navigator who steers the ship. The data that describe the moving objects are available as the output from the ARPA system, and can be used as input for computing procedures of a such decision support system. The system should find a set of effective solutions with respect to the assumed criteria and suggest some of them. The final acceptance of the decision suggested by the system (or the selection of an alternative decision from a set of the effective solutions) is made, of course, by the navigator.

A detailed analysis of models and the synthesis of algorithms for safe, optimum steering has been completed by Lisowski and Śmierzchalski [10]. The problem of determining a safe trajectory as a nonlinear programming task was formulated in [10], [19], where a kinematics model of the own-ship was applied.[1] Another possible approach to this problem relies on reducing the solution space to one of finite dimension by creating a so-called digitized matrix of permissible maneuvers for a given collision situation and a certain instant of time [11]. In [15], [16], [17] the problem of avoiding collisions was formulated as a multicriterion optimization task (three separate criteria were used). An attempt to estimate the safe trajectory using ge-

R. Śmierzchalski is with the Department of Ship Automation, Gdynia Maritime Academy, ul. Morska 83, 81-225 Gdynia, Poland. Email: roms@wsm.gdynia.pl.

Z. Michalewicz is with the Department of Computer Science University of North Carolina, Charlotte, NC 28223, USA and Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland. Email: zbyszek@uncc.edu.

[1] This paper uses the following terminology: the term 'own-ship' means the ship for which the path must be generated, and the terms 'strange-ship' or 'target' mean other vessels in the environment that must be avoided. This is a standard terminology [1], [2], [6].

netic algorithms was presented in [4]. The collision situation was modeled as a fuzzy process with many inputs, while steering rules were selected by a fuzzy classifier system.

The main goal of this paper is to discuss the problem of avoiding collisions at sea from the perspective of an evolutionary process. It seems that this problem is, in a sense, similar to the problem of steering a mobile robot. An evolutionary method (EP/N system) for generating paths for a robot in partially-known environments was presented in [20]. A modified version of the system has now been developed that takes into account specific characteristics of the collision avoiding process.

The new system ϑEP/N++ represents a significant extension of its earlier version, EP/N, that was tested in various environments as an approach for robot path planning. EP/N assumes static obstacles (whether known or unknown); consequently the speed of the robot (i.e., the speed of the navigated object) is not relevant to the problem of path finding (i.e., of finding an anti-collision trajectory). In contrast, ϑEP/N++ also handles also dynamic obstacles. This allows an anti-collision trajectory to consist of several segments, where each segment is traversed with a particular speed. The main innovation of this extended version is the existence of different types of static and dynamic constraints, which reflect the real environment with moving strange-ships (targets) and their dynamic characteristics. In addition, the speed of the own-ship need not be constant and the speed parameter can be utilized to gain additional efficiency. The evolutionary process searches for a near-optimum trajectory in a collision situation and takes into account a time parameter and the dynamic constraints, which represent moving strange-ships (shapes and dimensions of these strange-ships depend on assumed safety conditions). Also, the speeds of strange-ships need not be constant.

The paper is organized as follows. The next section contains a brief discussion on the structure of the steering system and provides an overview of methods for planning safe trajectories in collision situations. Section III defines the problem of planning the own-ship trajectory in collision situations, and discusses the issue of modeling moving targets in an evolutionary system. Section IV provides several examples of planning a safe trajectory. The initial experiments assume that the own-ships move with a constant speed; however, varying the speed of the own-ship during the maneuver may result in additional gains. Further experiments illustrate this point clearly. Section V concludes the paper.

II. A structure of own-ship steering system

The problem of generating a route for a ship voyage can be perceived as a set of tasks with various time horizons and multilevel control processes (see Figure 1). The most general problem with the longest time horizon (usually of several days) is termed "Route Planning (R_0)" and covers from the starting position of the ship up to the destination point. On this level, the route is selected on the basis of some economic factors, including fuel consumption, route length, and ship maintenance cost. Moreover, navigation constraints and marine weather forecasts are also taken into account.

A problem on a lower level of the planning process (so-called "Planning Safe Trajectory") is concerned with the generation of particular stages of the route. A safe maneuver must be generated in cases of approaching other targets and should take into account existing navigational limitations connected with the area's geometry (e.g., the existence of restricted zones and canals). This is the task with a relatively short time horizon (usually not exceeding one hour).

To generate a sequence of steering decisions, one can use various models of the problem (static, kinetic, dynamic, model of matrix, etc. [9]) and various methods for solving them (linear or nonlinear programming, dynamic optimization [10], [11], [16], [17], or as offered here, an evolutionary algorithm). The required trajectory is generated in two stages: "off line" and "on line." The "off line" stage assumes that all parameters involved (e.g., speeds and courses of targets) are constant. The ARPA system, however, monitors the values of all parameters on a continuous basis while the own-ship executes the trajectory. In the case of any change (course and/or speed of any target) during the "on line" stage, necessary corrections (imposed by safety requirements) to the trajectory are made. During this process the trajectory determined in the "off line" stage is taken as an initial solution for the "on line" stage and the ARPA information about navigation situation (including the prediction of target's positions) is used.

A new, corrected trajectory is passed to the "Adaptive Control" unit (see Figure 1), where the steering process takes place (this process includes many additional parameters like currents, winds, and the shape of the bottom of the sea) and is "adaptive" in the sense that the values of these parameters change continually. The unit "min Cost Criterion" (Figure 1) minimizes the effort of the main engine, taking into account overload of the torque and thrust of the propeller.

The last and lowest level of the steering system is connected with the direct control of the ship's movements in real time. Here, the control instruments (e.g.,

main engine governor and autopilot) are used to follow the trajectory selected earlier.

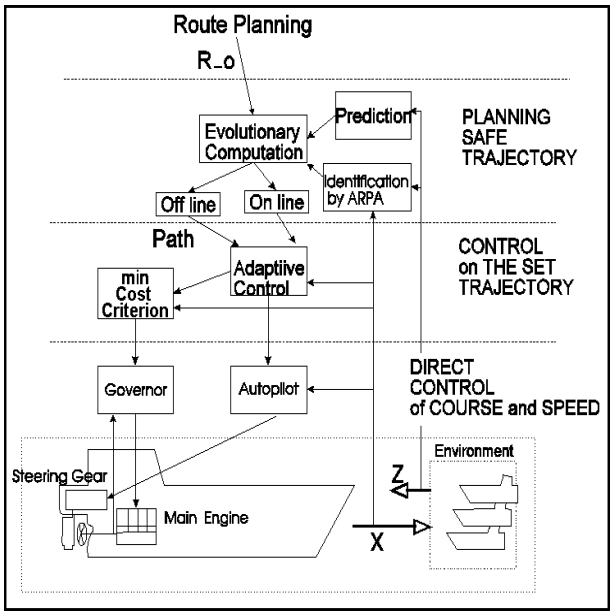


Fig. 1. Structure of an own-ship steering system in a collision situation with an evolutionary component. The system is divided into several parts: (1) route planning from the starting position of the ship up to the destination point (by evolutionary algorithm, where existing navigational limitations are taken into account; (2) adaptive control, where the steering process includes additional parameters (e.g., currents, winds, etc), (3) the minimization of the cost of main engine work, and (4) the steering of the ship movements in real time

The material presented in this paper concerns the process of planning a safe trajectory, i.e., estimating the optimum trajectory in a well-known environment modeled by navigation constraints and moving targets on the basis of the data recorded by the ARPA system. The resulting trajectory (that represents the passing path) consists of a number of line segments between the assumed starting and ending states. The formulation of the task of estimating the optimum safe trajectory in a well-known environment involves defining the steering goal and conditions at which this goal is to be reached.

The problem of safe steering of the own-ship in a collision situation is usually divided into three phases. In the first phase, the optimum safe trajectory of the ship is estimated ("off line" stage) for certain navigation conditions [5]; as indicated earlier, the navigation data are obtained from the ARPA system. These data constitute the input data for procedures computing the trajectory. In the next phase, the ship is steered along the estimated trajectory, following the distance and time preservation rule (i.e., the ship should arrive at particular locations at particular times to avoid collision). Due to the safety conditions related to the presence of other moving targets, the actual locations of the ship on the estimated trajectory must be correlated with the time of the maneuver. In the third and final phase ("on line" stage), the development of the situation is controlled in an online manner and in the case of changes in the parameters of motion of other targets, the actual trajectory is corrected. When estimating a corrected trajectory, the current location of the own-ship and current parameters of motion of the targets comprise the starting configuration. Such structure of an own-ship steering makes it possible to steer the own-ship in a well-known environment with both static and dynamic constraints, as well as to make adaptive corrections of the ship trajectory in order to follow unforeseeable changes in the situation at sea.

A. Methods for planning safe trajectories

Several methods for finding safe trajectories have been proposed [4], [10], [11], [19], [15], [16], [17]. They model the problem in different ways but most of them share common assumptions: (1) the encounter occurs at open sea (i.e., there is no land involved in the process of finding a trajectory), and (2) the strange-ships do not change their speed and course. The methods differ in the way they model the problem and the results and the computational effort of these methods vary significantly.

For experimental comparison of these methods we have used a set of typical collision situations, which is a standard collection of test cases for navigators (during a course on ARPA) at the Department of Navigation of Gdynia Maritime Academy.[2] For 12 test cases we report the average computational time of the method.[3] We discuss these approaches in turn:

Indispensable maneuver: This algorithm [18] prescribes the kinematic maneuver (single change of the course and/or the speed of the own-ship) for avoiding collision in a multitarget encounter. A geometric analysis of the vectors of velocity of the ships yields a prescription of the necessary changes of course and/or speed of own-ship in a situation of encountering a target. On this basis, a relation is derived for the minimal change of the course of own-ship to the left and to the right side of the ship at a given speed and at a given safe distance. The average computational time was 15 seconds.

Utilization of potential collision threat area: This algorithm [11] also returns a recommendation for a single change of the course and/or the speed of the own-ship.

[2]These test cases are solved on the radar simulator built by Norcontrol.

[3]All calculations were done on a Pentium Pro 200 MHz PC computer.

The algorithm determines the potential collision threat areas (PCTA). Geometrically, the safe condition is satisfied when the end of the vector of the own-ship is found outside of the PCTA danger area. The average computational time was 35 seconds.

Indispensable trajectory: The method [11] returns a sequence of changes for the course and/or speed of the own-ship. Inserting a subprocedure for the indispensable maneuver inside a program loop offers the possibility for describing the safe trajectory as a series of maneuvers. The durations of individual maneuvers are determined by the time of realization of the real maneuver considering the dynamical characteristics of the own-ship. The computational time was 5 minutes.

Optimal trajectory: The method [10], [19] also returns a sequence of changes for the course of the own-ship. The method models the problem as a nonlinear programming task with constraints, where a kinetic model of the own-ship is applied. The criterion was defined as the deviation from a given course. The safety conditions were modeled as moving areas with nonlinear admittance restriction. By digitizing the trajectory, the problem was reduced to a finite-dimension nonlinear programming task and solved using gradient methods with a penalty function. The computational time averaged 10 minutes. It is also the only listed algorithm that can be applied not only to "open sea" scenarios, but can include static navigational constraints (e.g., shore line).

The above comparison shows clear limitations of the existing methods and the need for a more general approach that:

- returns a sequence of changes for the course and/or speed of the own-ship
- is applicable in situations where targets may change their strategies (i.e., their courses and/or speeds)
- is applicable in all scenarios (with and without static navigational constraints)
- the average computational time remains reasonable (e.g., does not exceed 1 minute)

This was the main motivation for building a new version of the Evolutionary Planner/Navigator (ϑEP/N++) as a method for finding safe trajectories in collision situations. The experimental results discussed here indicate that the evolutionary system outperforms other methods and satisfies all the above requirements. Moreover, the evolutionary algorithm ϑEP/N++ is applicable to "off line" and "on line" stages of the process of safe steering in a collision situation.

III. Evolutionary algorithm and collision avoidance

This section defines the problem (environment and constraints), describes the problem of planning the own-ship trajectory in collision situations, and discusses the issue of modeling moving targets in an evolutionary system. We also describe the developed system, ϑEP/N++, and provide two simple examples of its actions.

A. Definition of environment and constraints

The ship sails in an environment with some natural constraints (e.g., land, canals, shallow waters) as well as other constraints resulting from formal regulations (e.g., traffic restricted zones, fairways). It is assumed that these constraints are stationary and that they can be defined by polygons — the very way these constraints are represented on the electronic maps. When sailing in an environment, the own-ship meets other sailing strange-ships. Some of these targets constitute a collision threat, while the others do not influence the safety of the own-ship.

The degree of the collision threat with dangerous targets is not constant and depends on the approach parameters: D_{CPA} (Distance at Closest Point of Approach) and T_{CPA} (Time of Closest Point of Approach), as well as on the speed ratio of the both ships, and the distance and bearing of the target.

It is assumed that a dangerous target is one that has appeared in the area of observation[4] and can cross the estimated course of the own-ship at a dangerous distance. In the evolutionary system, the targets threatening a collision are interpreted as moving dangerous areas having speeds that correspond to speeds of moving targets, which are determined by the ARPA system. The shapes of these dangerous areas, on the other hand, depend of the safety conditions: an assumed safe distance, D_{safe},[5] speed ratio, and bearing.

Figures 2 and 3 display models of the environment where:

- fixed navigation constraints are modeled using convex and concave polygons
- moving targets are modeled as moving hexagons
- the dimensions of the own-ship are neglected due to small length of the own-ship with respect to the maximum length of the areas representing the moving targets

B. Planning the trajectory in a collision situation

According to transport plans, the own-ship should cover a given route R_0 in some assumed time. On the

[4]Ranges of 5–8 nautical miles in front of the bow, and 2–4 nautical miles behind the stern of the ship are assumed. Their actual values depend of the assumed time horizon.

[5]A safe distance is selected by the operator depending on the weather conditions, the sailing area, and the speed of the own-ship.

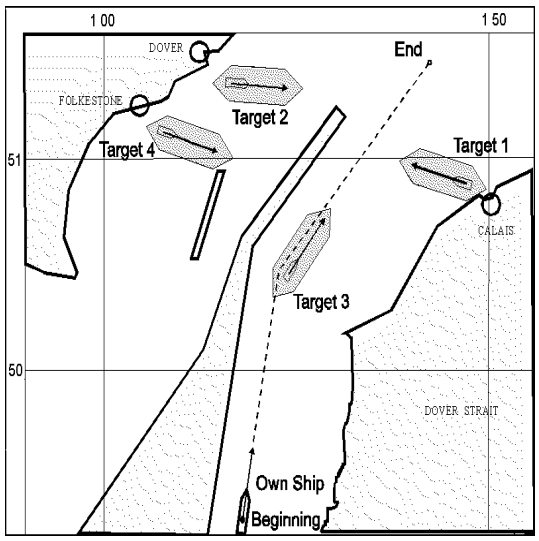


Fig. 2. Navigation situation in Dover Straits. There is an own-ship, four strange-ships, and several navigational constraints

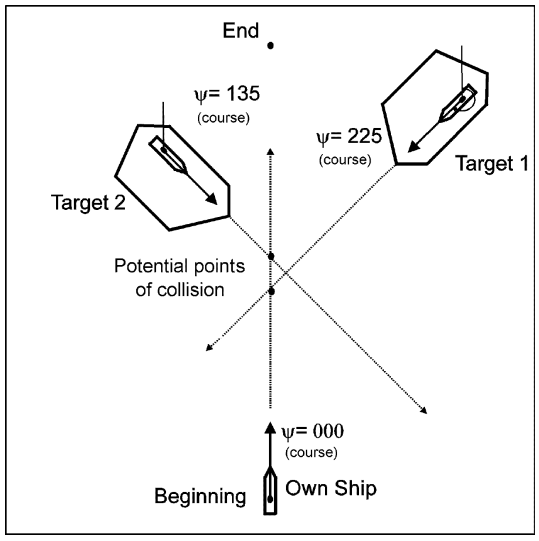


Fig. 3. Approaching two moving targets with hexagonal constraint shapes

other hand, it has to move safely down a given trajectory, i.e., it must avoid navigation obstacles and cannot come too close to other moving targets. Estimation of a ship's trajectory in a collision situation represents a difficult trade-off between a necessary deviation from a given course and the safety of sailing. Hence it is a multicriterion planning problem that takes into account the safety and economy of the ship motion.

The estimation of the own-ship trajectory in the collision situation consists of determining a path, S, as the part of the given route R_0, from the present location (starting point) $(x_0, y_0) \in R_0$ to the actual end point $(x_e, y_e) \in R_0$. This path has the form of a sequence of elementary line segments s_i $(i = 1, ..., n)$, linked with each other in turning points (x_i, y_i). The choice of the actual starting and end point depends on an assumed sensible horizon and is made by the operator. The boundaries of the environment are defined as

$$E = \{(x, y) \in R^2 \ : \ a \leq x \leq b, \ c \leq y \leq d\}; \qquad (1)$$

O_stat_j $(j = 1, ..., k)$ and $O\_dyn_j(t)$ $(j = k + 1, ..., l)$ represent the sets of static and dynamic constraints, respectively. Note that each dynamic constraint, $O\_dyn_j(t)$, is time-dependent; i.e., it defines different subareas of E for different values of t. Clearly, static constraints represent time-independent constraints (e.g., lands, canals, restricted zones, etc), whereas dynamic constraints represent strange-ships.

The space $SF(t)$ of safe (anti-collision) paths is defined as

$$SF(t) = E - \bigcup_{j=1}^{k} O\_stat_j - \bigcup_{j=k+1}^{l} O\_dyn_j(t). \qquad (2)$$

In other words, a path S is safe (i.e., it belongs to the set of safe paths $SF(t)$) if any segment s_i $(i = 1, \ldots, n)$ of S stays within the limits of environment E, does not cross static constraints O_stat_j, and at the time instances t determined by the current locations of the own-ship, does not come in contact with moving areas $O\_dyn_j(t)$ representing targets. Paths that cross the restricted areas generated by static and dynamic constrains are called unsafe, or dangerous paths.

The task of estimating the own-ship trajectory in a collision situation (so-called the *steering goal*) is performed as an evolutionary search for safe paths in the permissible space E, with subsequent selection of a near-optimum path S^* from the set SF with respect to the fitness function (defined by the path cost).

C. Evolutionary system: ϑEP/N++

Evolutionary algorithms [13], [14] have received considerable attention regarding their potential as optimization techniques for complex real-world problems. These techniques, based on the important principle of "survival of the fittest," model some natural phenomena of genetic and phenotypic inheritance and Darwinian strife for survival. They also constitute an interesting category of modern heuristic search. They constitute a class of adaptive algorithms with operations based on probabilistic methods for creating new individuals in a population of solutions. In the discussed method of the safe trajectory estimation, they are used for creating and maintaining a population of passing paths, and through a process of variation and selection, finding a near-optimum solution.

In [20] a description of an evolutionary algorithm, Evolutionary Planner/Navigator (EP/N) was provided

as a novel approach to path planning and navigation. The system unified offline planning and online planning/navigation processes in the same evolutionary algorithm that (1) accommodated different optimization criteria and changes in these criteria, (2) incorporated various types of problem-specific domain knowledge, and (3) enabled good trade-offs among near-optimality of paths, high planning efficiency, and effective handling of unknown obstacles. All reported experiments were limited, however, to static obstacles. Consequently, the speed of the controlled robot was of no significance.

Based on the EP/N planning concepts presented in [20], a modified version of the system (ϑEP/N++) has been developed which takes into account the specific character of the collision avoiding process. The new system preserves the original structure of the EP/N (e.g., it is also a steady-state system where two populations separated by one generation differ at most by a single individual, it has the same set of eight variation operators). The main innovation of the modified version is that it processes both static and dynamic constraints, which reflect the real environment of fixed navigation constraints as well as moving strangeships, whose shapes and dimensions depend on assumed safety conditions (the safe distance between the passing targets, their speed ratio, and bearing).

There are several consequences of this modification and the new system, ϑEP/N++, differs from the original EP/N in several aspects:

- it processes dynamic as well as static constraints
- it introduces the concept of time, which is essential while dealing with movable obstacles
- it allows the own-ship to change its speed

Consequently, some changes were made with respect to path representation (e.g., each path segment includes a value for speed for this segment) and operators. The evaluation function was changed as well: The concept of the best path, apart from distance, smoothness and clearance, also includes a component corresponding to the time needed to traverse the path. It was also necessary to develop new procedures for modeling dynamic obstacles for a given path and calculating a position of the own-ship with respect to these obstacles. The outline of the algorithm is shown in Figure 4.

A path S is represented by a single individual (chromosome). In the original EP/N [20], each individual consisted of variable-length sequence of genes that specified co-ordinates (x_i, y_i) of turning points between line segments s_i and s_{i+1}. The starting point (x_0, y_0) and the end point (x_e, y_e) were the same for all individuals. Each gene also contained a link to the next gene of the same chromosome and a state variable b, providing information such as whether or not (1) the knot

```
Evolutionary Navigator ϑEP/N++
{   T = 0;
    input_data_for_parametrizing_operation();
    input_data_for_defining_environment();
    P(T) = creation_of_initial_chrom_population();
    building_dynamic_obstacles();
    population_evaluation(P(T));
    while (≠ termination_condition)
        {
        T = T + 1;
        Op = operator_selection();
        Par = selection_of_parents_or_parent();
        offspring_creation(Op, Par);
        building_dynamic_obstacles();
        population_evaluation(P(T));
        introduction_of_new_individual();
        selection_of_best_individual(P(T));
        }
}
```

Fig. 4. A high-level description of the structure of the ϑEP/N++ algorithm.

point was feasible (i.e., outside obstacles), and (2) the path segment connecting the knot point to the next knot point was feasible (i.e., without intersecting obstacles). Thus, a path (or chromosome) could be either feasible or infeasible. A feasible path was collision-free, i.e., has only feasible nodes and path segments. In the ϑEP/N++ version, this representation was extended: the gene i includes the speed v of the own-ship for traversing segment i (see Figure 5).

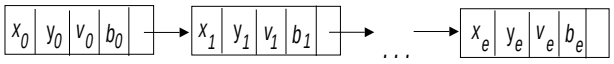


Fig. 5. A linked list chromosome representing a path. Each node contains x and y coordinates of the turning point together with a state variable b, which provides information on feasibility of the knot point and the following segment, and with a speed variable v, which determines the speed of the own-ship for traversing the segment starting at this turning point. The point (x_1, y_1) is the first turning point after the starting point and the point (x_e, y_e) is the end point (note that the starting and end points are fixed).

The initial population of m individuals is generated randomly. The maximum length of a chromosome, i.e., the number of turning points (genes), is an input parameter q of the system. Each individual contains a number of genes selected in a random manner (uniform distribution over the range $[3..q]$), and the coordinates (x_i, y_i) of each gene are randomly selected from the permissible space E. The speed variable v_i takes a random value selected (again, with uniform distribution) from a discrete domain of available speeds.

The experiments (discussed further in Section IV) indicated that the computational time of the algorithm (or, the number of generations required to reach a quality solution) depends on the number m of individuals in the population and the maximum length q of chromosomes. However, small values of m and q resulted in inferior solutions and required larger number of generations. To resolve the above tradeoff, we have set $m = 30$ and $q = 10$. We have also set the boundaries of environment E: $a = c = 0$ and $b = d = 7$ nautical miles (see formula (1)), which was a reasonable choice for ships moving with a speed up to 14 knots.

The original version of EP/N used eight types of operators to evolve chromosomes. The EP/N was a steady-state evolutionary system where only one operator was applied within a single generation to produce offspring.[6] The produced offspring (by a selected operator) replaced the worst individual in the population. Thus, the populations $P(t+1)$ and $P(t)$ differed by a single individual. The process terminated after some number of generations, which could be either fixed by the user or determined dynamically by the program itself, and the best chromosome represented the near-optimum path found.

This structure was preserved in ϑEP/N++ version with two minor changes. First, selection of the operator is performed randomly (mainly to reduce the number of parameters of the system). As before, an offspring is produced by application of a single operator (it replaces the worst individual in the population). Second, the set of operators was extended by an additional operator, *speed mutation*, and now it includes: soft mutation, mutation, adding a gene, swapping gene locations, crossing, smoothing, deleting a gene, individual repair, and speed mutation. We briefly discuss these operators in turn.[7]

Mutate_1 is used for fine tuning node coordinates in a feasible path for shape adjustment. Given a path, the operator randomly selects[8] its nodes for adjusting their coordinates within some local clearance of the path so that the path subsequently remains feasible.

Mutate_2 is used for imposing a large random change of node coordinates in a path, which can be either feasible or infeasible. Given a path, the operator randomly selects a node and changes the coordinates of this node randomly.

[6]The first version of EP/N required fixed probabilities for all operators. Later, advanced versions of EP/N [20] adapted these probabilities during the run. This was the main reason for applying one operator only in a single generation, as it was easier to evaluate operators' performance.

[7]For a full discussion on the first eight operators, see [20].

[8]Here and in the rest of the descriptions of operators, "random selection" means selection with equal probability for all outcomes.

Insert-Delete operates on an infeasible path by inserting randomly generated new nodes into infeasible path segments and deleting infeasible nodes (i.e., path nodes that are inside obstacles).

Delete removes nodes from a path, which can be either feasible or infeasible. If the path is infeasible, nodes for deletion are selected randomly. Otherwise, the operator decides whether or not a node should be deleted based on some heuristic knowledge. In the case where there is no knowledge supporting the deletion of a node, its selection for deletion is decided randomly with a small probability.

Crossover recombines two (parent) paths into two new paths. The parent paths are divided randomly into two parts respectively and recombined: The first part of the first path is put together with the second part of the second path, and the first part of the second path with the second part of the first path. Note that there can be a different number of nodes in the two parent paths.

Swap exchanges the coordinates of selected adjacent nodes in a path to eliminate two consecutive sharp turns. The path can be either feasible or infeasible. The probability for selecting a node n_i and the next node n_{i+1} is proportional to the sharpness of the two turns (measured by angles between the path segments) at the two nodes.

Smooth smooths turns of a feasible path by "cutting corners," i.e., for a selected node, the operator inserts two new nodes on the two path segments connected to that node respectively and deletes that selected node. The nodes with sharper turns are more likely to be selected.

Repair fixes a randomly selected infeasible segment in a path by "pulling" the segment around its intersecting obstacles.

Speed mutation replaces one v value for a randomly selected path segment by a random value from a predefined finite domain of possible speeds.

Some operators (e.g., "repair") defined for the original EP/N [20] modify the path by selecting an infeasible segment and then pulling the segment around the intersecting obstacles, thus repairing the selected segment. Note, that the action of the counterpart operator in ϑEP/N++ is the same, however, it repairs infeasible segments in the presence of dynamic constraints (it pulls the segment around static and dynamic obstacles).

Although "change of the trajectory" is considered as the basic anti-collision maneuver [11], in some cases an additional "change of speed" might be beneficial. For

example, if the trajectory of the own-ship passes close to the stern of some strange-ship, it might be desirable to reduce the speed of the own-ship to minimize the total length of the trajectory and to increase safety of the passage. The ϑEP/N++ system can be run in two modes: with a constant or variable speed of the own-ship (i.e., the speed mutation operator may be switched off).

As in any evolutionary algorithm, the evaluation function should distinguish between better and worse solutions (paths) for the actual steering goal in the examined environment. Thus the evaluation function $Path\_Cost(S)$ must include both the safety conditions $Safe\_Cond(S)$ of sailing along the path S, and the economic conditions $Econ\_Cond(S)$ resulting from the shape of the own-ship's route. The safety conditions are satisfied for a given path S if it does not cross the boundaries of constraints (such a path S is called a safe path). As with the original EP/N, there are two different evaluation functions for safe and dangerous (unsafe) paths. For a dangerous (i.e., unsafe) path, a number of border crossings is examined together with the length of the penetration distance inside the restricted dangerous area. On the other hand, in the case of a safe path, the quality of the path is estimated using the total past cost ($Path\_Cost(S)$):

$$Path\_Cost(S) = Safe\_Cond(S) + Econ\_Cond(S); \quad (3)$$

where

$$Safe\_Cond(S) = w_c \cdot clear(S); \quad (4)$$

$$Econ\_Cond(S) = w_d \cdot dist(S) + w_s \cdot smooth(S) + w_t \cdot time(S); \quad (5)$$

All functions, $clear(S)$, $dist(S)$, and $smooth(S)$, have the same meaning as in EP/N (see [20]); however, the procedure to calculate the clearance is more complex now as dynamic obstacles are considered. Additionally, $time(S)$ returns the total time of the own-ship to pass the path S.

Note that safety and economics are combined into one objective function. The constants w_c=1, w_d=1, w_s=3, and w_t=1, which were used as weight coefficients in all experiments, were tuned on the basis of several experiments (i.e., we tried to minimize the deviation between the best trajectory found by the evolutionary algorithm and the trajectory recommended by an experienced navigator). These values are likely far from being optimal, but they demonstrate the workings of the ϑEP/N++ system well.

As there are two independent evaluation functions (for feasible and infeasible paths), it is necessary to have a mechanism for comparing safe (feasible) with unsafe (infeasible) paths. Here we assumed that any safe trajectory has a better fitness than any unsafe trajectory. It is quite easy to implement this mechanism: It is sufficient to add an appropriate penalty constant to the evaluation scores of all infeasible paths. This mechanism, together with the "repair" operator, drives the population (which initially consists of infeasible trajectories) into a feasible area of the search space. It might be difficult to argue, however, that the ϑEP/N++ system *guarantees* a feasible solution. Some environments (e.g., a narrow harbor with over 100 ships moving in cycles) may require sophisticated trajectories that involve many small turns; for some other environments, a safe trajectory may not exist. On the other hand, in normal circumstances (e.g., the standard test cases considered in this paper), the system always finds feasible trajectory (in the first few generations).[9] The termination condition of the evolutionary loop (while loop in Figure 4) was "lack of progress in 100 iterations."

D. *Modeling of moving targets in evolutionary environment*

In our initial experiments (section IV) the own-ship moves with a speed ϑ (along the a safe path S) from the starting point (x_0, y_0) to the end point (x_e, y_e), and at the initial instant t_0, the motion of the strange-ships is defined as uniform. For each target, its motion is represented by the following parameters: bearing, distance, speed, and course, estimated by the ARPA system. The path of the own-ship has the form of a sequence of elementary line segments s_i $(i = 1, ..., n)$, linked with each other in turning points (x_i, y_i).

These initial experiments considered the speed of the own-ship constant; however, it is possible to gain additional efficiency while varying the speed. We return to this issue later in the paper (section IV-A).

It is relatively easy to initialize the population of paths: Each path (individual) can be generated randomly. Next, each path is evaluated. To determine whether or not a path is safe, the path is examined with respect to the set of static and dynamic constraints. The instantaneous locations of the dynamic areas with respect to the evaluated path depend on time t_c, determined by the first crossing point (x_c, y_c) between the own-ship's path S and the trajectory of the target. For example, in the Figure 6, these crossing points are the points of the biggest collision threat for paths 1, 2, and 3. Having known the length of the line segment from the starting point (x_0, y_0) to the crossing point (x_c, y_c) and assuming that the own-ship will keep moving with a uniform speed ϑ, it is possible to determine the time

[9]The system found feasible solutions in scenarios up to 20 targets, as required by the International Maritime Organization rules.

t_c that the own-ship needs in order to cover this distance.

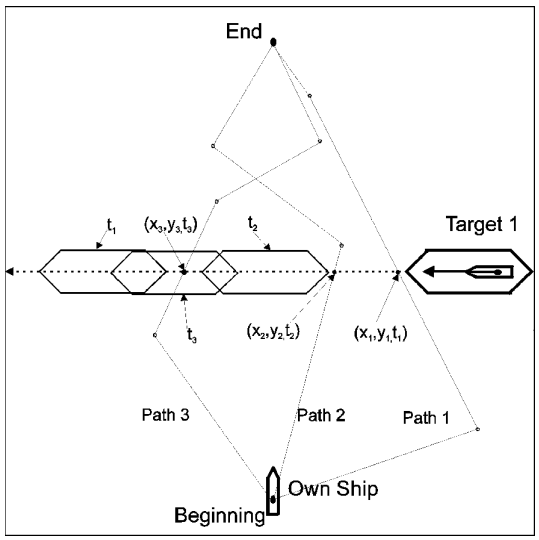


Fig. 6. Crossing paths and the corresponding dangerous areas

After time t_c, the instantaneous location of the target with respect to the own-ship is modeled as a dangerous area of hexagonal shape. Referring again to Figure 6, three locations of the target (at times t_1, t_2, and t_3) are given for three paths. Note that the path segment of path 1 between the own-ship and the intersection with the trajectory of the target is the longest one (i.e., longer than similar path segments of paths 2 and 3). Consequently, t_1 is larger than t_2 and t_3, and the hexagonal shape of the target for t_1 is the leftmost one. Of course, as explained earlier, the detailed shape and dimensions of the hexagon depend on the safety conditions given by the operator.

After the paths are evaluated, selected paths are modified by a specialized set of operators (for details, see [20]).

The values assumed in the paper are the following:
- the distance in front of the bow that guarantees avoidance of the collision is equal to $3 \cdot D_{safe}$ (in practice, safe distance D_{safe} is taken from the range between 0.5 and 3.0 nautical miles)
- the distance behind the stern is equal to D_{safe}
- the width of the dangerous area on each side of the own-ship is chosen with the preference of the ship's passage behind the stern of the target, which depends on the course and bearing of the target

E. Two simple examples

Before we present the results of the evolutionary system on several test cases, we provide two simple examples where the set of static constraints is empty and the dynamic constraints are defined by one or two strangeships, respectively.

The first example (Figure 7) shows the situation when the own-ship approaches a single target on its right side.[10] The population consisted of 10 individuals (paths), and the system converged after 300 generations (3 seconds[11] of CPU time). It is clear that the own-ship, steering along the developed trajectory, will pass the target safely, passing it behind the stern. It is interesting to note that initially (see $Generation = 50$) the own-ship tried to move "left" (somewhat along the target), but clearly, a much better maneuver is to go slightly right (as it is the case for $Generation = 300$).

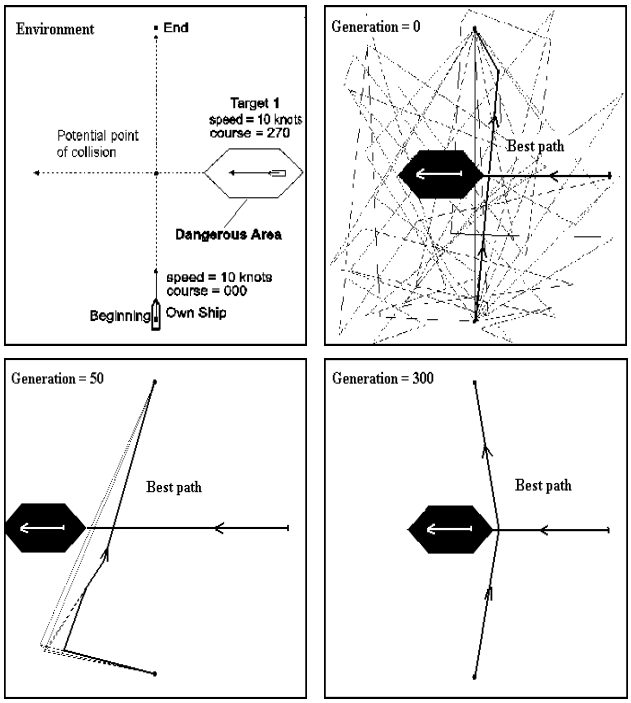


Fig. 7. Evolution of paths for the case of approaching one moving target

The second example (Figure 8) is similar to the first except that there are two moving targets sailing in opposite directions on the right and left sides of the own-ship. The population consisted of 10 paths, and the system converged after 450 generations (5 seconds of CPU time). The optimal trajectory secures the passage of the own-ship behind the sterns of the targets 1 and 2.

Note that in all three motion diagrams of Figures 7 and 8 (as well as in all further figures) the locations of the dynamic areas are shown (black hexagons) with

[10] As usual, time horizons for collision avoidance are around 30 minutes, we assumed $x = y = 8$ nautical miles.

[11] All calculations were done on a Sun Ultra-Enterprise 4000 with 2 processors UltraSPARC 167MHz.

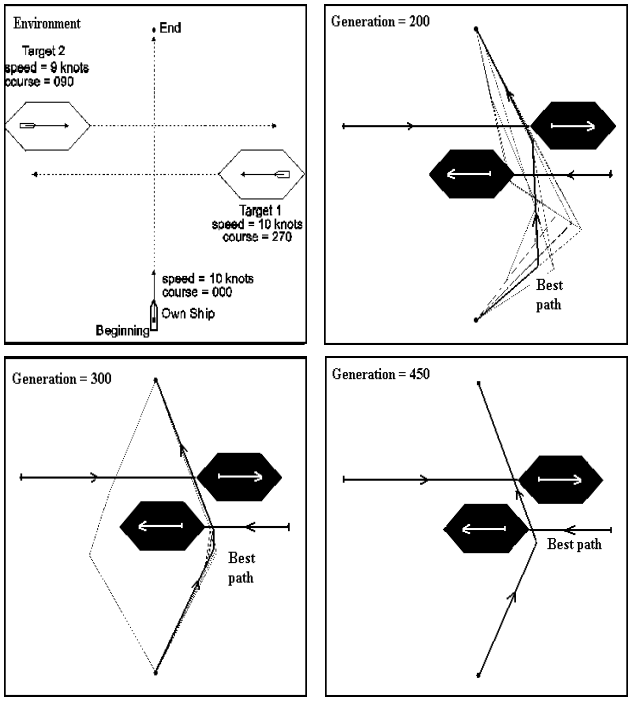


Fig. 8. Evolution of paths for two targets being approached

respect to the best path (as explained in section III-D, these locations depend on time determined by the first crossing point between the own-ship's path and the trajectory of the target).

IV. Planning a safe trajectory

The operation of the system has been examined for a number of collision situations. We discuss these environments in turn. As indicated earlier, we assume initially that the speed of the own-ship, v, is constant. In all the following experiments, the population size is 30, i.e., the evolutionary system processes 30 paths. Twenty independent runs were made for each collision situation. In each case, all 20 runs converged to almost the same trajectory, subjectively, and the progress made by the algorithm was very much the same. Thus the following discussion applies to any of the 20 runs.

In some real situations like straits, channels, etc., the number of involved ships might be quite significant (e.g., the system ARPA monitors up to 20 targets). However, only few of them are present in the potential collision threat area; a typical number of ships involved varies between 1 and 3.[12] In the same time these situations require a higher number k (see formula (2)) of static navigational constraints; often $k \geq 4$. This is an underlying scenario for experiments reported in this section.

The first example (Figure 9) is quite characteristic for sailing in narrow passages. It models an environment with static navigational constraints and with three moving targets. These three targets have different speeds (10.0, 8.0, and 5.0 knots, respectively) and they move in different directions (their courses are 225, 135, and 270 degrees, respectively) from different locations (targets 1 and 3 are initially located in the upper right corner of the environment, whereas target 2 — in the upper left corner).

The system converged after 1100 generations (42 seconds). The estimated trajectory secures a safe passage of the own-ship behind the sterns of the targets 1 and 3, and well in front of target 2.

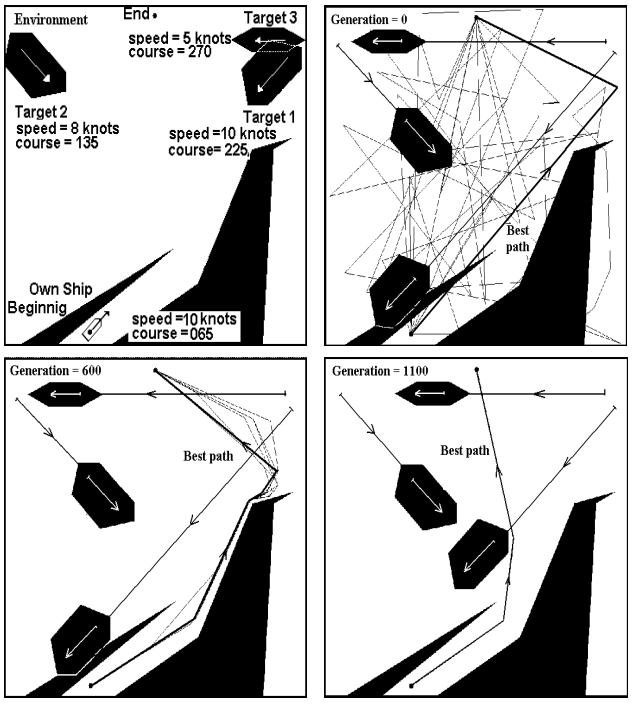


Fig. 9. Path evolution for the case of approaching three moving targets in the presence of static navigation constraints; example #1

The second example (Figure 10) represents two targets moving in parallel in the same direction (from right to left), however, with different speeds. The environment has some static navigational constraints. The system converged in 800 generations (23 seconds). The best trajectory secures the passage of the own-ship behind the sterns of targets 1 and 2.

In the third example (Figure 11) there are three moving targets distributed in three corners of the environ-

[12] The only areas where larger numbers of ships might be present are port entrances. For example, the number of ships entering the port in Rotterdam is approximately 50,000 per year. In such scenarios special traffic control systems are used (e.g., VTS — Vessel Traffic Services), whereas ARPA is used as a secondary support system.

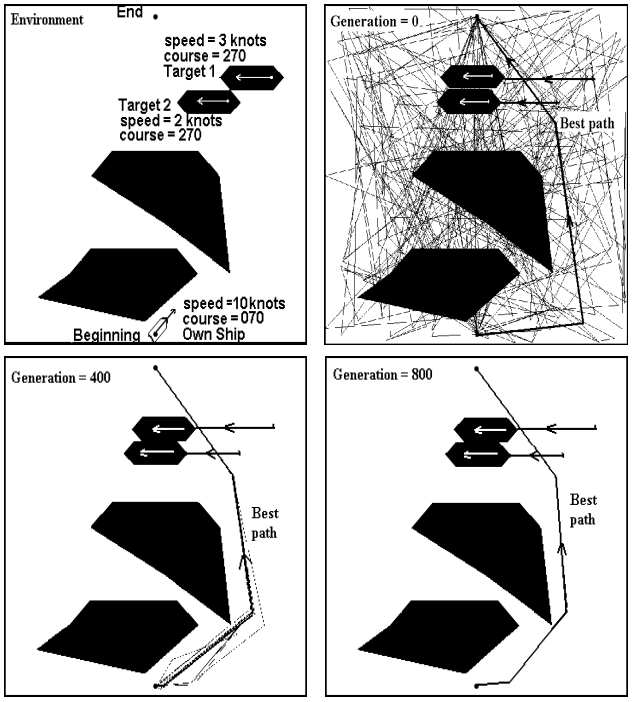


Fig. 10. Path evolution for the case of approaching two moving targets in the presence of static navigation constraints; example #2

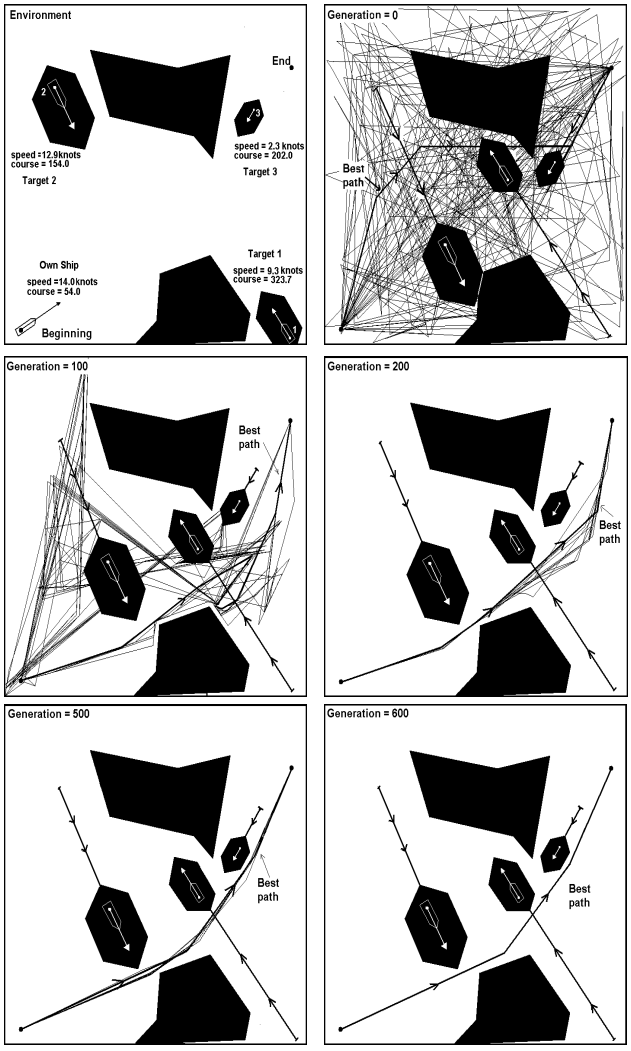


Fig. 11. Path evolution for the case of approaching two moving targets in the presence of static navigation constraints; example #3

ment: upper left, upper right, and lower right. The starting position of the own-ship is in the fourth corner: All targets and the own-ship move towards the center of the environment. There are also two static navigational constraints. The system converged after 600 generations (28 seconds). The best trajectory secures the passage of the own-ship behind the stern of target 1, in front of target 2, and along the side of target 3.

The fourth example also represents the same environment as the previous one (example #3), however the speed of target 2 is changed from 12.9 knots to 16.9 knots. Note, that in such a case the own-ship cannot secure its passage in front of target 2, thus it is necessary to pass this target behind its stern (Figure 12). Clearly, the system finds this solution easily.

These four examples illustrate the point that an evolutionary algorithm can be used effectively for solving the problem of avoiding collisions at sea ("off line" stage). Note, however, that in all experiments the own-ship was assumed to move with a uniform speed.

A. The importance of speed of the own-ship

Let us consider an additional example. Assume, the environment has six static navigational constraints (see Figure 13) and two targets. The own-ship starts close to the upper-left corner of the environment and must reach a location on the opposite side of the strait (close to the lower-right corner). Note that there is a passage between two narrow islands located in the middle of the strait, however, the movement of targets 1 and 2 makes this passage difficult. Note also that the speed of the own-ship is $\vartheta = 8.6$ knots.

Since the targets 1 and 2 are blocking the passage of the own-ship between two islands, the system developed a trajectory (Figure 13) that is "going around" the islands rather than between them. In that case, target 1 does not pose any collision threat; however, it is still necessary to avoid target 2. This task is accomplished by a tiny maneuver at the upper tip of the upper island: By making a sharp right turn and shortly afterwards, a left turn, the own-ship can avoid the hexagon of target 2.

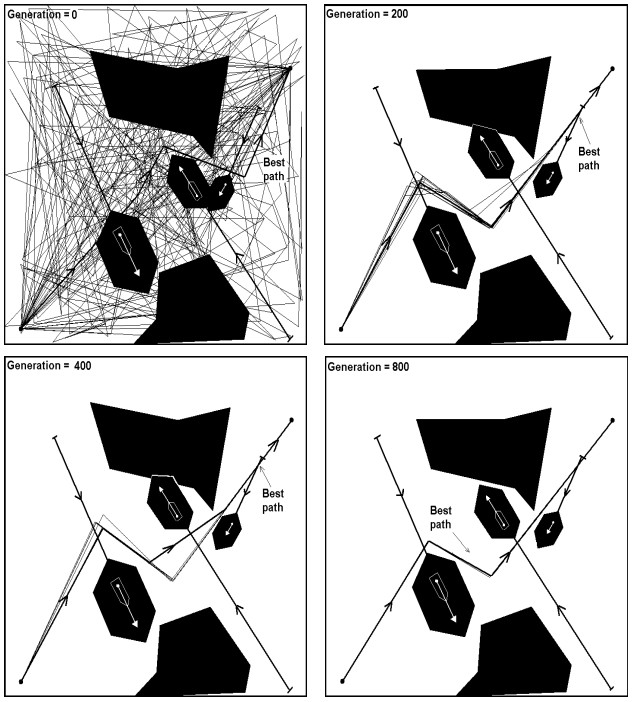


Fig. 12. Path evolution for the case of approaching two moving targets in the presence of static navigation constraints; example #4

It is interesting to repeat the same experiment, where the speed of the own-ship is reduced: $\vartheta = 5.6$ knots. In such circumstances the own-ship would approach the gap between islands at a slower rate. Consequently, it would pay off to develop some "waiting maneuvers" to let the target 1 pass the gap area. This was precisely the outcome of such an experiment (see Figure 14) where the best trajectory, after several tiny maneuvers at the beginning of the path, finds its way between islands and avoids both targets: It passes behind the sterns of targets 1 and 2.

The last two examples demonstrate that the speed of the own-ship can have an enormous influence on the final shape of the trajectory. So let us examine then the most interesting possibility when the own-ship can change its speed during the maneuver. In reality, these changes are quite limited due to the fact that "change of the trajectory" is considered as the primary anti-collision maneuver [11]. Of course, anti-collision maneuvers also allow changing the speed of the own-ship, but only when such a change decreases probability of collision (note that the unwillingness to change the speed of a ship is considered as one of the reasons for the Titanic disaster [12]).

In the following example we assume that the speed of the own-ship can vary, and that it can be either 3.6, 8.6, or 13.6 knots (slow ahead, half ahead, and full ahead).

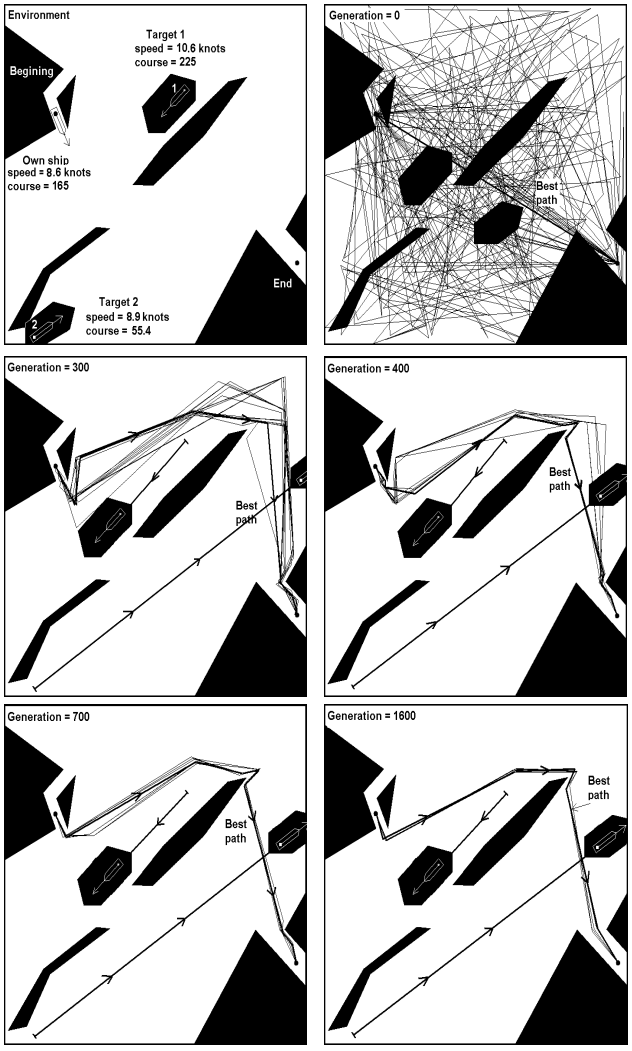


Fig. 13. Path evolution for the case of approaching two moving targets in the presence of static navigation constraints; the speed of the own-ship is $\vartheta = 8.6$ knots

Note that this simple change triggers many important changes in the code of the evolutionary system. First, the speed of the ship is represented in a chromosome for every segment of a path. Further, it was also necessary to develop operators that were responsible for changing the speed values. If speed takes one value from a predefined (and relatively small) set of values, a binary representation of the speed (together with a binary flip-bit mutation) is appropriate (this is the current state of development in our system, ϑEP/N++). However, if the changes in speed are arbitrary, it can be represented as a floating-point number (together with a Gaussian mutation, for example). Also, it is possible to represent rather a *change* in speed as opposed to speed itself — this is the area for further studies. Figure 15 displays the result of such an experiment. Again, we have used

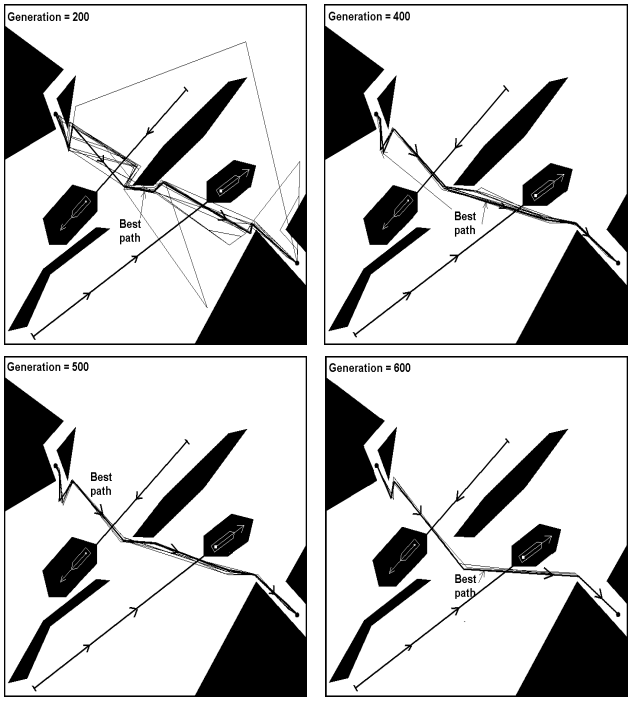


Fig. 14. Path evolution for the case of approaching two moving targets in the presence of static navigation constraints; the speed of the own-ship is $\vartheta = 5.6$ knots

the same environment as in the previous experiments (reported in Figures 13 – 14). In accordance with the "common sense" for a such environment, the initial segments of the best trajectory are covered with a minimum speed of 3.6 knots, whereas the final two segments are covered with maximum speed of 13.6 knots.

B. Further experiments with ϑEP/N++

Consider an additional scenario, in which a target changes its speed. This corresponds to "on line" stage of the steering process, where any changes are taken into account. It would be interesting to check the performance of the ϑEP/N++ system when, for example, the moving target 3 reduces its speed at some time t_1 (the environment and three targets are these of Figure 9). Note that the initial speed of the target 3 is 5.0 knots.

Now, let us consider two cases:

- at time t_1 target 3 reduces its speed to 3.0 knots
- at time t_1 target 3 reduces its speed to 4.0 knots

In the experiment, until time t_1, the own-ship has been traversing the near-optimal trajectory, identical to the one displayed in the lower-right square of Figure 9. Until that time, the speed of the target 3 was constant, equal to 5.0 knots, and from the time point t_1 there is a change in the environment: The target reduces its speed.

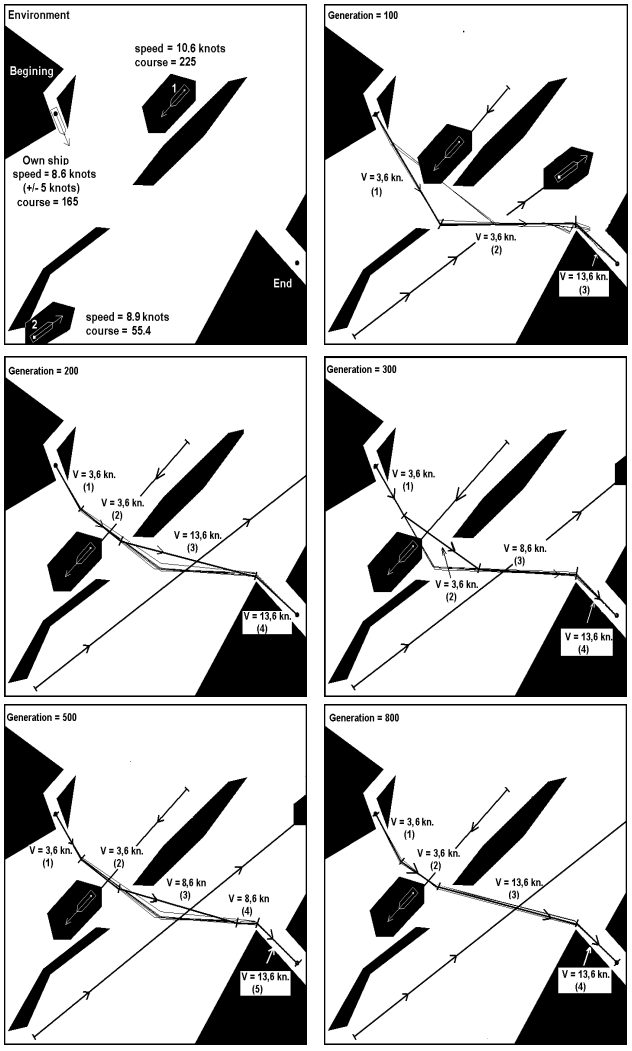


Fig. 15. Path evolution for the case of approaching two moving targets in the presence of static navigation constraints; the speed of the own-ship is variable

Figure 16 displays the results of two experiments, which correspond to the cases of speed reduction to 3.0 or to 4.0 knots, respectively. In the former case (left part of Figure 16) the system found a safe path for the own-ship in front of the bow, and, in the latter case (right part of Figure 16) the path was behind the stern of the target. Each case took 1100 generations.

This example demonstrates the flexibility of the ϑEP/N++ system, which can easily incorporate the concept of variable speed of the targets. This is important since such a model (which includes variable speed of the own-ship and variable speed of the targets) more faithfully represents the real environment.

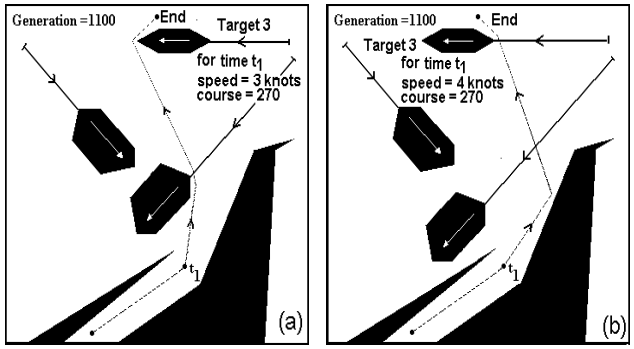


Fig. 16. Near-optimum paths of the own-ship when the target 3 reduces its speed (a) from 5.0 to 3.0 knots, and (b) from 5.0 to 4.0 knots

V. Comparisons and conclusions

The evolutionary method of estimating the safe and optimum passing path, being the own-ship's trajectory in the environment with static and dynamic constraints, represents a new approach to the problem of avoiding collisions at sea. A number of preliminary tests, presented in the paper, allows us to formulate the following conclusions:

- evolutionary algorithms can be used effectively for solving the problem of avoiding collisions at sea, where an environment is modeled as a set of polygons representing navigation constraints and moving targets (the detailed shape and dimensions of the hexagon depend on safety conditions and parameters of motion entered by the operator)
- an evolutionary algorithm can be used to adapt the speed of the own-ship to optimize the maneuver even further
- the task of evolutionary estimation of the own-ship trajectory in a collision situation is reduced to an adaptive search for a set of safe paths S in a permissible space E, with subsequent selection of the optimum trajectory with respect to the fitness function
- it is quite easy to extend the model to include variable speed of the targets
- the average computational time (on the standard twelve test cases; see section II-A) was 55 seconds
- the running time of the algorithm is a linear function of the number of dynamic and static constraints. For example, the computational time for typical environment with two static and two dynamic constraints was around 30 seconds (approx. 800 generations), whereas an addition of one or two extra targets increases the time to 40 or 50 seconds, respectively. On the other hand, the addition of one or two extra static constraints increases the time to 33 and 36 seconds, respectively.

It is interesting to compare various algorithms for planning safe trajectories in collision situations (see Section II-A). In the following study for each method we considered (1) required computational effort, (2) the scope of the problem being solved, (3) feasibility of using the algorithm in real navigational situations, (4) type of allowed maneuvers (e.g., course or/and speed), (5) assumed safe distance between ships, (6) number of targets, and (7) type of constraints. The following algorithms were included in the study:

- *Indispensable maneuver.* The algorithm returns a single change of the course and/or the speed of the own-ship for avoiding collision in a multitarget encounter. Some assumptions made by the method include constant course and speed of all targets. The algorithm does not include any static constraints, i.e., it assumes an open sea scenario: The maximum number of targets is 20. The algorithm introduces an additional parameter, the so-called *delay time*, which allows approximating the dynamic behavior of the own-ship. The method was tested by executing computer simulations (the average computational time was $t_{av} = 15$ seconds) and in a real environment (instruction ship of Gdynia Maritime Academy; area: Bay of Gdańsk, Poland).
- *Utilization of potential collision threat area.* Due to simple graphic of the potential collision threat area (PCTA), it is possible to display it on a monitor of the anti-collision system in a real time. Thus the operator can make the decision on the basis of the simulation process (which assumes also constant course and speed of all targets). The size of the PCTA area depends on the assumed safe passage distance between ships. A large number of targets may complicate the displayed output, as a PCTA area is displayed for each target. As in the previous method, the algorithm returns a single change of the course and/or the speed of the own-ship. It also assumes open seas scenario. The method was tested by executing computer simulations. The average computational time was $t_{av} = 35$ seconds. This method was included as an option in the system ARPA 2000 produced by *Radwar* in Warsaw, Poland (1989–1991).
- *Indispensable trajectory.* The algorithm returns a safe trajectory as a series of maneuvers (changes of the course and/or the speed of the own-ship). As with the previous algorithms, it assumes an open seas scenario. The algorithm can be perceived as extension of the "Indispensable maneuver" method, as it generates (apart from collision-free trajectory) an additional trajectory for returning to the ship's original course. A safe distance D_{safe} is another parameter of this method; usually $0.5Nm \leq D_{safe} \leq 3Nm$ (it depends on the sailing conditions: weather, number of targets, etc). The method was tested by executing computer simulations (the average computational time of the method was $t_{av} = 4$ minutes) and during a test trip on the ship

m/v *Inowroclaw*.

• *Optimal trajectory.* The algorithm returns a safe trajectory as a series of maneuvers (changes only of the course of the own-ship). As opposed to the previous three algorithms, it can be applied not only to "open sea" scenarios, but can include static navigational constraints (e.g., shore line). The method models the problem as a nonlinear programming task with constraints; the solution trajectory is optimal with respect to moving targets and static constraints (a safe distance is preserved and minimal deviation from the original trajectory is guaranteed). The targets are modelled as moving circles; their radii depend on the safe distance D_{safe}. The method was tested by executing computer simulations (the average computational time was $t_{av} = 10$ minutes) and on *Norcontrol* simulator.

• *Evolutionary trajectory.* The algorithm returns a safe trajectory as a series of maneuvers and it can be applied to scenarios with static and dynamic constraints. Each change of a course segment may have its own speed: The computed trajectory may take into account targets' changes of course and/or speed. Safety conditions are defined by shape and size of ship's safety area. The method was tested by executing computer simulations (the average computational time was $t_{av} = 40$ seconds) and on a *Norcontrol* simulator.

The evolutionary method is a clear winner because of its generality and a reasonable computational time. Recently this evolutionary method was applied in real scenarios. An interesting comparison was made between the recommendations of evolutionary algorithm and actions of an experienced captain. This comparison test indicated that the solutions were almost identical. However, it appeared also that evolutionary system was more flexible: It controlled the movement parameters of targets online, constantly adapting the solution trajectory to the current situation. Figures 17, 18, and 19 display how a collision situation was handled by a captain and evolutionary algorithm, respectively.

The introduction of additional elements to the current version of the system (to include other environment changes) does not impose any significant problems in the evolutionary path planning, and, undoubtedly, makes the process more similar to real navigation situations. Also, newly unfolding situations can be, in a natural way, incorporated into the evaluation function, and the solutions adapt to the changes. Further research in this area is directed at estimating the ship trajectory when the moving strange-ships change the parameters of their motion in an unforeseen manner while the own-ship moves along its trajectory.

It seems also that construction of a hybrid decision support system for choosing a proper maneuver may be a step in the right direction. In such a system, the problem of avoiding collisions might be divided into two stages. During the first stage, a collection of safe maneuvers can be found quickly (for example, by the indispensable maneuver algorithm; see section II-A). These imperfect (but feasible) solutions can constitute the initial population for the evolutionary ϑEP/N++ system, which would return a near-optimum solution. The influence of the initial population (e.g., its feasibility) on the computational complexity of this evolutionary algorithm and the quality of the results, is also the subject of further studies.

Note also that the current version of ϑEP/N++ assumes that a constant speed is assigned to each segment of a trajectory, and that the own-ship can change speeds instantly when it moves from one segment to the next. Thus it is necessary to address the issue of *realizability* of a given trajectory, taking into account specific characteristic of the own-ship (this problem is common to all planning methods). An algorithm for checking the realizability of a trajectory was discussed in [11], [18], where a real trajectory is generated on the basis of a given one and the characteristics of the own-ship. This algorithm will be incorporated into the next version of the system. An additional issue that was not discussed in the paper is connected with the precision of control equipment and the sensitivity of a solution on measurement errors.

Acknowledgments

The research reported in this paper was partially supported by the grant 8 T11A 004 14 from the Polish State Committee for Scientific Research and the ESPRIT Project 20288 Cooperation Research in Information Technology (CRIT-2): "Evolutionary Real-time Optimization System for Ecological Power Control." The authors wish to thank anonymous reviewers for their useful comments.

References

[1] B.A. Colley, R.G. Curtis, and C.T. Stockel, "Maneuvering times, domains and arenas," *Journal of Navigation*, vol.36, no.2, pp. 324–328, 1983.

[2] B.A. Colley, R.G. Curtis, and C.T. Stockel, "A marine traffic flow and collision avoidance computer simulation," *Journal of Navigation*, vol.37, no.2, pp. 232–250, 1984.

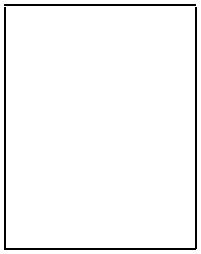
[3] P.V. Davis, M.J. Dove, and C.T. Stockel, "A computer simulation of multi-ship encounters," *Journal of Navigation*, vol.35, no.2, pp. 347–352, 1982.

[4] T. Furuhashi, K. Nakaoka, and Y. Uchikawa, "A study on classifier system for finding control knowledge of multi-input systems," in *Genetic Algorithms and Soft Computing*, F. Herrera and J.L. Verdegay (Eds), Physica-Verlag, pp. 489–502, 1996.

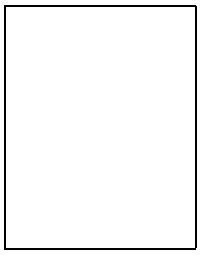
[5] *International Maritime Organization Preference Standards for Automatic Radar Plotting Aids (ARPA)*, Resolution A. 422 (XI), Nov. 1979.

[6] M.K. James, "Modeling the decision process in computer

simulation of ship navigation," *Journal of Navigation*, vol.39, no.1, pp. 76–84, 1986.
[7] K.D. Jones, "Decision making when using collision avoidance system," *Journal of Navigation*, vol.31, no.2, pp. 173–180, 1978.
[8] W.G.P. Lamb, "The calculation of marine collision risks," *Journal of Navigation*, vol.38, no.3, pp. 365–374, 1985.
[9] J. Lisowski, "A simulation study of various approximate models of ships dynamics in the collision avoidance problem," *Foundation of Control Engineering*, vol.10, no.2, pp. 176–183, 1985.
[10] J. Lisowski and R. Śmierzchalski, "Assigning of safe and optimal trajectory avoiding collision at sea," in *Proceedings of 3rd IFAC Workshop Control Applications in Marine System*, Thor I. Fossen (Ed), Trondheim, Norway, pp. 346–350, 1995.
[11] J. Lisowski and R. Śmierzchalski, "Methods to assign the safe maneuver and trajectory avoiding collision at sea," in *Proceedings of 1st International Conference Marine Technology*, T. Graczyk, T. Jarzebski, C.A. Brebbia, and R.S. Burns (Eds), Szczecin, Poland, pp. 495–502, 1995.
[12] W. Lord, *A Night to Remember*, Holt, Rinehart, & Winston, November 1955.
[13] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, Berlin, 3rd edition, 1996.
[14] Z. Michalewicz and D.B. Fogel, *How to Solve It: Modern Heuristics*, Springer-Verlag, Berlin, 1999.
[15] R. Śmierzchalski, "The application of the dynamic interactive decision analysis system to the problem of avoiding collisions at the sea," (in Polish), in *Proceedings of the 1st Conference on Aviation*, Jawor, Poland 1995, Research Works of Technical University of Rzeszow no. 135 part 2, pp. 141-147.
[16] R. Śmierzchalski, "The decision support system to design the safe maneuver avoiding collision at sea," in *Proceedings of the International Conference on Information Sysytems Analysis and Synthesis*, Nagib C. Callaos (Ed), Orlando, USA, pp. 95–103, 1996.
[17] R. Śmierzchalski, "Multi-criterion modeling the collision situation at sea for application in decision support," in *Proccedings of the 3rd International Symposium on Methods and Models in Automation and Robotics*, S. Banka, S. Domek, and Z. Emirsajlow (Eds), Międzyzdroje, Poland, pp. 699–705, 1996.
[18] R. Śmierzchalski, "Analysis and synthesis of navigator decision support algorithms in collision situation at sea," (in Polish) submitted for publication, 1999.
[19] R. Śmierzchalski and J. Lisowski, "The process avoiding collision at sea as a non-linear programming task," in *Proceedings of 14th International Congress on Cybernetics*, Namur, Belgium, pp. 627-632, 1995.
[20] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive Evolutionary Planner/Navigator for mobile robots," *IEEE Transactions on Evolutionary Computation*, vol.1, no.1, pp. 18–28, 1997.



Roman Śmierzchalski graduated in 1979 from Technical University of Gdansk in Poland, Electrical Department specializing in ship automatics. He received the PhD degree from Technical University of Gdansk, Shipbuilding Institute in 1988. He has worked as lecturer in Gdynia Maritime Academy. He worked as a electric officer on ships belonging to Polish Ocean Lines, and also in foreign companies: Greek and German. Moreover, for a year he worked on the "Dar Mlodziezy" sailing vessel, also as an electrical officer. He took part in research connected with the development of an anti-collision system where, in particular, he developed algorithms of mathematical estimation of maneuvers. Recently he was a chief manager of grant entitled "computer methods of safe control of a ship motion" financed by the Polish State Committee for Scientific Research. He has the authorities of the naval rank examiner in the range of naval automatics, conferred by the Marine Office in Gdynia.



Zbigniew Michalewicz is Professor of Computer Science at the University of North Carolina at Charlotte. He completed his MSc degree at Technical University of Warsaw in 1974 and he received PhD degree from Institute of Computer Science, Polish Academy of Sciences, in 1981. His current research interests are in the field of evolutionary computation. He has published several books, including a monograph (3 editions), and over 160 technical papers in journals and conference proceedings. He was the general chairman of the First IEEE International Conference on Evolutionary Computation held in Orlando, June 1994. He has been an invited speaker of many international conferences and a member of 40 various program committees of international conferences during the last 3 years. He is a current member of the editorial board and/or serves as associate editor on 9 international journals (including IEEE Transactions on Evolutionary Computation). Recently he published (together with David B. Fogel) a new text on modern heuristic methods [14].

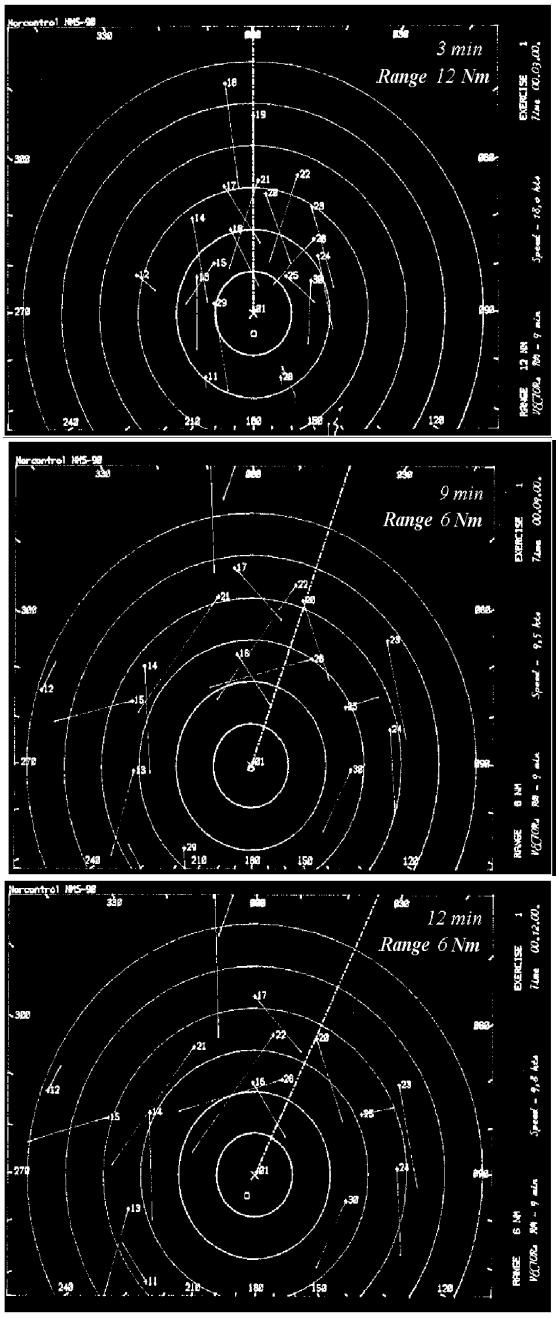


Fig. 17. Stages in relative move of a solution of collision situation (number of targets = 20) proposed by an experienced captain on a *Norcontrol* simulator (01 corresponds to the own-ship, targets relative vectors of speed = $9minutes$, $D_{safe} = 1Nm$). The figure in the top shows that the targets 16, 18, 19, 22, and 26 constitute a collision threat, as their speed vectors cross the dangerous area of the own-ship 01 defined by a circle of radii $D_{safe} = 1Nm$. To avoid a collision, the captain makes a right turn, changing the course of the own-ship to 025 (figure in the middle). Next stage (displayed in the bottom figure) demonstrates phase of the maneuver after 12 minutes. Figure 18 displays further stages of this collision situation

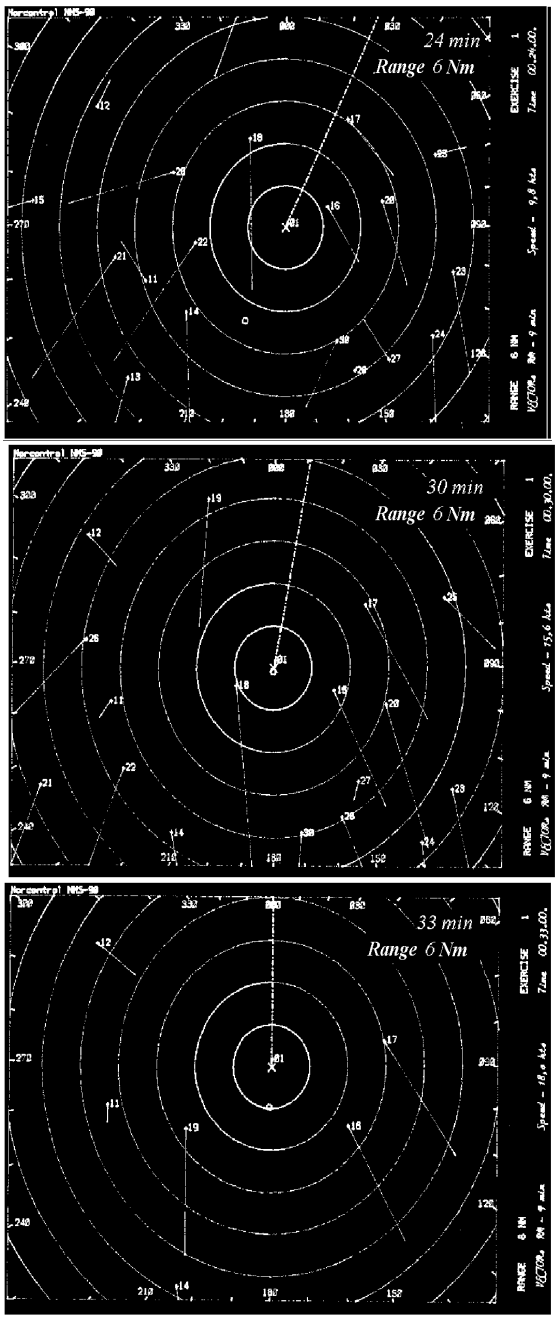


Fig. 18. Continuation of Figure 17. These three stages (top, middle, and botom figures) demonstrate phases of the maneuver after 24, 30, and 33 minutes, respectively; in the last figure the own-ship is back on its original course

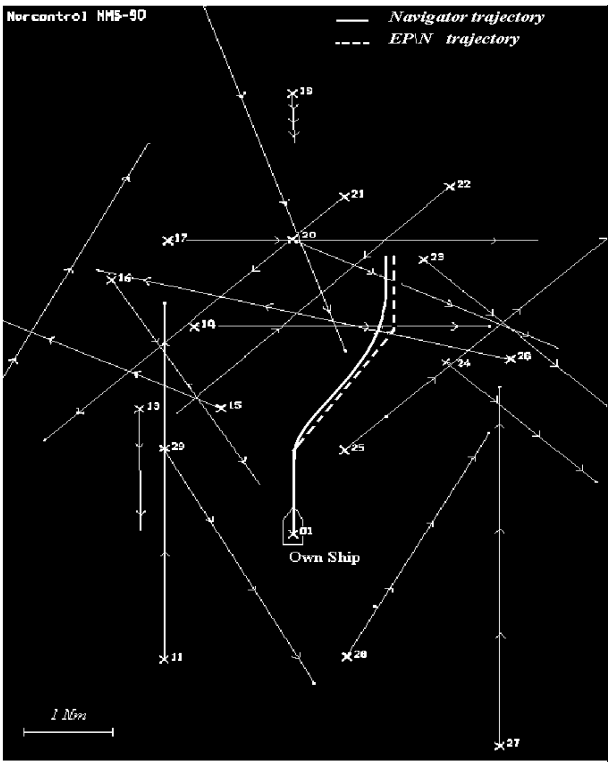


Fig. 19. Real-time solutions for a collision situation from Figure 17 proposed by an experienced captain (continuous line) and by evolutionary algorithm (broken line) ($D_{safe} = 1Nm$). Both solutions recommend a "right turn"; they differ only a little