

GASPS — Genetic Algorithm Search Path Simulator

M. A. Bednarczyk^{1,2,3}, Z. Kitowski⁵, M. Piotrowski⁴, A. Przybyszewska^{2,3},
T. Puźniakowski^{2,3}, J. Pyrchla⁵, A. Siekielski^{2,6}, J. Sławiński²,
and S. T. Wierzchoń^{1,3}

¹ Institute of Computer Science, Polish Academy of Sciences, Sopot, Poland

² Polish-Japanese Institute of Information Technology, Warszawa, Poland

³ Institute of Informatics, University of Gdańsk, Gdańsk, Poland

⁴ Hydrographic Support Squadron of The Polish Navy, Gdynia, Poland

⁵ Polish Naval Academy, Gdynia, Poland

⁶ Innovative Data Analysis Company, Gdańsk, Poland

Abstract

The paper describes preliminary findings developed in the course of a project which aims to provide optimized solutions to the problem of castaways search path generation.

The theory underlying today's standards is rooted in the 2nd World War and based on probabilistic analysis. Our aim is to explore the space of the search paths with genetic algorithms. Correspondingly, the practical goal of the project is to provide implementations that can be used by practitioners. A prototype of such tool called GASPS has already been created in collaboration with the Polish Naval Academy.

This paper we briefly describes theoretical assumptions underlying our approach, specifies potential chromosomes representation, as well as applied genetic operators.

Keywords: genetic and evolutionary algorithms, search path, search-and-rescue

1 Introduction

The starting point of our studies, see (Pyrchla, 1999; Bednarczyk and Pyrchla, 2000), into the principles of the naval search and rescue (abbr.: SAR) planning was a realisation of a gap.

On the one hand there is an old theory which underlines the best practices in the area, as codified in internationally accepted standards, see (IAMSAR, 1999). This theory is rooted in WWII and based on a probabilistic analysis. It seems unavoidable that any general theory of that kind has to be based on many simplifying assumptions. One of them, still reflected in the IAMSAR manual, is that the position of the searched object at the time of the accident, called *datum*, is known.

This assumption is sometimes satisfied, e.g., when the distress signal is issued by a mechanical device. In practice, however, the data received by the rescue

coordination authorities is often unreliable or missing. Quite often it comes from human observers who witnessed some signals of the accident. Therefore any analysis of such observations should take into account the lack of precision and partiality of data. Thus, in practice the assumption underlying the theory are not satisfied, which demonstrates one dimension of the gap — the gap between the theory and practice.

The gap has also another dimension. Namely, to the best of our knowledge the development of the software tools supporting the planning of the SAR missions is lagging behind. Most known software tools do not amount to much more than the calculators supporting the procedures described in (IAMSAR, 1999). Thus, they do not cope with the practical consequences of having uncertain and partial observations around. One such consequence is that the search area that can be described on the basis of partial data is usually not regular.

The above, together with the presence of non-uniform water currents makes the standard theory hardly applicable and suggests that a more flexible approach to the generation of the search paths may give better results than the search paths advocated by IAMSAR. To test the above hypothesis an application written in Java and called Genetic Algorithm Search Path Simulator (abbr.: GASPS) has been constructed, see (Bednarczyk et al., 2005; Pyrchla, 2008; Piotrowski, 2009).

It consists of two major components.

1. A search path *generator*, which traverses the population of search paths, by randomly choosing some of them and applying radical and/or local modifications (see section 4). The “weak” individuals are then removed from the population. The whole process is iterated many times to generate the best possible search path.
2. A search path *evaluator*, which computes the efficiency of the search path calculating probability of detection using a large set of randomly generated castaways. The efficiency of the path is computed by means of a function which assigns to each prefix of the path the percentage of all castaways that have been located that far (see Fig. 5). Thus, one of the simplest criteria of judging a path is to consider the value of the function at the end of the path.

The representation of search paths is described in section 2 and the initial population in section 3. The modifications of search paths based on the genetic paradigm are discussed in section 4, while the process of evaluation of a search path is described in section 5.

2 Search path as a *chromosome*

“Representation is a key issue in genetic algorithm work because the representation scheme can severely limit the window by which the system observer its world” (Koza, 1989, p. 4).

GASPS solves problem of finding optimal path, using evolutionary algorithm approach. The paths are possible solutions, they represent routes which would be followed by rescue boat. We use a simple and natural representation of a path, by an array of nodes (points), each of them being a pair of two coordinates. In GA’s

terminology, the sequence of points corresponds to *chromosome* or *individual*, and each node to a *gene*. Every two adjacent points create a vector (see Fig. 4). The time-interval between two consecutive nodes is assumed to be constant. Thus, a path in this representation corresponds to a sequence of vectors, one attached to the top of another.

Alternative representations of search paths currently investigated are described in section 7.

3 Initial population

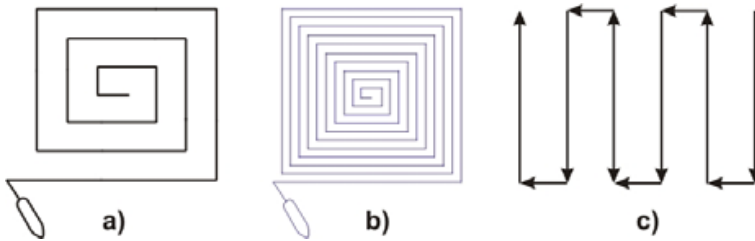


FIGURE 1: Standard search paths.

Initial population of chromosomes is generated in rectangular area selected by a user (see Fig. 2).

In the current implementation the paths used which live in the initial population are taken to be the traditional naval search paths. There are two kinds of them (see Fig. 1):

1. Expanding square spiral (a and b).
2. Parallelograms (c).

In both cases, specimens are created by three types of scaling:

1. The smallest ones, that cover about 70% of the blue rectangle.
2. Medium - they cover about 100% of the blue rectangle.
3. Big ones - they are about 130% of the area selected by user.

The initial population consist of a variety of specimen spanning each kind and each size all around the rectangle shape provided by the reader.

4 Genetic modifications of search paths

Due to the use of problem-specific solutions representation in GASPS, we also had to implement special genetic operators to modify them. We had to analyze the effectiveness of various *mutation* and *crossover* mechanisms that can be used in our case.

In areas where the genetic algorithms (GA) are traditionally applied, such as *the traveling salesman problem* (TSP), there are dozens of proposed mechanisms

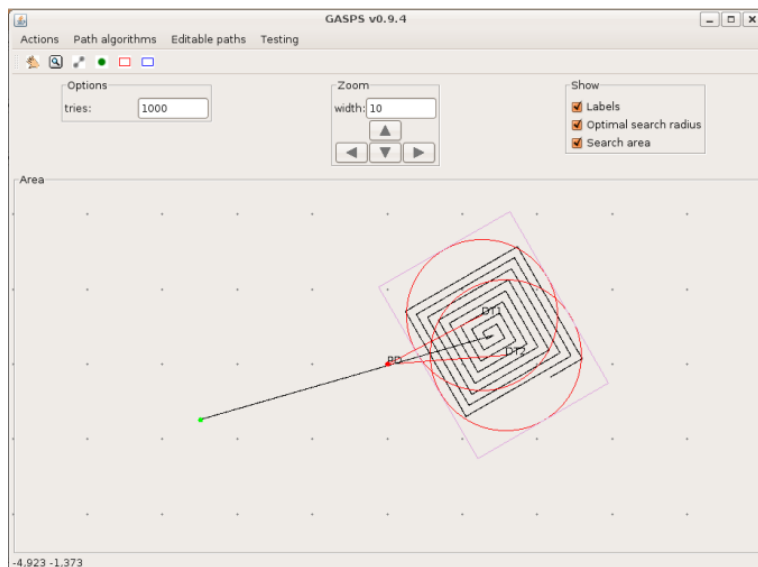


FIGURE 2: Expanding square shaped search path, and the user interface.

(Kuijpers *et al.*, 1999). To the best of our knowledge the area which we have at hand has not been tackled by means of GA. What sets our problem apart from TSP for instance is that in our case the solution space is continuous. One could even say that what we face is the problem of optimal cover of a 2-dimensional area with a 1-dimensional path. What is more, some of the theoretically perfect modification mechanisms may be no-realizable in practice. For instance, a ship may not change its course by 90° instantly.

4.1 Mutation

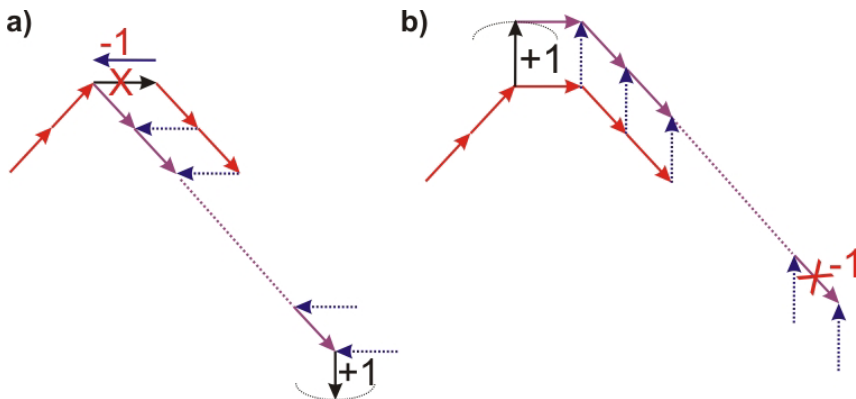


FIGURE 3: Mutations: a) removing one gen (vector); b) adding random gen

We use two kinds of mutation: removing or adding a vector at random position (i). In both cases, the rest of the path (starting with position i in case of removing or $i+1$ in case of adding a node) is moved in the way that it begins in the i th or ($i+1$)th node. Then, because our path's length must not change, we have to add one vector at the end of it or remove the last one (depending on the operation that has been performed). Both kinds of mutation are illustrated at Fig. 3.

4.2 Crossover

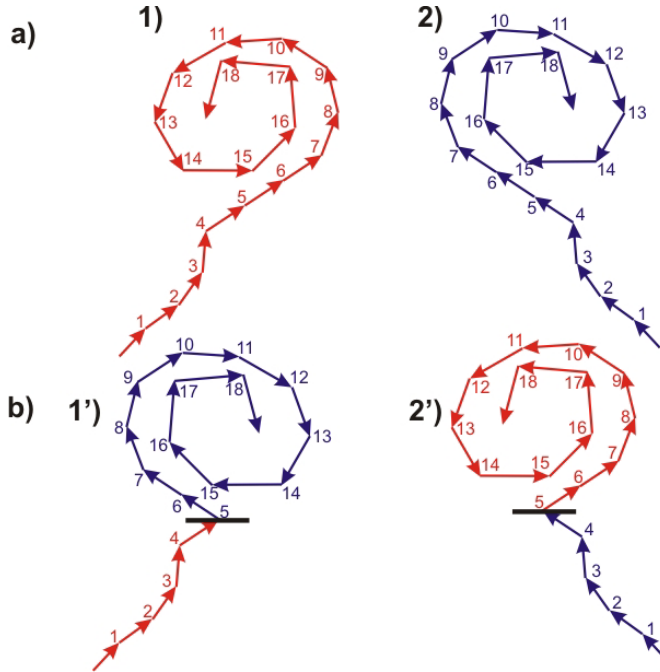


FIGURE 4: Crossover operator: a) parent chromosomes; b) pair of their offspring. Crossing point was randomly chosen at 5th gen.

As in standard GA's, we choose crossing point randomly. Then we split both paths at this point and stick parts coming from different parents together (see Fig. 4). We also had to implement some repair mechanisms in case of producing incorrect paths (for example, when we mate two paths with opposite directions). The simplest solution is to insert than one additional element at the crossing point.

5 Path evaluation

Evaluation of solutions in GASPS is not a trivial issue as we don't have any fitness function typical of evolutionary algorithms. The first idea to deal with this problem was to create a *simulator* which would be capable of feigning the process of searching for an object along a given search path. The simulator can

then be used to provide a probabilistic account of the quality of the search path with respect to a given search area.

Main steps in simulation process:

1. Generate a large number of the search objects (castaways) with respect to probability density associated to the search area.
2. Run the simulator on the path and on each castaway object (the same set of objectset is used for every path evaluation).
3. Compute the efficiency of the search path as a compound average of all simulation results. Specimen fitness is a number of successful¹ simulations.

During the simulation process, each castaway is moved according to following rules together with the searching object:

1. Castaway speed is fixed at start of simulation and is constant. Speed is calculated as sum of total water current and leeway. We also include water current and leeway error.
2. Castaway direction is also computed at the beginning of simulation and fixed at total surface drift direction. During each step of simulation there is also added random deviation, calculated as follows:
 - Decision whether deviation would be towards left or right is set at the beginning of simulation and stays constant during simulation.
 - Deviation degree is a small random value.

The efficiency of the search path is represented as a function which assigns to each node of the path the percentage of all searched object that have been located that far (see Fig. 5). Thus, one of the simplest criteria of judging a path is to consider the value of the function at the end of the path. In SAR this value is known as the *probability of detection*, (abbr.: POD).

Another possibility that can be easily implemented is to associate higher scores with the initial fragments of the search path — this would provide higher scores to paths which result in finding the searched object faster. This could be more appropriate, for instance when searching for castaways in cold water when the time is important.

The above procedure can be applied to arbitrary path to test how, on average, it copes with a given problem. Recall that a problem is represented here by a set of search objects, each encoded together with its own drift scenario. The accuracy of the efficiency approximation of the path returned by the simulator depends on the number of tests performed — in practice we talk about hundreds at least. Thus, the existence of the simulator provides computational definition of the *fitness function* indispensable for applying selection process fundamental for evolution based systems.

The tests performed so far suggest that the paths generated by GASPS tend to be better than the ones created by standard algorithm. An example of this phenomenon can be seen on Fig. 5. The function describing the efficiency for the standard algorithm can be seen in the window in the top left part of the screenshot

¹The castaway has been found.

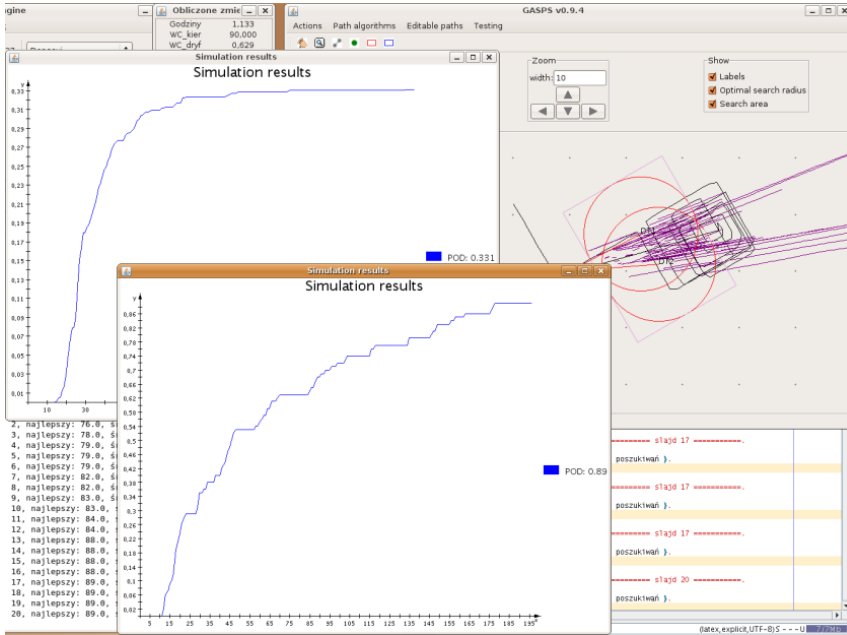


FIGURE 5: Search paths evaluations. The chart appears after simulation. It displays the number of ticks on vertical axis, and the number of simulations finished with success up till then on horizontal axis. There is also displayed the path’s POD.

titled „Simulation results”. The fitness function of the path found by genetic algorithm is in the other window titled „Simulation results”. In this particular test the standard algorithm reached POD around 33%. The paths generated by GASPS hovered at almost 90% POD.

6 Verification of simulator correctness

A crucial feature of every program is its correctness. Especially, if we are talking about software used to rescue people’s life. Even a small bug in the program may lead to severe consequences, including someone’s death. GASPS is an example of such an application. The search paths generated by the program must be valid. Moreover, these paths should be also optimal not only in the sense of minimalization of the detection time (which is of course the most important feature), but also in its simplicity.

In this section we discuss the correctness of computing the value of POD (*Probability of Detection*) by the GASPS’ simulator.

6.1 Assumptions

To make the task simpler several simplifying assumptions were made before the validation process:

1. The uniform probability model is used both when placing the castaway on the plane and when checking if the rescue team can see the object
2. The castaway is not moving during the simulation time
3. The simulation tick is large enough to avoid covering one area more than twice

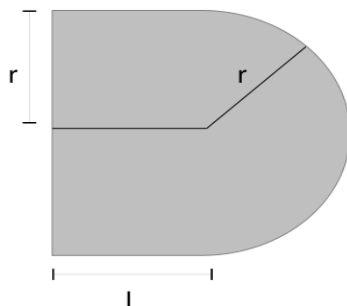


FIGURE 6: One tick's covering area.

6.2 Covering area

Let r be a visibility range of the detecting object, and let l be the distance made by the detecting object in one tick's time. The covering area of the one tick is shown in the Fig. 6. The path made by the detecting object is shown as a solid line. The object is moving from left to right, so there is no coverage on the left of the start point.

6.3 Search area

Let a and b be the dimensions of the search area. The example search area with the search path and covering area is shown in the Fig. 7. The distance made by the detecting object in one tick's time is equal to the visibility range of the detecting object. It is also equal to the $\frac{a}{n}$, where n is the number of the ticks (16 in this case). The path made by the detecting object is shown as a solid line. The area searched only once are marked by the light gray color. The dark gray areas are searched twice.

6.4 Probability of Detection

Let p be the probability of finding a subject when it is in a visibility range. We will be performing a search like shown in the Fig. 7. Our goal is to compute the Probability of Detect. The situation is simple, when $p = 1$. Then the computations are straightforward:

$$POD = \frac{2ar}{ab} = \frac{2r}{b} \quad (1)$$

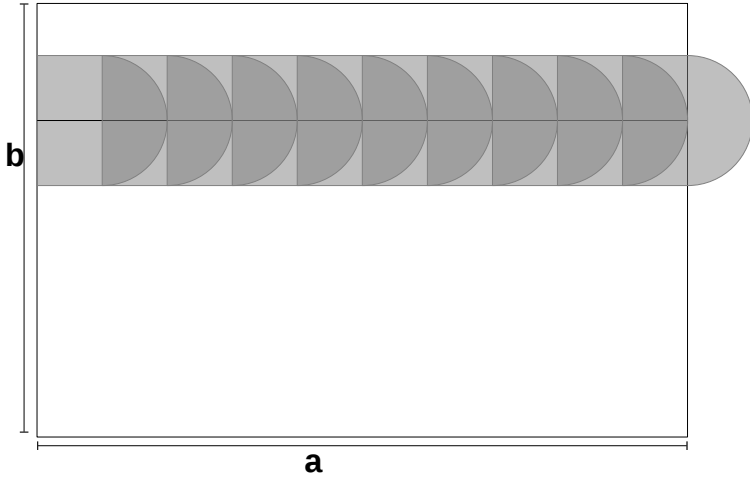


FIGURE 7: The example search area with covering area, where $l = r$.

In case when $p < 1$ some areas are checked twice, when first try gives a negative result.

Let n be the amount of ticks. POD when the area is searched only once can be expressed as a simple generalization of 1:

$$POD_1 = \frac{2rp}{b}$$

The case of the second searching can be expressed as a conditional probability:

$$POD_2 = \frac{p(1-p)\pi\frac{r^2}{2}(n-1)}{ab} = \frac{p(1-p)\pi\frac{a^2}{2n^2}(n-1)}{ab} = \frac{p(1-p)\pi\frac{a}{2n^2}(n-1)}{b}$$

So the final equation is:

$$POD = p\frac{2r + (1-p)\pi\frac{a}{2n^2}(n-1)}{b} \tag{2}$$

6.5 Simulation experiment

The simulations were performed with the following settings: $a = 6\text{NM}$, $b = 4\text{NM}$, $r = 0.6\text{NM}$, $n = 10$, and number of simulations: 100 000

The first run was performed with $p = 1$. The Simulator returned the following estimation of the POD.

$$POD = 0.30141 \tag{3}$$

On the other hand the equation (1) gives the following estimate.

$$POD = 0.3 \tag{4}$$

So the difference is less than 0.15%.

The second run was performed with $p = 0.345$. Application estimated:

$$\text{POD} = 0.15625 \quad (5)$$

And (from the equation 2) this should be:

$$\text{POD} = 0.15142 \quad (6)$$

So the difference is less than 0.5%.

Our tests show, that the GASPS simulator gives the results almost identical with the mathematical analysis.

7 New paths representations and genetic operators

Representation of paths as sequences of points are easy for evaluation, because every time we know where the moveable object is. It has some disadvantages, the most significant being the following:

1. Crossing over might break the correctness of paths,
2. The representation admits paths with unrealistic speeds.

That is why we consider alternative chromosome representations.

Currently we consider a representation that is both natural and easy to handle. The new path representation is based on angular and scalar velocities.

A path is still a sequence of n elements, each defining a part of it:

$$\text{Path} = [e_1, e_2, \dots, e_n]$$

i -th element is described by three values:

$$e_i = (\Delta t_i, (\omega_i, \nu_i))$$

where:

Δt_i - time in hours to cover given part of path.

ω_i - angular velocity of MO² in $\frac{rad}{h}$. If $\omega = 0$ MO moves straight.

ν_i - MO's velocity in $\frac{NM}{h}$.

Exemplary path is represented at the Fig. 8.

Path given in this way is easier for seamen to follow, as they use angular velocity to steer ships. Moreover it can be realized exactly as it is, as we take boat parameters into account while creating initial population and modifying paths. It is easier for us, as watercraft's capabilities are also given in form of quantities used for path elements description.

Genetic operations, such as mutation or crossover, can not destroy path's correctness, because the object movement is described by periods of constant derivatives. Mutations we consider are as follows:

- adding/removing one element (similarly as in the old representation)

²Moveable Object - object that will follow the path.

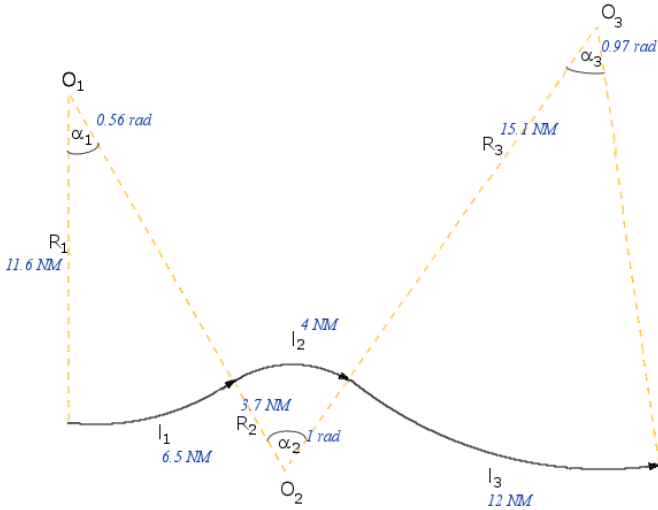


FIGURE 8: Exemplary path in the new representation: $Path = [(0.2, (2.8, 32.5)), (0.2, (-5, 18.5)), (0.4, (1.98, 29.82))]$

- modifying one element (random quantity or a set of random quantities)
- probabilistic mutation (as both mutations mentioned above, but it may occur for each element with rather low probability)
- swapping two elements
- combing: Replacing one element by short, standard path taken from patterns pallet. This should force searching some regions much more carefully.

Crossovers:

- as in the old representation: dividing two parent paths to two pieces at random point and than joining together parts coming from different parents, what gives two offsprings in result.
- multi-point: similar as above, but with a few crossing points
- uniform: for each gen in the offspring it is decided (with some probability) which parent will contribute it's gen in that position. If we want to have two offsprings, than the second one will receive a bit from the other parent.

We've also considered other path representations:

- $((\alpha, v), \Delta t)$
- $((\Delta\alpha, \Delta v), \Delta t)$
- $((\Delta\alpha, v), \Delta t)$
- $((x, y), "speedClass")$

We have made some theoretical work considering the problem, how easy is to break correctness in above path representations, but still our conclusions should be

approved by practical tests of algorithm operating on them effectively. That's one of the reasons for introducing HOG described in the next section .

8 GASPS in perspective

Genetic algorithm has some parameters which drive its behavior and convergence, some of them are numeric, and some other are problem-specific. The most important are:

1. crossover ratio - should be selected depending on problem³. Sometimes it is impossible to tell what value should be selected prior to some experiments.
2. mutation ratio - this also has great effect on algorithm. There are two major ways of mutating - by randomly changing fixed number of genes (usually 1), and by random checking each gene if it should be mutated (usually probability of changing particular gene is very small).
3. how to perform crossover - in classical genetic algorithm it is easy, because it operates on bits, but there are many other ways of representing genes, so sometimes it is possible to chose one from them.
4. how to perform mutation - sometimes there are many ways of mutating chromosome, this is different from mutation ratio, because it depends on chromosome representation.
5. selection method - the most popular is elitary selection, and it has also some parameters (like elite count).
6. representation of chromosome - sometimes there are several ways to represent solution.

There is another problem, how to select the best initial population, and how does this affect the results of genetic computations.

Obviously, we intend to have the most optimal parameters for our final application, because computation time in SAR operations is very important. Wrong parameters might cause program to waste computer resources and in limit it would not give any result at all. One way to cope with such a problem is to see it as an optimization problem of higher order. The problem can be seen as an optimization problem of higher order. We want to solve it using the mechanisms of GA themselves, and thus call the emerging Higher Order GASPS platform (HOG), see Bednarczyk *et al.* (2009).

The computation of a single search path with the help of GASPS when the number of iteration of the algorithm is limited to 50, and the number of test is limited to 100 takes several minutes on a modern PC. With the higher order optimization needed to compare various modification mechanisms one is forced to use GASPS as a basic simulation mechanism for the purpose of HOG.

We are going to provide an analysis of the effectiveness of various *crossover*, *mutation* and *selection* mechanisms that can be used in our case. To cope with this we are building a testing platform which allows us to compare the effectiveness of various mechanisms between each other.

³In classical genetic algorithm it should be higher, and in evolution strategy it should be lower, or particularly 0

In our institute we have a computation cluster consisting of five machines on which we are developing distributed architecture for this kind of computation. Our project is being developed in Java, and uses the idea of serialization to distribute computations among all machines. We called it DAC - Dynamic Agent Computation. This is different from other approaches, like MPI, because it sends classes, instead of only data, so computing many different instances of some program is more natural than having some points of synchronization, there is also possibility to use different versions of computation (for example 3 kinds of crossover).

9 GASPS in practice

The main purpose of our work is to use our ideas in practice. The possibilities and options are numerous. To mention just a few let us recall just two benefits of the approach.

First, the process of generation of the search paths takes into account changes in environment which have happened from the time of the accident until now. Intuitively, paths generated in this way should avoid searching non-interesting areas.

Second, our approach may cope with factors which were left untouched until now. In case of the Baltic Sea one such important factor is cold. Any rescue mission attempting to save human life should take into account that the chances of survival in cold water diminish quickly as the time passes by. In GASPS this issue can be addressed in a simple way. Namely, in the process of evaluation of paths done by the simulator the paths which succeed earlier should score high. One way to guarantee this is to provide a weight to a score which diminishes in time.

The authors understand that there is a huge distance between theoretically and computationally satisfactory solutions, and their successful implementation in practice. In the course of the project a number of practical tests will be performed. It is rather clear that a small number of tests cannot conclusively validate any framework such as GASPS — simply, the required number of such test would be bigger than those which we could afford.

Nevertheless, several such test envisaged in 2009 and 2010. The feasibility tests that started in 2008 aimed at verification of one practical issue. Namely, we have envisaged realisability of paths by fast boats where the process of following the path generated by GASPS was controlled via radio.

The result of one of the experiments can be seen on Fig. 9. One can see that the errors can be non-trivial. This observations poses practical problems for the application of GASPS in practice. We work on them at the moment.

References

- M. A. BEDNARCZYK, J. NEMANN, W. PAWŁOWSKI, A. SIEKIELSKI, and J. SŁAWIŃSKI (2009), Towards an Object-Oriented Framework for Higher Order Genetic Algorithms, in *International Conference on Artificial Inteligence*, to appear.

