1.

Unmanned Systems, Vol. 0, No. 0 (2014) 2–28

Minehunting Mission Planning for Autonomous Underwater Systems Using Evolutionary Algorithms

Nuno Abreu[a,b], Aníbal Matos[a,b]

[a] *INESC TEC, Campus da FEUP, Rua Dr. Roberto Frias, 378, 4200-465 Porto, Portugal*

[b] *FEUP - DEEC, Rua Dr. Roberto Frias, s/n, 4200-465 Porto, Portugal*
E-mail: nabreu@inescporto.pt, anibal.matos@inescporto.pt

Autonomous underwater vehicles (AUVs) are increasingly being used to perform mine countermeasures (MCM) operations but its capabilities are limited by the efficiency of the planning process. Here we study the problem of multiobjective MCM mission planning with AUVs. The vehicle should cover the operating area while maximizing the probability of detecting the targets and minimizing the required energy and time to complete the mission. A multi-stage algorithm is proposed and evaluated. Our algorithm combines an evolutionary algorithm (EA) with a local search procedure, aiming at a more flexible and effective exploration and exploitation of the search space. An artificial neural network (ANN) model was also integrated in the evolutionary procedure to guide the search. The combination of different techniques creates another problem, related to the high amount of parameters that need to be tuned. Thus, the effect of these parameters on the quality of the obtained Pareto Front was assessed. This allowed us to define an adaptive tuning procedure to control the parameters while the algorithm is executed. Our algorithm is compared against an implementation of a known EA as well as another mission planner and the results from the experiments show that the proposed strategy can efficiently identify a higher quality solution set.

Keywords: evolutionary algorithms; neural networks ; mission planning; AUV; 3D coverage; minehunting.

1. Introduction

Minefields are responsible for halting economic development by causing huge problems in worldwide shipping and, more importantly, causing injuries and death to humans. Nowadays the cost of producing and laying a mine is much less than the cost of removing it so there is a need to develop an accurate mine detection system that involves lower operating costs.[1] MCMs can be divided into two types: passive and active. Passive countermeasures involve reducing a ship's acoustic and magnetic signature to prevent mines from detecting it. Active countermeasures involve minesweeping or minehunting. Minesweeping is the clearing of a pre-defined area, whereas minehunting involves systematic detection and elimination of mines, one at a time. Sea minehunting has been typically conducted using systems towed from a surface craft or helicopters. One major step taken to improve these capabilities is the introduction of AUVs.

AUVs are able to perform underwater minehunting missions with higher efficiency, reduced search time and even with covert search capability when compared to traditional methods. The vehicle is launched, operated and recovered from a surface ship and carries mine reconnaissance sensors that are used to locate and identify mine-like objects (MLOs). In order to have such an important role in quickly and safely demining an area, the AUV needs to properly navigate through that area. Hence, a path needs to be carefully planned so all the area is sampled. There is an abundance of path planning algorithms in literature and, for this specific application, a small subset known as coverage path planning algorithms is usually used.[2] It is then understandable why the mission planning phase of traditional MCM operations is so important and often consumes as much time as the survey itself: a path needs to be found that maximizes the probability of detecting mines (therefore reducing risk) and minimizes the operating time and energy consumption.

EAs have been successfully used in the past for solving the multiobjective path planning problem.[3,4] This approach is an alternative to classical optimization methods with the capability of incorporating a variety of optimization goals, and is capable of solving the path planning problem involving a big search space quickly, although not guaranteeing that an optimal solution is found. One of its most important features is their capability of finding a distinct set of best solutions, optimizing each of the objectives in a different manner. The analysis of each of these solutions in terms of trade-offs provides a better understanding of the problem to the decision maker.

Coverage path planning (CPP) consists of determining a collision-free path that allows a robot to pass over all points of an area of interest. Since the robot must pass over all points in the workspace, the CPP problem is related to the covering salesman problem (CSP), a variant of the traveling salesman problem (TSP) where, instead of visiting each city, an agent must visit a neighborhood of each city.[5] It is essential to many robotic applications, such as demining, vacuum cleaning,[6] automated painting,[7] automated field operations,[8] window cleaners[9] and bathymetric surveys.[10]

A considerable amount of research addressing the CPP problem can be found in the literature. Choset [2] presented a survey on the field. CPP algorithms can be classified as heuristic or complete depending on whether or not they guarantee complete coverage of the free space.[2] Additionally they can be classified as either offline or online. Offline algorithms rely only on stationary information, and the environment is assumed to be known in advance. In offline algorithms all information about the terrain is previously known and assumed stationary and path planning is done entirely computationally. After the path is planned, the robot moves along the planned path until the end of the mission. In online algorithms, some or no prior knowledge about the environment to be covered is perceived previously. Sensors are needed in order to provide information about the environment, compensating for its absence. While the robot is navigating, the planned path will be updated accounting for the new information until the final position is reached. Although an online algorithm can plan for an unknown environment successfully, there is no guarantee that an optimal path will be obtained by using local information to guide path planning.

Next, a brief survey is presented covering some recent applications of CPP in the general field of robotics. Cheng et al. [11] presented an algorithm for time-optimal trajectory planning for an Unmanned Aerial Vehicle (UAV) executing 3D urban environments coverage. They developed a set of simplified coverage models to represent the urban features, converting the complicated 3D coverage problem into the problem of covering regular surfaces as hemispheres and cylinders. They also derived a lower bound on the time to achieve complete coverage of those surfaces assuming contant speed and designed an analytical motion plan whose coverage time is bounded by a constant factor times the lower bound. The designed coverage plan consists of a sequence of horizontal circles at different altitudes on the flight hemisphere. Each building was treated as an individual planning problem, ignoring nearby buildings. Their approach is validated using hardware-in-the-loop simulations involving a fixed wing aircraft.

Oksanen and Visala [12] proposed two greedy algorithms algorithms to find efficient 2D coverage patterns of agricultural fields. The necessary condition is to cover the whole field, and the goal is to find as efficient a route as possible. The first approach adopted the trapezoidal decomposition method and searched for the best driving direction and selection of subfields. The goal is to split a single field plot into subfields that are simple to drive or operate. The algorithm tries to iteratively decompose the field and remove the most efficient block (with the best cost), until the whole field is split. In order to find the best direction, the algorithm calculates the cost for a descrete set of directions, selects the best and refines the search by decreasing angle step size. The search continues until angle resolution is just below one degree. The drawbacks of the algorithm primarily concern the straight driving lines (caused by the decomposition). The underlying idea of second proposed algorithm is to follow the shapes of field edges and not to force them straight (following the shape of the edge or moving around the field as a spiral). The second algorithm is also an incremental algorithm, but the path is planned on the basis of the machine's current state and the search is on the next swath instead of the next subfield. Here, after each swath all the possible routes are simulated over a certain prediction horizon and the best of these is applied. This is continued until the whole field is covered by the operation.

Jin and Tang [8] developed an algorithm for 2D and 3D terrain field coverage path planning. The algorithm classifies the field terrain into flat and sloppy areas and then applies the most appropriate path planning strategy to each region minimizing headland turning cost, soil erosion cost, and skipped or overlapped area cost. The terrain characteristics have significant influence on the design and optimization of the coverage path. Elevation variations or slopes have considerable influence on soil erosion, skips and overlaps between furrows, and vehicle's fuel consumption. They also addressed terrain decomposition because the topography of a terrain field might have high variance between different areas making it difficult to find a single coverage pattern for the entire field. By decomposing the terrain into multiple subregions and calculating a different coverage planning pattern to each one, the coverage cost could be further minimized. A searching algorithm named "seed curve" was provided for the calculation of the optimized path. For each subregion, the final output of the planning algorithm is a set of curved paths that are side by side minimizing coverage costs. A recommended "seed curve" based on a customized cost function was searched for each subregion, and parallel coverage paths were generated by offsetting the found "seed curve" towards its two sides until the whole region was completely covered.

Xu et al. [13] presented an adaptation of an optimal terrain coverage algorithm for the aerial robotics domain. The general strategy involves calculating a trajectory through a known environment with obstacles ensuring complete coverage of the terrain while minimizing path overlap. The algorithm consists of partitioning the terrain into cells using the Boustrophedon Cellular Decomposition (BCD) method, and then generating an Eulerian circuit through all connected cells by solving a linear programming formulation. The Reeb graph, built using the critical points identified by the BCD algorithm, is used as input to the Chinese Postman Problem[14] to calculate an Eulerian circuit, which consists of a path traversing through every cell. Given an Eulerian circuit connecting all cells, each cell

can be covered by generating back-and-forth motions taking into account the kinematic constraints of the vehicle. The resulting trajectory is represented as a set of waypoints forming parallel sweep lines through the cell. The direction of coverage is defined as the orientation of the coverage execution as the UAV moves through the free space, which is orthogonal to all of the sweep lines. Extensive experimental results in simulation validated the presented system, along with data from more than 100 kilometres of real coverage flights using a fixed-wing aircraft.

In the underwater domain, coverage path planning research has its most important application in the context of MCM applications. Stack et al. [15] presented a 2D coverage algorithm for MCM using cell decomposition. They investigate a planning scheme for incomplete coverage. This scheme divides the search area into cells and surveys each cell using a line-sweep pattern with a row spacing that is larger than the sensor footprint, exploring the fact that mines are normally placed in lines. They assume that if mines are evenly spaced, then randomly varying the spacing between each row in the lawn-mowing pattern will decrease the probability of missing an entire mine line. A certain probability of detection (POD) is ensured by establishing bounds on row spacing. A perfect POD is assumed for any mine within the sensor footprint. Their coverage scheme aims at minimizing the size of the uncovered regions while keeping them distributed over the search area. The placement of the rows on the right and left sides of the area is random while the placement of the rows in the middle is a function of the estimated undetected mine locations.

Fang et al. [16] developed an algorithm for 2D offline global mission planning, involving the (boustrophedon) decomposition of the surveyable area into subareas using an approximation to the generalised Voronoi diagram, calculation of paths within subareas that allow for incomplete sonar coverage, and connection of subarea paths with transits to obtain a mission plan. They intend to cover a well-known planar seabed using an AUV fitted with side-looking sonar, maximizing area coverage rate. Paths within each cell were aligned with the edges of the cell produced by the decomposition process. They consider both even and uneven lawn-mowing coverage patterns but the spacing is fixed, proportional to range setting. Since they assume a planar seabed, they cannot consider a complex topography and therefore there was no need to implement variable spacing between the segments.

Williams [17] presented an 2D offline coverage algorithm for MCM that optimizes the spacing between parallel tracks in order to maximize POD, considering seabed type and range. It is assumed that the probability of detecting a mine, if one is present, improves to the maximum of any single view of a given location. They consider a scenario where an area must be surveyed to a certain detection level, regardless of the amount of energy and time required to execute a specific mission plan (they assume that the AUV has sufficient energy). Their track-spacing algorithm consists of an exhaustive greedy search for the best tracks. At each iteration, the gain in POD caused by every possible track is calculated and the track that maximizes it is chosen and added to the set of tracks to traverse. Because of the greedy approach employed, the set of selected tracks is necessarily ordered in terms of decreasing gain in average POD. To further improve these selected tracks a small geographical displacement of each track is considered. This process is iterative and executed until no improvement is observed. The main disadvantage of this greedy algorithm is that solutions are only locally optimal.

Das et al. [18] presented a 2D mission planner for robotic surveys in the ocean targeting the detection of harmful algal blooms. They use remote sensing (satellite and radar) to identify relevant bloom patches. After these areas are identified, an appropriate sampling path for the AUV is calculated. They use a lawnmower pattern with constant swath width and assume that the parameters of the bounding box that defines the survey area are predecided. Their goal is to attain maximum spatio-temporal sampling resolution at the regions of interest, thus maximizing the total signal intensity in that region (this measure is their sampling reward). To determine the location and orientation for a survey with a known layout and dimension they fit a bounding box to the identified region that has the maximum likelihood of an ongoing bloom. Then a survey path is generated within this area.

Yan et al. [19] developed a mission planner for oceanographic surveys which included a fault recovery architecture. The general idea of their algorithm is that some subareas need to be mapped using an AUV by performing a sonar survey following a lawnmower pattern. It maintains depth or altitude during the survey. They adopted a mission planning algorithm[16] to guarantee the complete coverage for each subarea, thus inheriting its disadvantages. In order to interconnect all the subareas while minimizing the length of the path, the authors formulated the problem as a prize winning salesman problem (PWSP) but no further information on the algorithm was provided.

Morin et al. [20] developed an offline coverage path planning algorithm for AUVs for performing MCM operations. They assume imperfect sensors and that multiple scans of the same area are independent. The POD depends on the seabed type of the surveyed region and on the range of the sensor. They use a cellular decomposition to represent the ocean floor by a grid of uniform square cells. The goal is to plan a path that achieves the minimal required coverage in each cell while minimizing the total travelled distance and the total number of turns. The robot can only move on the grid lines between the cells that represent the environment, therefore limiting the direction of movement of the vehicle to four directions. This is the big disadvantage of their algorithm. It is based on dynamic programming and on a travelling salesman problem reduction. The algorithm is composed of two main phases. In the first phase, a greedy technique constructs a set of disconnected vertical or horizontal segments such that a robot travelling along these segments will achieve the required coverage. In the second phase, they use a TSP reduction to optimally connect the segments obtained in phase 1, obtaining a feasible path of

minimal length.

At some point while the survey is being performed by the autonomous vehicle we may need to decide whether to continue with preplanned search paths or recalculate a new path in the presence of additional information. Over a long time interval, the benefits of adaptive autonomous search are overcome by performance of standard preplanned search paths.[21] If the goal of the mission is to find an object as soon as possible (as in search and rescue operations), it would be advisable for the vehicle to react to new information for a small interval of time. However, if the goal is to find all the objects in the area over a fixed interval of time then it is more beneficial to to maintain its initially planned paths for the duration of the search. Since many information is available a priori, an optimal path should be precomputed offline. Only if the information is found to be significantly different, while the mission is being executed and new data becomes available, should the path be efficiently modified. Although in this work we focus on optimizing the path offline, we present an interesting solution for online mission planning.

Paull et al. [22] developed an 2D online coverage method for robots equipped with side-looking sensors. This coverage method continuously directs the vehicle's heading using multiobjective optimization to maximize the information gain produced by the sensor measurements. The main advantages are that the AUV is able to maintain heading for better data mosaicing in the presence of currents or erratic waypoint tracking behaviour and it can adapt to changes in environmental conditions that can be detected in situ. This method is demonstrated in simulation and experimentally on an AUV conducting MCM operations. The results also show that the information gain approach alone is not sufficient as the mean path lengths are longer then in the cases using lawnmower tracks (common issue in greedy approaches). No information is given describing the methodology used to determine the set of weights in the utility function, which is an important process that dictates the quality of the final solution. Also, the optimal solution entirely depends on the chosen utility function, which is highly subjective.

This article consists of four sections and is structured as follows. Section 2 presents our multiobjective mission planning problem. Section 3 gives a brief introduction on multiobjective problems and discusses the applicability of EAs to solve them. Here we also provide a detailed description of the EAs that we designed to solve the problem under study and other relevant components. Section 4 presents and discusses the obtained results. Finally, in the last section we give some conclusions and ideas for future work.

2. The Mission Planning Problem

A typical mission is divided into three separate phases: the detection phase, the classification phase and the neutralization phase. In the detection phase, the location of potential targets is estimated using the data acquired by low resolution sensors that are able to sample a vast area. In the classification phase, the identified targets are revisited, possibly with higher resolution and low coverage sensors so a more accurate classification can be performed. In the neutralization phase, the MLO is literally neutralized, either by disarming it or by forcing it to detonate.

The most popular approach used for underwater mine hunting is performing complete coverage of the operational area. Complete coverage is achieved by ensuring that all the area is scanned by the robot's sensors. When conducting the detection phase with a sonar-equipped vehicle, a lawn-mowing search pattern with several parallel tracks is standard if no prior information on potential target locations is available.[23]

2.1. *Measures of mission performance*

In order to assess the effectiveness of a MCM operation we need to be able to estimate the detection performance that should be achieved in a specific mission.

2.1.1. *Lateral range*

The concept of a lateral range curve (LRC) was introduced by Koopman [24]. Imagine a searcher following an infinitely long, straight path, searching on either side of that path. Lateral range refers to the perpendicular distance an object is to the searcher's path, more specifically to where the searcher observes the object. That searcher's LRC $p(x)$ is the probability of detecting a stationary object that is at its closest exactly a distance x from the searcher's path. Figure 1 shows the relationship between coverage and probability of detection (POD) as derived by Koopman.

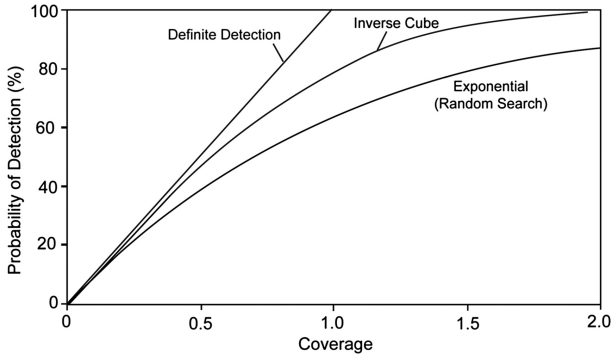


Fig. 1. Relationship between POD and coverage.

Theoretical lateral range curve

In order to calculate a map of POD in the operating area we need to be able to calculate the POD at any (x,y) point. Sonar equations are primary analysis tools for studying and

predicting sonar performance. Prior [25] presented a simple sonar model and derived a formula for calculating the POD from the beam origin on the sonar, considering sonar and environment properties.

Experimental lateral range curve

Target detection is influenced by many factors which may make the task of building an accurate model of the instantaneous detection unreasonable. The amount of data required to extrapolate a LRC directly is much less than that required to develop an instantaneous detection model. The LRC is derived experimentally by moving a sensor through an area where objects are randomly placed using parallel straight-line search transects. Development of LRCs requires a large sample set with many detection opportunities, where the object positions are known, in order to achieve a more statistically valid curve. Each point on the LRC represents the probability that a target will be detected from a specific closest point of approach range. Derivation of the POD versus lateral range data requires that the search results be processed to identify the range of every detection opportunity for each search object on each search transect.

2.1.2. *Probability of detection*

POD is an estimate of how likely it will be for a search performed in a given area to find an object, assuming it was there. POD measures sensor effectiveness, accuracy, and quality for the search task. It has become the de facto measurement used in search and rescue theory. It is known that an overlap in area coverage can improve the detection performance. The redundant information that is gathered helps to reduce ambiguity in a noisy environment. In this sense, we present another important measure of performance which is the cumulative POD. After covering a given area multiple times, the probability of detecting the search object, if it was present, should be increased as compared to having searched the area only once. However, searching the same area twice does not double the chances of detecting it. When it is assumed that multiple searches are executed independently of each other, the combined POD at lateral range x is given by:

$$PODcum = 1 - \prod_{i=1}^{n}(1 - POD_i) \tag{1}$$

The degree of correlation between the information gathered from two searches over the same area needs to have a direct impact upon the manner in which detection probabilities are combined. Here, we consider three different scenarios: no correlation, complete correlation or an indeterminate amount of correlation. When there is no correlation between the two searches, we use the formula presented above to calculate the cumulative POD. If assuming the existence of complete correlation, the combined detection probability is simply the higher of the two individual probabilities. When there is an indeterminate amount of correlation between the detection performance achieved in each search, an accepted practice[26,27] is to average the probabilities obtained assuming complete independence and complete correlation to estimate the combined detection probability. Modern research involving the calculation of the cumulative POD in different case studies assume complete independence, but there are some problems with this approach.[28] If we perform additional surveys using the exact same path in the same conditions (same sensors, detection algorithms, environmental conditions) we would acquire the same information (same path and conditions guarantees same perspective of the object) in each survey so the several PODs should not be combined since there is no information gain. Still, combining the probabilities should be a close approximation to reality since it is very difficult to replicate the exact same conditions in different experiments. The choice of methodology to be used to combine information from different surveys should also take into account the specific nature of the problem. Minehunting operations involve a danger or risk factor that makes the cost of overestimating the POD much higher than the cost of underestimating it. Ideally, we want to use a methodology that produces the most reliable estimates but, when in doubt, it might be beneficial to opt for a conservative estimate. These are obtained by assuming the existence of complete correlation, thus by choosing the higher of the individual estimates, which can be considered as the lower bound of the interval of admissible POD estimates.

2.2. *Covering with different types of sensors*

Now we present a brief description of the characteristics of conventional sidescan sonars and multibeam echosounders to understand how they can be used for detecting MLOs.

The operating principle of the sidescan sonar is based on forming an acoustic image by moving the sonar forward and compiling the sonar response from successive pings. A sidescan sonar uses transducers, one on each side, that emit fan-shaped acoustic pulses down towards the seafloor across a wide angle perpendicular to the path of the sensor through the water. They have several advantages, such as being available at a relatively low cost, being easily deployed on AUVs and their ability to produce acoustic images of the seafloor with high resolution making it capable of illuminating small targets such as the typical mines. All sidescan sonars suffer from an inability to sufficiently illuminate targets within the "nadir-gap" area. This is the part of the seafloor directly below the transducers that, because of the geometry of the sonar configuration, is under sampled.

The multibeam bathymetric sonar generate a large number of beams for each ping. It uses two or more perpendicular transducer arrays used to transmit and receive

the beams. Along each beam (or direction), the range (calculated from the time delay) to the seafloor is estimated. The range estimate in each beam and the angle of the received beam gives the relative height of the seafloor relative to the vehicle. By combining data from consecutive pings, a 3D map of the seafloor can be generated making them nearly optimal instruments for bathymetric terrain navigation. One disadvantage of this type of sonar is the decreasing angular resolution as a function of distance from the nadir resulting in decreased resolution in the outer swath.

In practice, the most relevant feature that makes it important for the mission planning algorithm to distinguish which type of sonar is used is the existence of the nadir gap in sonar coverage, as illustrated in figure 2. If it exists then a different technique should be used to optimize the coverage plan. The traditional approach to compensate for this property is to partially overlap consecutive swaths so the respective nadir gaps are covered, also known as uneven lawn-mowing.

The principal problem under study in this paper is how to design and implement a more flexible 3D path planning algorithm for the detection phase that enables an AUV to efficiently cover the bottom of a submerged area with no missed areas and with a specified minimum POD. The planner should identify a set of transects, representing sonar swaths, that maximize the estimated performance of a MCM operation, using the available knowledge and resources. Searching for a path requires the consideration of the environment characteristics (terrain topography), available resources (characteristics of the onboard sensors, available battery power), maximum time available for the mission and vehicle kinematic constraints. The planner should also return a map with the estimated detection performance for the survey area and the total amount of resource spent during the search. This leads to the formal statement of the mission planning problem.

2.3. *Decision variables*

Mission path

A mission path is represented by a set of consecutive straight-line transects also referred as swaths:

$$P = \{s_1, s_2, ..., s_n\} \tag{2}$$

The swaths represent the data acquired by the sonar while traversing the useful parts of the trajectory. When using the typical lawn-mowing coverage pattern, these swaths are parallel to each other.

Vehicle Velocity

The AUV will follow the specified path with a constant forward velocity relative to earth equal to v.

2.4. *Objectives*

Maximize probability of detection

One of the goals of the planner is to maximize the MCM effectiveness, which in this case is represented by the average probability of successfully detecting the target on the survey area:

$$max \sum \overline{POD_s}(S, P) \tag{3}$$

Minimize energy consumption

$$min\ E(P, v) \tag{4}$$

The amount of energy consumed by the vehicle depends on its velocity, orientation and the actual path. Both the energy consumed by the motors and the payload should be considered.

Minimize time to complete the mission

$$min\ T(P, v) \tag{5}$$

The time required to execute the mission depends only on the path and the component of velocity that is parallel to the track direction.

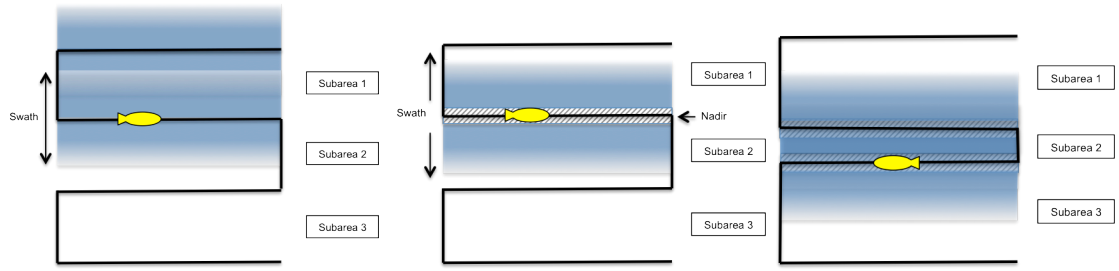
2.5. *Constraints*

The bounds on the optimizing parameters are presented in table 1.

Table 1. Design constraints of our optimization problem

Parameter	Type of constraint
Battery capacity (Wh)	vehicle performance
Percentage of energy available (%)	vehicle performance
Velocity relative to water (m/s)	vehicle performance
Maximum steering angle (^{o})	vehicle performance
Sonar maximum range (m)	sensor performance
Sonar vertical beam angle (^{o})	sensor performance
Maximum operating time (h)	mission requirement
Minimum POD (%)	mission requirement

Obviously the vehicle is confined to the workspace defined by the latitude and longitude of the survey area, the bottom floor and the sea surface. The terrain is considered to be static. It is assumed that there is sufficient knowledge with regard to the properties of the terrain in which the robot will operate.



(a) Coverage with a multibeam sonar (b) Coverage with a sidescan sonar (c) Uneven lawn-mowing technique used for covering the nadir gaps

Fig. 2. Different methodologies used in coverage problems.

3. Methods

3.1. *Evolutionary algorithms for multiobjective optimization*

In a multiobjective optimization problem, the goal is to find a set of different solutions representing distinct trade-offs among the different objectives. These solutions are known as Pareto-optimal solutions.[29,30] In problems with more than one conflicting objective there is no single optimum solution. There exists a number of solutions which are equally optimal and without any further information about the problem or about the decision makers preferences, no solution from this set can be said to be better than the other. This set of solutions can be found using the Pareto Optimality Theory.[29] The goal of multiobjective optimization is to find a set of solutions close to the Pareto-optimal solutions, diverse enough to represent the entire spread of the Pareto-optimal front.

The continuous improvement in hardware technology has allowed the use of EA with higher complexity, providing efficient means for addressing real world problems that traditional algorithms were unable to conquer. These techniques are based on the principles of natural evolution as presented by [31]. Evolution can be interpreted as adding new individuals to a population, sharing information of the fittest individuals. An EA consists of a population of candidate solutions, known as individuals, manipulated by a set of operators, called variation operators, and evaluated by a given fitness function. The fitness function determines how good an individual is and assigns a corresponding fitness value to the individual, which dictates his chances of survival. The calculation of the fitness values involves the evaluation of the considered objective functions.

Pareto dominance-based techniques remain the most popular selection scheme adopted by Multiobjective evolutionary algorithms (MOEAs), because of the several advantages that it provides. The SPEA 2 [32] and NSGA 2 [33] are two of the most popular MOEAs used when comparing a newly designed MOEA.

3.2. *Proposed EAs for mission planning*

Solving the path planning problem is computationally very expensive. The upper bound on the complexity of a n degree-of-freedom path planning problem is $O(n^n)$, meaning that the complexity of the path planning problem grows exponentially with the number of degrees-of-freedom.[34] Classical path planning approaches[10] are traditionally used to solve a single-objective optimization problem: find the shortest path between the initial and goal locations, using a representation of the environment. Planners with a shortest path objective may also implicitly satisfy other optimization objectives imposed by the specific representation of the environment that is being used. However, there is no classical path planning approach that can incorporate several different optimization criteria and handle changes in these optimization goals without changing the characteristics of the planner itself. In path planning, the shortest path may not always be the most efficient means of getting from start to destination. There are many other attributes of a path that may be desirable in addition to distance and ultimately it depends on the specific problem that is being addressed. Another disadvantage is its predisposition to getting trapped in locally optimal solutions. In some situations, it can be useful to have several alternative solutions to the path-planning problem rather than a single solution. For example, in dynamic environments, one or more of these paths may become infeasible, so one of the other feasible alternatives can then be chosen. Evolutionary approaches provide an alternative to classical functional optimization methods with the capability of incorporating a variety of optimization goals, and are capable of solving the path planning problem quickly, although finding an optimal solution is not guaranteed. The shortest path optimization objective is an important objective in the path planning domain but, realistically, it may not be the only optimization goal that is relevant. Real world problems may involve the optimization of a variety of complex and subjective objectives so it is preferable to develop an efficient and flexible planning methodology that can be successfully applied to different applications.

The preferred procedure to solve multiobjective optimization problems is to follow the preference-based approach, where the decision maker specifies his preference structure with respect to the multiple objectives. The ability of an EA to find multiple optimal solutions in one single iteration of the algorithm makes the EAs unique in solving multiobjective problems. Since the a posteriori preference articulation techniques for multiobjective optimization require multiple trade-off solutions to be found, an EA's capability of generating and handling a population of several individuals can be suitably used to find a number of solutions for the multiple-objective in a single iteration. As MOEAs have been successfully used to find good solutions for complex multiobjective problems in engineering, we decided to propose one MOEA to solve our mission planning problem. Our approach is a a posteriori technique, more specifically a Pareto-based approach, that on each iteration searches for a set of solutions near the true Optimal front. It shares the structure of the generic EA, but is adapted to solve our specific multiobjective path planning problem. The mission planning algorithm is an offline planner that should be executed before the vehicle is deployed in the target area.

3.2.1. *Individual representation*

In our EA, we consider an individual as a solution represented by a vector of real valued parameters $x \in R_n$. Thus, an individual is represented by:

$$Ind \in (t, \theta, d, a, v) \tag{6}$$

where t is the track spacing, θ is the track direction, d is the depth of the path (used if we are planning with constant depth), a is the altitude of the path relative to the sea floor (used if we are planning with constant altitude) and v is the velocity relative to the water that will be adopted by the vehicle while executing the mission. This set of parameters is needed to automate the procedure of generating different individuals in decision space. They also allow the comparison of high level proprieties of distinct individuals. These parameters are related to decision variables and therefore can be combined together to generate values for these variables.

3.2.2. *Standard EA*

Initialisation An initial population of individuals is randomly generated from a uniform distribution in decision space. When generating the different individuals we need to consider the boundaries of the parameters that define them. The total vertical distance available is calculated by subtracting a maximum depth (which considers the terrain's topography and the vehicle's minimum operating altitude) to a specified minimum operating depth. The values for the direction are bounded by 0 and 360 degrees. The values for velocity are bounded by the minimum and the maximum velocities defined for the specific vehicle being used. For the track distances, we decided to create a different set for each specific depth. The boundaries of these sets were calculated taking into account the maximal horizontal distance which is covered with the pre-specified maximum sonar angle, at a given depth. For each depth, the minimum distance between two tracks would be this single sided horizontal distance, while the maximum would be twice this distance. When separated by this maximum distance, two consecutive tracks will not produce overlapping swaths, constituting the upper bound of the set of track distances.

Evaluation The detection performance achieved by each individual is determined by calculating the cumulative POD (assuming no correlation between observations at the same location), a process that is detailed in 2.1.2, using a lateral range curve. The required energy to complete the mission is calculated by considering the average energy required by the AUV's onboard systems and the motors. The required time is calculated by considering the velocity of the vehicle relative to water and the current velocity at the operational area. Only individuals that present a detection performance level above the minimum and require a amount of time and energy to complete the mission below the pre-specified maximum are allowed to join the population.

Fitness Assignment Our algorithm adopted the fitness assignment strategy introduced by SPEA 2, which consists of calculating a strength value, a raw fitness value and a density value for each individual in the population. The strength value assigned to each individual represents the number of individuals dominated by him. The raw fitness value of an individual is determined by adding the strengths of the individuals that dominate him. In order to differentiate individuals with the same raw fitness value, density information is added to that value. The density associated with an individual is the inverse of the distance to the k-th nearest neighbour. By adding this density estimate to the raw fitness value, we obtain the fitness of an individual. We then filter the whole population by density by removing individuals from denser areas (closer then a specified minimum distance) in the objective space.

Environmental Selection Our approach is elitist and we do not implement a fixed archive or population size. We believe that this feature does not make sense since that could lead to a situation where we had clusters of individuals scattered around the search space. The objective of our evolutionary search is to produce the best possible Pareto Front, containing individuals uniformly scattered in space. Therefore, the number of individuals is not a parameter or a requirement. We instead decided to only control the density of individuals in the containers, which affects the number of individuals.

Mating Selection Our strategy for selecting the parents for the next generation population is to:

- mate individuals in the archive;
- mate individuals in the archive and the population.

The first action is achieved by combining properties of individuals in the archive which are close to each other in the objective space. We need the pair to be close to each other because we want to minimize the randomization effect that exists when we combine two different individuals. For the same reason, we mate an individual from the population with the closer individual from the archive, hoping that the offspring is closer to the known Pareto front.

Recombination Our main goal is, for each pair of individuals in the mating pool, to improve the strengths of the less fit individual by sharing attributes of the fitter. The procedure consists of identifying of the fitter individual, normalizing of the objective function values and improving the strengths of less fit individual. Each individual can have one strength: detection performance, energy consumption or time required to complete the mission. Several actions can be performed to improve the performance on the identified objective. To improve the detection performance we can change the depth, the track spacing or the type of fitting. To improve the energy consumption or the time required to complete the mission we can change the track spacing, the direction, the velocity or the type of fitting of the trajectory. Changing the direction can have a big impact when dealing with currents. Then we assign a probability for each of those actions. For example, if we intend to improve detection performance then we can assign the following set of probabilities: 40 % for changing depth, 40 % for changing track spacing and 20 % for changing the type of fitting of the trajectory. This means that we need to determine which action will be executed by obtaining a random number with a uniform distribution. The recombination is then executed by creating a new individual which is an evolution of the unfit individual, sharing the previously chosen attribute with the fit individual.

Mutation The mutation strategy is similar to the one used in recombination: improve the performance of an individual in the archive relative to the objective where he performs better by simply modifying a given attribute by a small amount. To avoid the loss of good individuals from the population, and to improve the speed of convergence of the EA, asexual reproduction is also used to copy the non-dominated individuals (in the archive) to the next generation.

Termination The execution of the EA stops when one or more pre-specified termination criteria are met:

- a pre-specified number of generations is achieved;
- the maximum time allocated for the process is achieved;
- the algorithm converges (no improvement over a several generations).

While the termination criteria is not met the algorithm resumes execution on the evaluation step.

3.2.3. *Informed EA*

Realistic and complex engineering problems, like the one we are discussing here, involve objective functions that can be very expensive computationally. Consequently, it is advisable to assess the integration of approximate models, which are usually orders of magnitude faster. These approximate models provide less-accurate but more efficient estimates of the original objective function values and can be either readily available or can be learned on-line (during the course of the optimization) or off-line (by sampling and building a model before optimization). This can be interpreted as the addition of a learning mechanism because the information acquired due to the interaction with the environment will allow the acceleration of the evolutionary process.

As a result, we decided to combine our EA with an approximation technique, more specifically ANNs. EAs usually need a sufficiently large number of function evaluations to reach the optima of the problem. By using an approximate model of the problem we are in fact reducing the number of expensive exact function evaluations. The main task of the approximate model is to capture the general trend of the fitness landscape and guide the search towards the better regions while reducing the number of exact function evaluations needed to find the best solutions. This guidance is executed by assessing the performance of multiple candidate individuals using the approximate models, with a much smaller evaluation time, and choosing only the best to integrate the next generation population. One of the challenges that arises is the need to update the approximate model using new information from exact function evaluations throughout the execution of the algorithm, adding finer details to the fitness landscape.

It is important to develop an algorithm that adjusts the training process to the performance of the neural network in an changing environment. If the data used for training is not representative of the decision space, then significant estimation errors may take place. This can easily happen if the training data is insufficient, which is common during the initialization phase of the EA as the number of exact function evaluations is very limited in order not to take too much time. As the number of generations increases, more exact function evaluations will be available covering different regions of the search space. Another difficulties are how to improve the generalization capability of the network and manage a continuously growing training data set without running out of memory or increasing the processing time beyond given limits (see subsection 3.7).

Initialisation The initial population is populated with N_p individuals randomly generated from a uniform distribution in decision space. These individuals are evaluated with the original objective functions since at this point the approximate model is still not available. It is the most important phase because it will create the first overview of the fitness landscape that the algorithm will use to guide the search and also because these exact evaluations will be used to train our approximate model. The choice of optimum value for the N_p parameter is critical for the success of the algorithm as higher values will create a more detailed picture of the fitness landscape at the cost of a longer processing time since all objective function evaluations are exact.

Neural Network Training The first phase of each generation is dedicated to neural network training. We implemented a classical multilayer feedforward architecture using the standard error back-propagation algorithm with sigmoidal activation function. Our ANN structure is characterized be receiving in the input layer six parameters that define an individual, generating the estimated objective function values at the output layer. Our EA keeps a history of all precise function evaluations performed to date and this information is used every k generations to train the approximate model. Since the amount of data increases each generation, there is a need to control this growth (see subsection 3.7).

The first time this phase is executed several distinct network topologies are compared. Since it is proven [35] that a ANN with two hidden layers can approximate any function with negligibly small error, we test networks with different number of neurons in each layer. As indicated by [35], we use as maximum number of nodes for the first and second hidden layers the output of the following equations, respectively, ensuring that the training data will not be overfitted:

$$N1_{max} = \sqrt{(m+2)N} + 2\sqrt{\frac{N}{(m+2)}} \tag{7}$$

$$N2_{max} = m\sqrt{\frac{N}{(m+2)}} \tag{8}$$

The minimum values for the neurons were obtained through experience: for the first hidden layer we chose twice the number of input neurons and for the second twice the number of output neurons. A third parameter needs to be used to iterate between all possible values, called $\Delta_{neurons}$, limiting the search space for the ANN structures. All the ANNs in the network container are trained and validated. In order to train the ANNs and validate the models we divide the set of precise function evaluations into two sets: a training set (80%) used for tuning the ANNs and a validation set (20%) used for performance evaluation. The training and validation steps performed followed common procedures, such as normalizing the data tuples used, starting the learning process from different random initial states (weights are initialized to random numbers between -1 and 1) and observing performance along the training iterations. The performance measure used was the mean square error (MSE) function. The learning process was controlled based on the behaviour of the MSE function on the training set. After this initial phase (in generation 1), ANNs are ranked by validation MSE and only the best n_{nn} are kept and updated, using the one with smaller validation MSE. Training stops when the specified maximum number of epochs is achieved, if the training error is smaller then the desired error or when it converges to an error bigger then the desired one. The first ANN to achieve the desired error is the one that is used as the approximate model. Therefore, the other ANNs in the container are only trained in the worst case scenario.

Fitness Assignment / Termination / Mating Selection Same procedure of the original EA.

Recombination Instead of the stochastic process used in the original EA, now all possible offspring (all attribute combinations of the parents) are evaluated using the approximate model. This could not be done before because we were using the original expensive objective functions, so we had to test one random combination of parameters per set of parent individuals. Like in the original EA, we try to improve the strengths of the less fit parent, i.e. the objective function that is closer to the current optimum. Then several possible combinations of path attributes are identified and evaluated using the approximate model. Next, these candidate offspring are ranked in terms of improvement over the less fit parent (considering the selected objective function). If there exists a candidate offspring that represents an improvement, then the best one is evaluated using the exact objective functions and added to the population of the next generation. The proposed method should significantly speed up the EA because we are able to test a huge number of candidate individuals almost instantly.

Mutation Now we evaluate several possible mutations from the individuals in the archive using the approximate model. Like in the original EA, we try to improve the strengths of the selected individual, i.e. the objective function that is closer to the current optimum. Basically, our mutation process consists of performing sensitivity analysis using the ANN, testing different variations of the original individual by modifying to some extent each of his parameters (ANN inputs) and analysing the corresponding estimated function values. Our algorithm initially identifies n_m randomly generated variations (e.g. 20 different values of depth in order to improve POD). A parameter named mutation magnitude (percentage) is introduced, specifying the maximum amount of variation that is allowed for each path attribute. All identified variations of the individual for all specified actions are then evaluated using the approximate model, ranked in terms of improvement over the

original individual, and then, if there is an improvement, the best one is evaluated using the original objective functions and added to the population of the next generation. This should improve the quality of the generated individuals because we are able to test a huge number of candidate individuals almost instantly.

3.2.4. *Evolutionary performance evaluation*

To properly evaluate and analyse MOEA performance, three different properties should be studied:

Convergence The evolution of the search process is assessed by analysing convergence. This is done by comparing sets of individuals from consecutive generations and determining the number of individuals that are dominated in each set.[36] In elitist algorithms, the archive at a generation is compared with an older archive. The number of solutions in the old archive that are dominated by the current archive and the number of older archive members that remain in the current archive should be computed as such measures indicate the improvement in the quality of solutions. The comparison of two different archives should be more straightforward if the number of dominated solutions is scaled by the size of the archive. The smaller the number of dominated solutions, the better is the convergence. This convergence metric is important in cases where a priori knowledge of the Pareto-optimal front is not available, like in the problem we are addressing here, implying that the accuracy cannot be assessed.

Uniformity The uniformity metric estimates how uniformly the individuals are distributed in the Pareto front, which is a requirement since one of our goals is to acquire a set of solutions that cover the entire Pareto-optimal region. The uniformity metric is defined as:[29]

$$\Delta = \sum_{i=1}^{N} \frac{|d_i - \overline{d}|}{N} \tag{9}$$

$$\overline{d} = \frac{1}{N} \sum_{i=1}^{N} d_i \tag{10}$$

where d_i is the crowding distance of an individual in objective space. Crowding distance is defined as half the perimeter of the largest hypercube around a particular individual without surrounding any other individuals.[29] In order to compute Δ, the crowding distance between consecutive non-dominated individuals (obtained by ranking the individuals according to each objective function) needs to be calculated. The average of these distances $\overline{d}$ also needs to be calculated. The boundary points are assigned a crowding distance of twice the distance to the nearest neighbour. It combines the values of different objective functions but, since they are normalized, the units are irrelevant. This measure is similar to the standard deviation of the crowding distance, therefore a small value of the uniformity metric characterizes a good set of solutions.

Spread This metric measures how wide-spread the individuals in the Pareto front are. It is calculated as the volume of the largest hypercube in objective space that contains all individuals. A large spread is desired to obtain a bigger trade-off surface. As the spread metric is determined considering just the individuals located at the boundary of the Pareto front, it is very sensitive to the presence of isolated points that can artificially improve it. Therefore, diversity can only be assessed by performing a combined analysis of the uniformity and spread metrics.

3.3. *Local optimization*

In this section, motivated by the efficiency of both evolutionary global search and local search strategies, we extend our algorithm into a multi-stage multiobjective EA. This hybrid algorithm combines the strengths of both evolutionary and local search, in particular of EAs and Simulated Annealing (SA).[37] To the best of our knowledge, SA has never been combined with an EA for solving coverage problems before. SA gives us the opportunity to quickly improve any solutions that are being considered, working as an unrestricted mutation operator in order to move to new non-dominated solutions. Unrestricted because in this stage we are not limited to the representation of the individual used in the evolutionary process and therefore we are able to fine tune any solution found by previous stages without compromising the efficiency of the evolutionary search process. Thanks to the complementary properties of EAs and local search heuristics, hybrid approaches often outperform either method operating alone. However, they still require excessive computational effort.

Our local search stage has been designed to optimize two distinct objectives, namely:

- Minimize the uncovered area;
- Maximize the average POD.

These objectives are prioritized differently according to our specific mission planning goals: we can only try to increase the average POD if there are no more insufficiently covered areas. The basic idea behind the algorithm is to try to locally improve the population of non-dominated solutions found by the EA (the solutions in the external set are the initial solutions to this procedure). This is a combinatorial problem where we have to choose the best set of inter track distances maximizing the coverage of an area characterized by a specific topography. We use distinct strategies depending on the type of sonar that is being used for seafloor mapping, more specifically if using multibeam (no nadir gap) or sidescan sonar (with nadir gap).

Strategy for the case without nadir gap

Initialization The algorithm starts by determining the original spacing between tracks, which was kept constant in the EA. Then it sets initial values for annealing process parameters, namely, the temperature and the neighbourhood size. The neighbourhood size is set to the initial track spacing that was determined previously. The initial temperature needs to be hot enough to allow a move to almost any neighbourhood state. The correct set of parameters for the cooling schedule and the for termination procedure were found through experimentation until a consistent improvement of the original solutions was verified.

Evaluation The algorithm analyses the performance of a solution by evaluating the objective functions in each subarea. We consider a subarea to be a region of the terrain delimited by a set of tracks. Basically we determine the percentage of the area that is insufficiently covered (per subarea and total) and the average POD (per subarea and total). Each track position is bounded by the position of the neighbouring tracks. Then we compare the current solution with the previous solution and with the best solution obtained so far. If it is better than the best or better than the previous solution, accept the solution. If it isn't better then the previous one, then apply the Metropolis criterion [38] and accept the solution with a given probability (fallback to the best if not accepted). At the end of this phase the temperature of the annealing process is reduced by applying a geometric decrement.

Check for termination This algorithm assumes that the annealing process will terminate when the temperature reaches a minimum value, if there has been no change in state for a certain number of iterations or if the maximum number of iterations is reached.

Mutation After analysing the performance, the subarea with a higher percentage of insufficiently covered area is selected for mutation. The main idea here is that if we place two tracks closer to each other higher coverage will be achieved in the area in between, at the cost of lower coverage in other areas. So this mutation operator basically reduces the spacing in the selected subarea and increases the spacing in other subareas because the sum of inter track distances is constant. The amplitude of the mutation Δ_i is obtained from equation 12, where r is a random number between -1 and 1, N is the neighbourhood size and T is the temperature of the process. It must be inferior to the previous track spacing s_i in subarea i. Then the spacing for that subarea is updated by subtracting the calculated delta value (equation 11). Each of the other areas will see its spacing increased considering to the amount of uncovered cells in each of them: areas with a lower ratio will have a bigger increment then areas with a higher ratio. The contribution of each subarea Δ_j, needed to compensate for Δ_i, is determined by calculating a set of weights using equation 16, inversely dependent on the the amount of uncovered cells in each subarea (see equation 14). Note that the described procedure is only executed if in the previous step it was found that there still existed uncovered areas. If that is not the case then we will try to maximize the average POD using a similar procedure where we try to decrease track spacing in the subarea where the average POD is lower. In this case the contribution of each subarea Δ_j depends directly on the average POD in each of those subareas (equation 15).

$$s_i = s_i - \Delta_i \tag{11}$$

$$\Delta_i = r \times N \times T \tag{12}$$

$$\Delta_i = -\sum_{\substack{j=1 \\ j \neq i}}^{k} \Delta_j \tag{13}$$

$$w'_j = \frac{1}{cells\_min_j} \tag{14}$$

$$w'_j = avgpod_j \tag{15}$$

$$w_j = \frac{w'_j}{\sum_{\substack{j=1 \\ j \neq i}}^{k} w'_j} \tag{16}$$

$$\Delta_j = w_j \times \Delta_i \tag{17}$$

Build path A 3D path is built using the determined set of inter track distances, which are 2D and measured horizontally. The path is determined considering its original height or depth, depending on its type of fitting (constant altitude or depth). The algorithm will continue evaluating different sets of inter track distances until the terminating criteria is met.

Strategy for case with nadir gap

Since the quality of data at the nadir gap is poor, a smarter approach can be taken to improve it. The most used technique to handle this problem is the uneven lawn-mowing coverage pattern, where consecutive pairs of tracks cover each other's nadir. This strategy differs from the previous one on the evaluation and mutation phase.

Evaluation Here we make the distinction between odd and even subareas. This distinction is made because each type of subarea is analysed differently. According to the adopted coverage pattern, even subareas will have smaller spacing since the adjacent tracks need to be closer together in order to cover their nadir gaps. Even subareas contain the complete nadir region belonging to each of the adjacent tracks, even the portion that lies on the adjacent subareas (the nadir extends to each side of the track considering that the sonar is being carried with zero roll angle). This happens because, when analysing performance on these areas, we need to make sure that they are covered by the pair of tracks that confine each subarea.

Mutation This mutation process is similar to the previous one, except that adjusting odd subareas only affects other odd subareas, while adjusting even subareas only affects adjacent subareas. The purpose of controlling track spacing in an even subarea is mainly to cover their nadir gaps, therefore only the adjacent subareas should compensate the variation of track spacing applied to it. These subsareas should remain untouched when adjusting odd subareas, hoping that the nadir gaps remain covered. Since the sea bottom may have a complex topography, adjusting a subarea may produce a negative impact on performance elsewhere.

3.4. *Environment and path representation*

We assumed that the knowledge about the environment is available because it wouldn't make sense to optimize a path to an unknown type of terrain. Information about the topography of the survey area is uploaded from a database or other available sources. For testing purposes a random terrain generator was used. A regular grid is then created inside this area. Each cell holds topographic information about the seafloor (longitude, latitude and depth) and a corresponding detection performance estimate that is obtained when a given solution is evaluated. Choosing the correct resolution is a critical task, since its manipulation is computationally intensive and it also affects the accuracy of the detection performance estimation.

Typically, coverage problems are solved by employing a path planning strategy that uses a set of equally spaced parallel tracks to cover the terrain. The path, composed by these tracks, is defined by a set of consecutive points which will always lie on the convex polygon that delimits the survey area. The location of these points is not related in any way to the location of the cells in the grid since these are only used to provide information about the seafloor and to estimate detection performance. It is not strictly required that tracks are parallel to each other but this usually happens because it makes it easier to generate a path that doesn't leave unexplored areas of space. Moreover, the success of the mission is highly dependent on the quality of the sonar acquired data. Typically a sonar does not work well when the vehicle is turning (the data gets distorted) so it is imperative that the path includes long straight tracks with minimal number of turns.

3.5. *Trajectory fitting to 3D terrain*

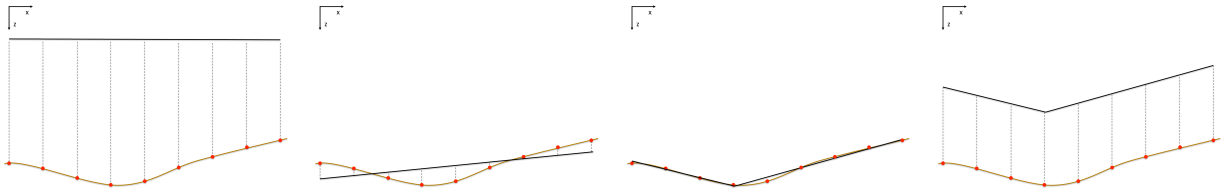
We consider that a path can be fitted at constant depth or at constant altitude. A path with constant depth will not take into account the structure of the terrain, although it has to meet the spatial constraints defined by the environment. Such paths are more efficient in terms of required energy and time, but if the topography of the survey area is complex (rough instead of a smooth surface, with a high standard deviation on elevation) it may exhibit poor detection performance. On the other hand, a path defined to maintain a constant altitude from the bottom terrain can display a better overall detection performance (certainly decreasing its standard deviation) but it will require more time and energy to be executed by the vehicle since its length is bigger. Such a path can be seen as a vertical projection of the constant depth path on the bottom. The calculation of a constant altitude path involves dividing each (constant depth) track in smaller subtracks and fitting them to the terrain, between two specific horizontal positions, performing a simple linear regression in 2D space. The algorithm for fitting tracks to the bottom is as follows:

1. Select a track to be fitted;
2. Estimate the depth of several points along the vertical projection of track on the terrain. The distance between the points is a pre-specified parameter;
3. Perform a linear regression to fit a straight line to the calculated positions on the terrain, maintaining the direction defined by the track;
4. Calculate the regression errors along the fitted track;
5. If the highest error is higher than a given pre-specified maximum error, then divide the track at the corresponding position and repeat the procedure for each subtrack. Track division leads to a reduction of the regression error;
6. Add an offset to the fitted track to ensure that it is positioned above the surface of the terrain. Ensure that its starting position is not at a higher depth than the ending position of the previous fitted track;
7. Add an offset corresponding to the desired altitude.

A graphical demonstration of this algorithm is presented in figure 3. Depth estimation is performed using Inverse Distance Weighting [39] of the depth measurements surrounding the prediction locations.

3.6. *Using infeasible data*

Some EAs regard infeasible solutions as useless individuals but various research showed that using infeasible individuals may affect and benefit the search process.[40] While keeping the search space small is a valid reason to discard



(a) Depth estimation (b) Linear regression (c) Track division and new regression (d) Add offset representing altitude

Fig. 3. Graphical demonstration of the track fitting procedure.

infeasible solutions, it was reported that techniques that apply this action perform worse than techniques that allow these solutions to become involved in the evolutionary process.[41] Yu and Zhou [40] presented a theoretical study that showed that including infeasible solutions could significantly affect the performance of EAs by accelerating the speed of convergence towards optimal solutions.

We propose a method that allows the use of infeasible individuals in the EA, taking them into account in the construction of the approximate model. This will help the exploration process in the boundary between feasible and infeasible regions. This allows the model to predict the location (feasible or infeasible region of the search space) of the individual being evaluated, which in turn will enable the EA to choose the best candidate individual to be a part in the next generation. In practice, we add a fourth output to the ANN model which describes the feasibility of a given solution and train the ANN with feasible and infeasible data. This information is then considered by the variation operators of the Informed EA and will determine if the candidate solution is accepted for precise evaluation and consequent integration in the next generation population. The integration of such strategy creates a data management problem since we need to sample the entire search space.

3.7. *Data density reduction*

An accurate approximation of the exact fitness function may be difficult to achieve when the amount of data available is inadequate, the data is not uniformly distributed and as a result not fully representative of the decision and objectives spaces. Generally speaking, the more accurate and the denser the training data is, the more accurate the approximate model will be. While a huge amount of the training data may guarantee generality of the model, it will require a very long training time. Improved efficiency can be achieved if redundant data is identified and eliminated from the dataset while maintaining generalization.

In this paper we propose a method to select efficient training data. Training data will be gradually generated during execution of the algorithm and will be continuously used for the on-line training of the ANN. Data selection and management are critical in this case, as the amount data will become extremely large if it is continuously accumulated. Density is controlled by establishing a minimum distance between individuals in decision space. We decided to introduce different distances in feasible and infeasible space to have a better understanding of the impact of this mechanism on the overall performance of the ANN. In the next section a series of experiments are performed to evaluate the behaviour of the adopted strategy.

4. Results

We have carried out a large amount of simulations to test our algorithms. With the following experiments we intend to:

- achieve a better understanding of the search capabilities of our algorithms;
- prove that our informed EA is superior in terms of performance to the standard EA;
- show that the integration of our local search strategy increases the efficiency of the search process;
- demonstrate that our multi-stage multiobjective algorithm can solve the mission planning problem successfully.

As the optimal Pareto front for our mission planning problem is not known, we use a procedure adopted by [42] to obtain an approximation of the Pareto front for each algorithm and different sets of settings. The procedure consists of the following steps:

- each algorithm with a specific combination of settings is executed 10 times, obtaining 10 independent sets S_i of non-dominated solutions;
- these sets are combined and the non-dominated solutions are stored in a set S_f;
- the non-dominated solutions in S_f are used as an approximation of the true Pareto-optimal set.

The process of tuning an EA for a given application is complex because there is a large number of alternatives and very limited knowledge about the effect of the parameters on the algorithm's performance is available. There is still a lack of theoretically proven principles to help the user find good values for these parameters. The current opinion on EAs admits that specific problems require specific EA

configurations for acceptable performance.[43] Therefore, an exhaustive analysis of the effects of the main parameters needs to be executed in order to understand their impact on the search process.

The following simulations were run in Ubuntu 12, running in the Parallels Desktop virtual machine on an Apple MacBook Pro, 2.3 GHz Intel Core i5 with 10 GB ram.

4.1. *Tuning the standard MOEA*

4.1.1. *Population size*

We intend here to obtain a better understanding of the role that the population size plays in the performance of our EA. To this purpose we performed a number of experiments that consisted in running our EA using distinct populations of sizes, namely containing 50, 100 and 200 individuals. The results of the experiments are presented in the dominance table 2, where the different obtained Pareto fronts are compared by assessing the ratio (%) of the number of non-dominated solutions of distinct sets. A brief description of the obtained datasets is also shown in table 3. Figure 4 shows the evolution of the individuals in the archive during time.

Table 2. Dominance table comparing the performance of the algorithm with different maximum population sizes.

Dom	50	100	200
50	-	1.49	2.41
100	28.13	-	4.82
200	43.75	19.40	-

Table 3. Description of the obtained dataset comparing the performance of the evolutionary process with different population sizes.

Pop. size	Gen's	Arch. size	Volume	Uniformity
50	475	64	0.04708	0.02746
100	421	67	0.07648	0.01811
200	327	83	0.07344	0.01392

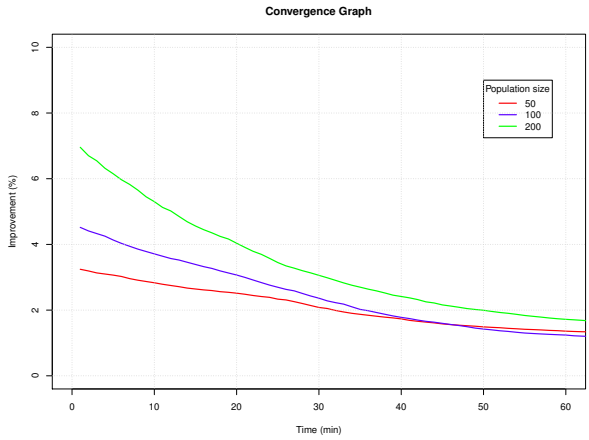


Fig. 4. Improvement plots for the considered population sizes. Archive population stabilized with evolution as convergence was achieved.

These results indicate that the performance of the EA improves by increasing population size. This is demonstrated by the growing dominance ratios and the faster convergence speed achieved by increasing this parameter. This improvement comes at the cost of a larger computational complexity, expressed by a decreasing number of generations in the same period of time. There is a trade-off between computational complexity and quality of the solutions. The EA with smaller population size has has reasonable computational complexity but is converging to poor solutions prematurely. Diversity is an important factor here because if no prior information about the best regions of the search space is available, then it is expected that the more diverse the initial population is, the greater the chances to find good solutions are. Therefore, the number of individuals in the population is an important parameter to control the amount of diversity. Our problem instance is characterized by a large decision space (many variables with a large range of admissible values) and in this case smaller population sizes proved to be insufficient because the individuals did not represent a large enough sample of the decision space.

Most of the performance gain is justified mainly by the use of a higher number of individuals in the initial population. Our elitist strategy, characterized by the use of an unrestricted external set with the best solutions to date, and our density control mechanism, that does not allow similar solutions in both the archive and the population, lead to a particular situation where the maximum population size was never achieved (there was no truncation with the tested range of population sizes). Therefore, a higher population size in our case implies a higher diversity among the individuals in the population. A higher number of diverse individuals is potentiating exploration, more prominently in the beginning as shown in figure 4 where we can witness higher initial improvement ratios for the larger population sizes.

Table 3 also shows that the sets with higher population sizes, 100 and 200, had similar hypervolume metric values, both much higher then for the set with 50 individuals. This means that the obtained Pareto front is much wider in these cases but, as expected, the individuals in the archive of the largest set are more uniformly distributed since this archive also contains more individuals.

It is then clear that the tuning of the population size has significant influence in the efficiency and convergence of our EA.

4.1.2. *Variation probabilities*

Finding robust parameter settings that do not lead to premature convergence is not simple, as the impact of these settings on the EA's performance is complex. This said, simulations need to be performed in order to assess the influence in the search process and to find the suitable recombination and mutation rates for our algorithm.

In this experiment, these rates are fine tuned by conducting a series of experiments using different combinations of these rates, seen on table 4 and evaluating the impact on performance of the algorithm. We tested 4 different parameter values for each rate, creating a total of 16 pairs of rates.

The outcome of our simulations is presented in table 5. It was found that simulations with higher recombination rates performed better since, in general, sets with higher rate dominate more and are less dominated in average. The same conclusion can be taken for the mutation rate, although it is not so evident. In order to gain a better understanding of the behaviour of the algorithm with the different mutation rates, we ploted the improvement rates correspondent to the sets 13 to 16 (fixed recombination rate of 1.0) as can be seen in figure 5. It shows that higher mutation rates lead to more stable rate of improvement (less fluctuations in evolution). The set 13 (in red) converged faster to sub-optimal solution as can be seen from the dominance table 5, indicating that the rate is too small. Sets 15 and 16, with mutation rates of 0.7 and 1.0, show a stable and sustained evolution even in the later stages of the simulation, indicating that the performance could be superior if the time allocated for the simulations was increased. Table 4 shows that runs with higher rates seem to populate archives with a higher number of individuals, occupying a bigger hypervolume and with better distributed individuals. Sets 12 and 16 are the best from the group. When comparing both sets, set 12 dominates twice the number of individuals that set 16 dominates. Set 12 has similar number of individuals in the archive to set 16, but these are better distributed. Considering these results, the best settings would be set 12 since we consider the dominance aspect more important then the uniformity of the individuals in objective space.

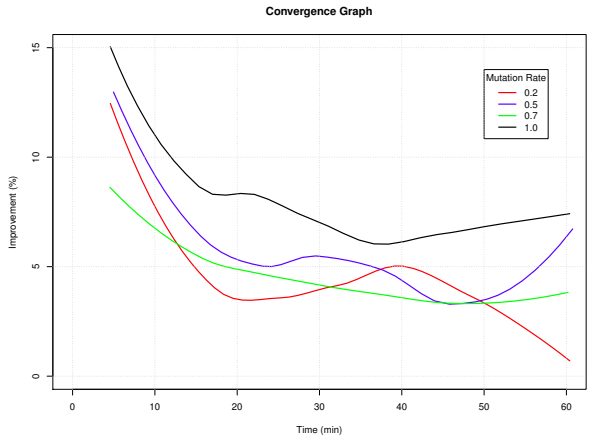


Fig. 5. Improvement plots comparing the evolution using the different mutation rates with a recombination rate equal to 1.0. Archived population stabilized with evolution as convergence was achieved.

During design phase of this algorithm we did not consider using recombination or mutation probabilities (meaning we wanted to use all individuals for reproduction), despite the fact that they were used in most case studies found. We intended to use the recombination process to explore the search space and the mutation process solely to improve the known Pareto front. It is understandable that there may be a need to limit the exploration process but this does not apply to the exploitation process where we can produce a mutated individual that dominates the original. These results show once more that the optimal combinations of rates are problem and algorithm dependent.

4.1.3. *Mutation magnitude*

The mutation magnitude parameter represents the maximum amount of change that is allowed in a single mutation operation. The actual mutation size will be defined by a uniformly distributed random number between zero and the product of the mutation magnitude parameter and the size of the variable range. A mutation with large magnitude is likely to produce large variations which would facilitate better exploration of the undiscovered regions of the search space while a small magnitude usually produces small variations that are better for exploitation of the already found solutions. Whether potentiating exploitation or exploration would be the best objective for this operator is not clear, so we decided to manually tune this parameter to have a better understanding of its impact on overall performance. The dominance table 8, obtained through experimentation with different magnitudes described in table 7, shows that the best performance was indeed obtained with the smaller mutation magnitude (0.1), as the set 1 presents higher av-

Table 4. Sets of probabilities used for recombination and mutation.

Set	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P_R	0.2	0.2	0.2	0.2	0.5	0.5	0.5	0.5	0.7	0.7	0.7	0.7	1.0	1.0	1.0	1.0
P_M	0.2	0.5	0.7	1.0	0.2	0.5	0.7	1.0	0.2	0.5	0.7	1.0	0.2	0.5	0.7	1.0

erage dominance rate and is less dominated by others.

Table 7. Sets of mutation magnitudes using by the mutation operator.

Set	1	2	3	4
Mag	0.1	0.2	0.3	0.4

Table 8. Dominance table comparing the performance of the algorithm with different mutation magnitudes.

Dom	1	2	3	4	Avg
1	-	11.22	9.57	37.65	15.71
2	1.96	-	5.22	36.47	11.46
3	5.88	7.48	-	40.48	13.73
4	1.96	0.00	0.87	-	0.98
Avg	6.13	9.35	8.26	41.00	

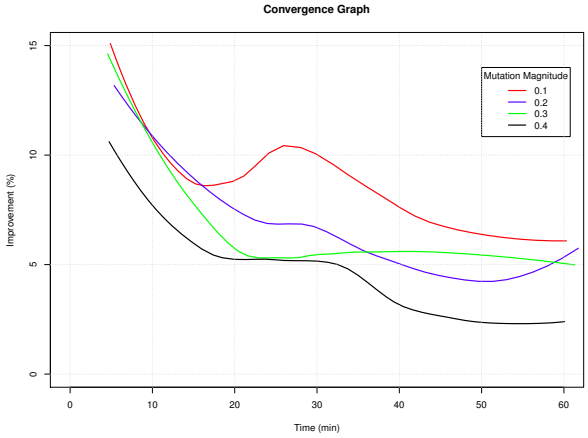


Fig. 6. Improvement plots comparing the evolution using the different mutation magnitudes.

The information provided by the convergence graph 6 helps justifying this best performance by displaying a much higher improvement ratio throughout the simulation run. We can clearly see that the improvement ratio follows a specific trend, increasing with decreasing magnitudes. The lack of convergence (none converged to zero improvement) indicates that better results could be achieved if more time was allocated for the simulation runs.

4.2. *Tuning the informed EA*

Here we are going to evaluate the impact of some of the parameters that control the fitness approximation strategy which was adopted. These parameters affect the integration of the model in the EA and the level of approximation of the model.

4.2.1. *NN training generation gap*

The main reason that lead to the introduction of a ANN training gap is that the process of ANN training can allocate a large amount of time in the evolutionary process, so we need to assess the impact of such operation on the performance of our EA. This type of control is usually performed once in a fixed number of generations. This method has the disadvantage of forcing a constant control frequency, meaning that the algorithm cannot react to the availability of recent training data that represents previously unknown regions of the search space. Since the training dataset size is expected to stabilize during the evolutionary process, the accuracy of the model should increase over time due to the consecutive training phases.

This lead us to perform a series of experiments comparing the performance of the informed EA using distinct training generation gaps (1, 5, 10 and 20 generations).

Table 9. Dominance table comparing the performance of the EA using different training generation gaps.

Dom	1	5	10	20	Avg
1	-	20.00	12.12	28.41	20.18
5	12.73	-	26.26	38.64	25.88
10	4.04	18.89	-	18.18	13.70
20	14.29	8.89	10.10	-	11.09
Avg	10.35	15.93	16.16	28.41	

Numerical results presented in table 9 show that the algorithm performed better using generation gaps of one and five, with the expected drop-off in performance for higher

Table 5. Dominance table comparing the performance of the algorithm with different variation probabilities.

Dom	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	Avg_1	Avg_2
1	-	16.88	49.35	7.41	18.92	10.75	28.75	6.67	10.00	25.88	9.57	0.00	21.65	7.61	14.89	0.93	15.28	
2	7.23	-	25.97	3.70	8.11	3.23	15.00	1.11	3.33	16.47	5.32	0.93	12.37	2.17	6.38	0.93	7.48	9.89
3	1.20	3.90	-	2.47	4.05	1.08	0.00	0.00	1.11	2.35	2.13	0.00	1.03	2.17	0.00	0.00	1.43	
4	4.82	11.69	58.44	-	20.27	6.45	23.75	6.67	7.78	28.24	10.64	3.74	17.53	9.78	18.09	2.78	15.38	
5	0.00	10.39	27.27	3.70	-	3.23	13.75	2.22	2.22	11.76	3.19	0.00	7.22	4.35	3.19	0.00	6.17	
6	10.84	22.08	63.64	6.17	17.57	-	33.75	8.89	7.78	42.35	10.64	2.80	17.53	7.61	13.83	3.70	17.95	11.71
7	2.41	5.19	18.18	2.47	6.76	3.13	-	2.22	4.44	11.76	6.38	0.00	4.12	3.26	3.19	1.85	5.03	
8	8.43	18.18	55.84	7.41	21.62	7.29	38.75	-	7.78	31.76	17.02	0.93	18.56	11.96	17.02	2.78	17.69	
9	7.23	11.69	51.95	6.17	20.27	7.29	30.00	6.67	-	32.94	15.96	1.87	23.71	6.52	13.83	2.78	15.93	
10	3.61	5.19	25.97	6.17	6.76	0.00	6.25	2.22	1.11	-	3.19	1.87	4.12	3.26	1.06	0.00	4.72	15.05
11	1.20	9.09	28.57	4.94	16.22	2.08	20.00	3.33	6.67	20.00	-	1.87	10.31	3.26	9.57	0.93	9.20	
12	12.05	25.97	64.94	11.11	32.43	20.83	53.75	21.11	23.33	57.65	36.17	-	35.05	20.65	29.79	10.19	30.33	
13	2.41	5.19	20.78	4.94	6.76	4.17	13.75	2.22	8.89	17.65	7.45	0.00	-	2.17	4.26	0.93	6.77	
14	1.20	14.29	46.75	4.94	13.51	4.17	23.75	5.56	3.33	25.88	10.64	0.00	17.53	-	13.83	0.93	12.42	14.50
15	9.64	12.99	44.16	6.17	18.92	9.38	20.00	6.67	5.56	17.65	12.77	0.93	11.34	8.70	-	0.93	12.39	
16	13.25	28.57	70.13	7.41	29.73	10.42	56.25	20.00	15.56	49.41	25.53	4.67	23.71	17.39	24.47	-	26.43	
Avg_1	5.70	13.42	43.46	5.68	16.13	6.23	25.17	6.37	7.26	26.12	11.77	1.31	15.05	7.39	11.56	1.98		
Avg_2	17.07				13.47				11.61				8.99					

values. The ideal training generation gap needs to be determined considering the specific problem instance being addressed and the chosen optimization algorithm because the training gap is dependent on the total number of generations (in our experiment the average was around 30 for one hour simulations) and on the number of generations needed to stabilize the training dataset (not achieved here).

4.2.2. *Mutation Settings*

The dominance table 11, obtained through experimentation with different settings described in table 10, shows that the best performance was obtained with the smaller mutation magnitude (0.1), as the set 1, 2 and 3 present higher average dominance rate and are less dominated then the others. As for the number of mutations that should be performed using the approximate model, the dominance table shows that the best results are achieved using 50. When using small mutation magnitudes there is not a real advantage in using a higher number of mutations since the individual's fitness values will be very close to each other when ranked.

Table 10. Sets of mutation magnitudes and number of mutations used by the mutation operator.

Set	1	2	3	4	5	6	7	8	9
Mag	0.1	0.1	0.1	0.2	0.2	0.2	0.3	0.3	0.3
Mut	20	50	100	20	50	100	20	50	100

4.2.3. *Filtering in feasible and infeasible space*

We intend first to compare the performance of the training procedure using the distinct data densities presented in table 12 and then to assess its impact on the performance of the EA. The sampling density experiments were performed on a continuously updated ANN training dataset where each data tuple was obtained through exact objective function evaluations.

As expected, we achieve better training performance by using smaller distances between tuples in feasible space. Table 13 shows that the average training error, obtained through split sample validation, increased as the distance between tuples also increased. This suggested that the optimal normalized minimal distance between tuples in the feasible region should be 0.05 (or less). Comparing the sets with the same minimal distance in the feasible region reveals that the training error decreases if the minimal distance in the infeasible region is increased, reducing the total number of tuples in this region. Table 14 presents the results of multiple simulations with the different trained ANNs. It is clear that the data density reduction mechanism (increased distance between data tuples) influences our EA: the average dominance of the obtained Pareto fronts decreased as data density decreased. The impact of the variability in ANN training performance observed earlier, in feasible and infeasible spaces, is also evident on the obtained results.

In any situation the created ANN was trained properly and was able to give reasonable estimations when fed with unknown inputs. Training data reduction handles data redundancy and improves data processing efficiency in terms of both storage and processing time. With this methodology, an efficient dataset can be maintained for on-line training of the ANN.

4.3. *Adaptive tuning*

The manually determined set of parameter values may not offer the best performance as they are kept constant throughout the evolutionary process. We decided that the only parameters that should be dynamic are the mutation magnitude and the ANN generation gap. Variation rates should be kept constant since the informed algorithm will only generate individuals that are estimated to be non-dominated, hence guaranteeing that the evolutionary process is progressing. Our strategy consists in a decreasing variation of the mutation magnitude during the execution of the algorithm. The appropriateness of a small or large mutation magnitude changes dynamically depending on the state of the search process as well as the properties of the search space. A decreasing scheme was chosen because we expect that a larger magnitude to facilitate the exploration of the of the search space in the beginning and smaller magnitudes to help exploitation of the already determined solutions in the end. The objective is to maintain the improvement ratio above a pre-specified minimum value until the end of the run, achieving a balance between exploitation with the exploration in the search space. If the improvement ratio, calculated at the end of each generation, falls below a minimum value, the mutation magnitude is multiplied by a constant (inferior to 1) similar to the cooling ratio in SA, decreasing its value. The previously determined magnitude will be used as initial value. Similarly, the ANN training generation gap should increase while the training error is kept close to the desired value.

The results of the experiment are presented in table 15. The population determined by running the informed EA with adaptive tuning is superior to the one determined by the manually tuned EA. The Pareto Front was wider, with more individuals and these were more uniformly distributed. This proves that an adaptive mutation magnitude helps in increasing the diversity which in turn helps balancing exploration and exploitation of the search space which finally helps in improving the solution quality and the convergence rate of the algorithm. The ANN training generation gap, which started as 1, remained the same throughout the execution of the algorithm as the training error was higher then the desired value. The time that we had allocated for this experiment was not enough to observe a variation of this parameter. Nevertheless, we concluded that the impact of dynamically controlling this parameter

Table 11. Dominance table comparing the performance of the EA using different mutation settings.

Dom	1	2	3	4	5	6	7	8	9	Avg_1	Avg_2
1	-	6.45	7.53	13.25	12.50	25.81	19.39	24.72	33.67	6.06	
2	22.33	-	12.90	13.25	22.34	24.73	26.53	28.09	36.73	9.09	20.73
3	11.65	6.45	-	10.84	11.46	23.66	26.53	31.46	33.67	8.08	
4	10.68	1.08	7.53	-	14.58	13.98	23.47	19.10	21.43	6.06	
5	11.65	7.53	11.83	14.46	-	18.28	21.43	28.09	27.55	2.02	14.27
6	10.68	2.15	5.38	7.23	14.58	-	21.43	10.47	29.59	4.04	
7	5.83	2.15	6.45	9.64	10.42	19.35	-	19.10	26.53	4.04	
8	8.74	6.45	7.53	4.82	9.38	9.68	18.37	-	17.35	6.06	10.12
9	7.77	7.53	3.23	6.02	8.33	11.83	13.27	24.72	-	7.07	
Avg_1	10.39	5.70	7.53	10.60	12.23	18.92	21.33	22.96	28.88		
Avg_2		7.87			13.92			24.39			

Table 12. Sets of decision space distances used by the filtering mechanism.

Set		1	2	3	4	5	6	7	8	9
Minimum Distance	Feasible	0.05	0.05	0.05	0.10	0.10	0.10	0.20	0.20	0.20
	Infeasible	0.05	0.10	0.20	0.10	0.20	0.40	0.20	0.40	0.80

Table 13. ANN training performance using datasets with the distinct densities presented in table 12. We present the average number of tuples in the dataset during the execution of the EA. Split sample validation was performed for assessing the training error, involving randomly splitting the dataset into two subsets, using one for training and the other for validating the generality of the result. The validation error is the one that is presented.

Dataset	1	2	3	4	5	6	7	8	9
Avg Tuples	746.63	609.29	702.45	346.43	313.86	268.86	113.71	89.50	77.60
Avg MSE	0.01934	0.01472	0.00896	0.03211	0.02961	0.01515	0.05850	0.02258	0.00679

is small since our training algorithm detects error convergence and terminates the procedure without significant loss of time.

Table 15. Comparison of the obtained non-dominated sets using different tuning procedures.

Tuning	Manual	Adaptive
% Dominated	25.42	4.17
Individuals	59	96
Generations	31	37
Volume	0.07198	0.07554
Uniformity	0.02003	0.01098

4.4. *Standard EA VS informed EA*

Since the EAs have a different structure we decided to analyse their performance at specific instants of time. Table 16 shows the results of the simulation. We can observe that at the initial stages the Pareto set obtained by the informed EA is dominated to a larger extent by the one obtained by the standard EA. This can be due to the longer initialization phase that is executed on the former, as the neural network needs to be trained, while the latter starts the evolutionary process sooner. The increasing dominance that the Pareto set generated by the informed EA exhibits is a consequence of an improving representation of the search space, maintained by the ANN training dataset, as the search progresses. This proves that the integration of the ANN is advantageous to the search process since it is guiding the search to more promising regions.

Table 14. Dominance table presenting a comparison of the different Pareto fronts obtained by using distinct ANNs.

Dom	1	2	3	4	5	6	7	8	9	Avg_1	Avg_2
1	-	13.95	1.19	16.49	14.81	18.28	27.17	15.48	11.11	14.81	
2	17.78	-	0.00	21.65	14.81	20.43	27.17	11.90	12.12	15.73	21.63
3	27.78	37.21	-	36.08	28.70	35.48	46.74	34.52	28.28	34.35	
4	6.67	5.81	1.19	-	0.00	6.45	11.96	9.52	5.05	5.83	
5	21.11	24.42	5.95	17.53	-	13.98	22.83	25.00	11.11	17.74	12.09
6	13.33	18.60	2.38	16.49	4.63	-	20.65	15.48	10.10	12.71	
7	15.56	10.47	1.19	11.34	1.85	7.53	-	13.10	7.07	8.51	
8	16.67	17.44	4.76	20.62	18.52	22.58	22.83	-	17.17	17.57	12.69
9	15.56	12.79	2.38	17.53	7.41	13.98	11.96	14.29	-	11.99	
Avg_1	16.81	17.59	2.38	19.72	11.34	17.34	23.91	17.41	12.75		
Avg_2		12.26			16.13			18.03			

Table 16. Comparison of the obtained non-dominated sets as the EAs are executed.

Time (min)	10		20		40		60	
EA	Std	Inf	Std	Inf	Std	Inf	Std	Inf
% Dominated	2.70	26.67	10.11	18.60	23.71	12.50	30.39	4.76
Individuals	74	75	89	86	97	80	102	84
Generations	6	4	15	14	26	31	37	50
Volume	0.06510	0.07065	0.06787	0.07186	0.07241	0.07832	0.07780	0.07369
Uniformity	0.01499	0.01392	0.01336	0.01322	0.01145	0.01593	0.01255	0.01441

4.5. *Mission planning with EAs*

It is important to exemplify what will be the typical output of the execution of our planning algorithm and demonstrate the role that the decision maker will have to play. For this test, a smooth terrain was randomly generated using the Diamond-Square algorithm, a fractal algorithm that's popular for generating realistic looking terrain. We decided to stop the algorithm after one hour so we could analyse its behaviour in detail. Our planning problem was subject to the constraints detailed in table 17. We selected five distinct solutions from the Pareto front shown in figure 7 and these are fully described in table 18. They were chosen because they clearly demonstrate the trade-offs between the objective function values obtained in our mission planning procedure. We can observe that a higher detection performance implies more time to complete the mission. As expected, we can also observe that the detection performance is better when using smaller inter track distances (a higher number of tracks is used). The velocity also has considerable impact on performance. We came to this conclusion because the best solutions (in terms of detection performance) use the minimum values for this variable as the energy that is saved by doing so is being used to increase the length of the path by using a higher number of tracks.

Table 17. Design constraints used in this test.

Parameter	Bounds	Type of constraint
Battery capacity	800 Wh	vehicle performance
Velocity (/water)	$1 \leq v \leq 3$	vehicle performance
Sonar max range	80 m	sensor performance
Sonar beam angle	$10 \leq \alpha \leq 70$	sensor performance
Max operating time	10 h	mission requirement

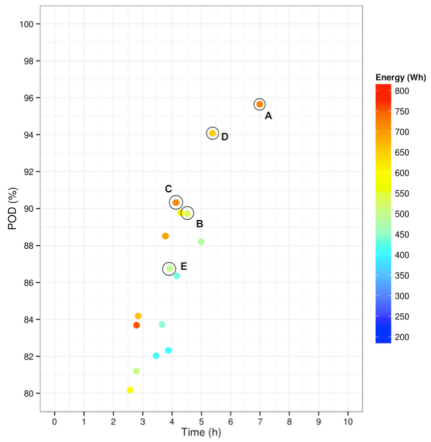


Fig. 7. Example of a Pareto Front containing several solutions for the mission planning problem.

4.6. *Local optimization*

A group of experiments was executed with the goal of demonstrating the effect of the inclusion of a local search algorithm in our mission planning procedure. Non-dominated solutions previously found by our informed EA were used as initial solutions here. Table 19 presents descriptive statistics for the optimized set of solutions for covering a given area with a multibeam sonar. It shows the variation in the amount of insufficiently covered cells ("MinRatio") and the variation in POD. In average we obtained approximately a decrease of 1% in the former and a increase of 0.1% in the latter, clearly showing the improvement made by local search in this scenario. In order to analyse this results we need to consider that the solutions being optimized were the best solutions found by an evolutionary process and that all these solutions had an average POD higher than 80%. This said, big improvements were not expected from small adjustments, at least in a smooth 3D terrain such as the one being considered. In fact, given that the objectives in this process have different priorities, we were expecting that a decrease in the amount of insufficiently covered cells caused a decrease of the average POD. Insufficiently covered areas should be subject to more intense coverage, sacrificing performance in other areas if needed. But we found that the local search algorithm is able to adjust the path successfully without sacrificing mission performance in terms of coverage quality (though it may use more vehicle resources). Figure 8 shows a graphical representation of a given solution before and after the local optimization procedure was executed. The improvement in this case was a decrease of 10.0% in the amount of insufficiently covered cells and an increase of 0.3% in the average POD. This is visually identified by the presence of lighter shades of red in the optimized solution plot.

Table 19. Descriptive statistics of the obtained solutions for the coverage problem using a multibeam sonar.

	Δ_{MR} (%)	Δ_{POD} (%)	Time (s)
Min	-11.88	-1.60	21.00
1Q	-1.82	-0.10	25.50
Mean	-1.15	0.09	58.48
3Q	-0.01	0.13	44.00
Max	0.00	3.70	338.00
SD	2.35	0.70	61.45
Var	5.51	0.48	3776.45

Similar experiments were performed, but using a sidescan instead of multibeam sonar. Table 20 presents the descriptive statistics for the optimized set of solutions. We observe that the average improvement in this scenario is almost 5 times more than in the previous one. This demonstrates the need for uneven lawn-mowing coverage pattern when using a sidescan sonar. The obtained results allowed us to take some additional conclusions related to the usefulness of this search process. As explained earlier, it is required an even number of tracks to cover the nadir gaps caused by the use of a sidescan sonar. If the original number of tracks is odd, then we need to add or remove a track in order to use our algorithm. The simulations showed that while the performance is positively affected by an addition of a track, it is negatively affected by its removal. The local optimization algorithm could not compensate the decrease in performance caused by the removal of a track. Table 21 presents the descriptive statistics for the optimized set of solutions excluding the ones where tracks were removed. It can be seen that the average POD increases and that the amount of cells with insufficient coverage is reduced. In this scenario the preferred action is to add a track, but since mission planing constraints need to be respected, it may not be possible to do so. Therefore we conclude that, in a time critical application such as the one being addressed here, it may be better to simply skip the local optimization of solutions that required removal of a track since it is not worth the extra computational time. Figure 9 shows a graphical representation of a given solution before and after the local optimization procedure was executed. The improvement in this case was a decrease of 14.0% in the amount of insufficiently covered cells and an increase of 0.9% in the average POD. This is visually identified by the almost inexistence of red regions in the optimized solution plot. Notice that the tracks were simply rearranged maintaining its direction and depth and that no tracks were added or removed. This demonstrates the usefulness and complementarity of both our local optimization algorithm and our EA.

Table 20. Descriptive statistics of the complete set of solutions obtained for the coverage problem using a sidescan sonar.

Table 18. Some distinct solutions in the Pareto Front.

Index	$\overline{POD}$ (%)	Energy (Wh)	Time (s)	Depth (m)	Altitude (m)	TrackDist (m)	Velocity (m/s)	Direction (◦)
A	95.65	728.80	06:59:20	-	13.24	41.13	1.07	91.44
B	89.72	537.04	04:31:11	-	11.63	63.26	1.14	32.31
C	90.33	723.73	04:07:59	82.38	-	57.46	1.34	259.87
D	94.08	656.93	05:22:51	81.24	-	50.96	1.15	16.70
E	86.75	500.53	03:55:52	-	18.00	72.31	1.18	201.79

	Δ_{MR} (%)	Δ_{POD} (%)	Time (s)
Min	-19.41	-8.84	24.00
1Q	-10.86	-0.01	29.75
Mean	-4.99	0.59	64.74
3Q	-0.73	1.49	51.75
Max	11.81	5.51	380.00
SD	5.80	1.76	67.04
Var	33.64	3.10	4494.34

Table 21. Descriptive statistics of the set of solutions excluding the ones where tracks were removed.

	Δ_{MR} (%)	Δ_{POD} (%)	Time (s)
Min	-19.41	-1.99	24.00
1Q	-11.68	0.01	29.50
Mean	-5.43	0.81	64.77
3Q	-0.74	1.54	50.50
Max	2.09	5.51	380.00
SD	5.56	1.28	68.58
Var	30.85	1.64	4702.95

4.7. *Mission planner performance assessment*

Table 22. Summary of the mission planner by Williams [17].

Williams's planner features
- Local planner, considers single convex areas
- 2D algorithm
- Only considers track spacing
- Maximizes the reward obtained from performing a track
- The POD is a function of range and seabed type
- Assumes the sensor is a sidescan sonar
- Optimizes track spacing (considers nadir gap)

From all the algorithms that were discussed previously for the underwater CPP domain, we selected the mission planner presented by Williams [17] since it is better suited to a search and rescue problem with similar assumptions to ours. It optimizes track spacing between all tracks covering the area of operations and also considers POD as a function of range and seabed type, some features that other algorithms fail to include. A detailed description of its features is presented in table 22, but this study should be complemented with a few experiments to prove the superiority of our algorithm (Informed EA). Since its primary concern is to calculate track spacing between parallel tracks, we needed to feed that algorithm with information calculated by our algorithm such as ideal vehicle velocity, coverage depth and direction. Therefore, the comparison made here is not fair since our algorithm needed a higher amount of time to calculate these parameters. However, this is a distinctive feature of our algorithm and even if this information is shared with the other algorithms, ours will still outperform theirs. The experimentations consisted of mission planning using two different terrains. The first is a simple planar terrain with a 2% slope and with an area of one square km. The other is a terrain with a complex (irregular) topography calculated using a diamond-square algorithm. The results are presented in tables 23 and 24.

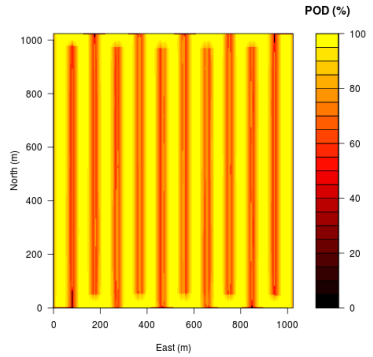
Table 23. Obtained solutions using different local planners on simple terrain.

Planner	$\overline{POD}$ (%)	Energy (Wh)	Time (h:m:s)	Evaluation Time (m:s)
Informed EA	95.775	521.612	03:19:48	01:47
Williams	89.978	594.202	03:47:29	01:10

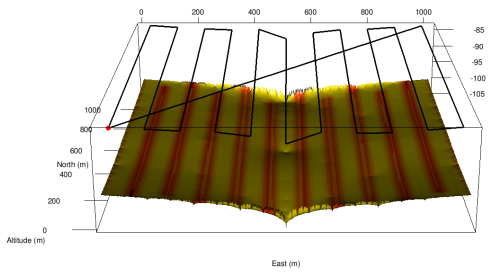
Table 24. Obtained solutions using different local planners on complex terrain.

Planner	$\overline{POD}$ (%)	Energy (Wh)	Time (h:m:s)	Evaluation Time (m:s)
Informed EA	98.226	547.486	04:27:25	00:54
Williams	96.081	480.205	03:54:22	02:17

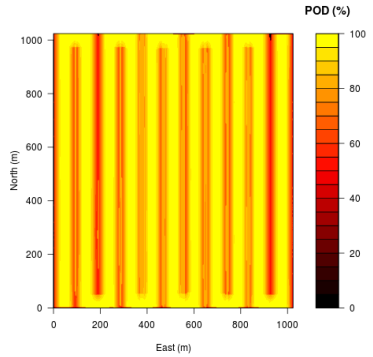
In both scenarios our algorithm displayed the best estimated detection performance. Our algorithm has a high



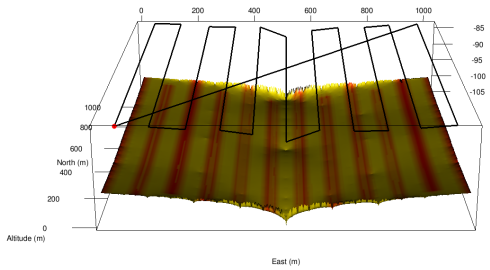
(a) 2D coverage plot of original solution.



(b) 3D coverage plot of original solution.



(c) 2D coverage plot of the optimized solution.



(d) 3D coverage plot of the optimized solution.

Fig. 8. 3D local optimization of a solution obtained by the EA using a multibeam sonar.

execution time for two specific reasons. First, because it is a complex optimization technique that estimates several mission parameters. Second, because the path fitting algorithm that creates a constant altitude path is computationally expensive. That was the reason why our algorithm was slower in the experiment with the simpler terrain, as that was the only case where the solution had constant altitude. The other mission planners do not have this capability. These experiments prove that our informed mission planner is more complete, although at the cost of higher complexity and higher execution time.

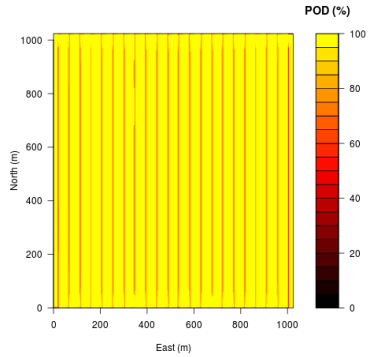
5. Conclusions

This paper introduced an multiobjective multi-stage approach combining EA with simulated annealing for planning minehunting operations in static 3D environment with predictable terrain. Our algorithm maintains a diverse population of feasible solutions in order to explore the search space and uses simulated annealing to improve the best solutions found and produce new solutions in the neighbourhood. Our experiments showed that the proposed local optimization phase significantly helps to improve the performance of the algorithm, however at the cost of a higher computational time. We demonstrated the superiority of our informed algorithm, that used an approximate model to guide the search, over an adaptation of SPEA 2 designed for our particular problem. The influence of several parameters was also assessed. We found that the adaptive mutation magnitude contributes to a better performance. Finally, we also exemplified what would be the typical output of the execution of our planning algorithm and demonstrated the role that the decision maker may have to play when planning a minehunting mission with an AUV.

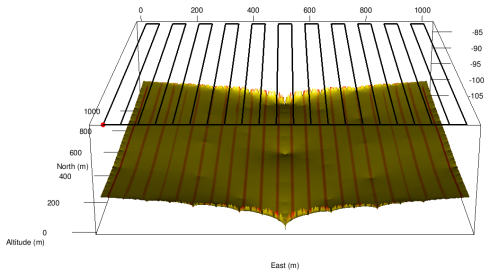
In the near future we are going to explore mission planning with distinct priorities for specific areas. The idea is to use these algorithms to obtain a Pareto front for each area an then to efficiently try to interconnect the coverage paths, thus becoming a variant of the travelling salesman problem.

Acknowledgements

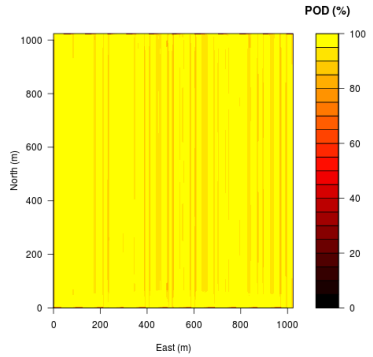
This work is financed by the ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by Na-



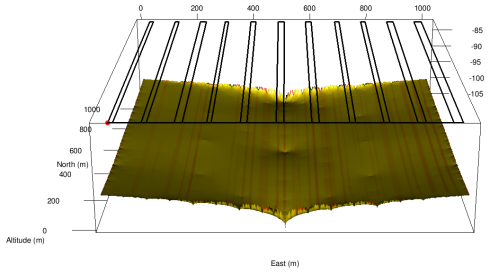
(a) 2D coverage plot of original solution.



(b) 3D coverage plot of original solution.



(c) 2D coverage plot of optimized solution.



(d) 3D coverage plot of optimized solution.

Fig. 9. 3D local optimization of a solution obtained by the EA using a sidescan sonar.

tional Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-037281.

References

[1] T. M. Cashman, *Sweeping Changes for Mine Warfare: Controlling the Mine Threat* (Naval Postgraduate School, California, USA, 1994).

[2] H. Choset, Coverage for robotics - a survey of recent results, *Ann. Math. Artif. Intell.* **31**(1) (2001) 113-126.

[3] K. Fujimura, Path Planning with Multiple Objectives, *IEEE Robot Autom. Mag.* **3**(1) (1996) 33-37.

[4] J. Xiao, Z. Michalewicz, L. Zhang and K. Trojanowski, Adaptive Evolutionary Planner/Navigator for Mobile Robots, *IEEE Trans. Evol. Comput.* **1**(1) (1997) 18-28.

[5] J. Current, Multi-objective design of Transportation Networks, *Ph.D. thesis* (The Johns Hopkins University, Baltimore, 1981).

[6] S. Yang and C. Luo, A neural network approach to complete coverage path planning, *IEEE Trans. Syst. Man. Cybern. B Cybern.* **34**(1) (2004) 718-724.

[7] W. Sheng, N. Xi, M. Song, Y. Chen and P. Macneille, Automated CAD-guided robot path planning for spray painting of compound surfaces, *Proc. of the 2000 IEEE Int. Conf. Int. Robot (IROS)* (2000), pp. 1918-1923.

[8] J. Jin and L. Tang, Coverage path planning on three-dimensional terrain for arable farming, *J. Field Robotics* **28**(3) (2011) 424-440.

[9] H. Zhang, D. Westhoff, J. Zhang and G. Zong, Service robotic systems for glass curtain walls cleaning on the high-rise buildings, *VDI BERICHTE* (2006).

[10] E. Galceran and M. Carreras, Efficient seabed coverage path planning for ASVs and AUVs, *Proc. of the 2012 IEEE Int. Conf. Int. Robot (IROS)* (2012), pp. 88-93.

[11] P. Cheng, J. Keller and V. Kumar, Time-optimal uav trajectory planning for 3D urban structure coverage, *Proc. of the 2008 IEEE Int. Conf. Int. Robot (IROS)* (2008), pp. 2750-2757.

[12] T. Oksanen and A. Visala, Coverage path planning algorithms for agricultural field machines, *Journal of Field Robotics* **26**(8) (2009) 651-668.

[13] A. Xu,P. Virie and I. Rekleitis, Optimal complete ter-

rain coverage using an unmanned aerial vehicle, *Proc. of IEEE ICRA 2011* (2011), pp. 2513-2519.
[14] J. Edmonds and E. Johnson, Matching, Euler tours and the Chinese postman, *Mathematical Programming* **5** (1973) 88-124.
[15] J. R. Stack and C. M. Smith, Combining random and data-driven coverage planning for underwater mine detection, in *Proc. of IEEE Oceans 2003* (USA, San Diego, CA, 2003), pp. 2463–2468.
[16] C. Fang and S. Anstee, Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle, in *Proc. of IEEE Oceans 2010* (Australia, Sydney, 2010), pp. 1-8.
[17] D. P. Williams, On optimal AUV track-spacing for underwater mine detection, in *Proc. of IEEE ICRA 2010* (USA, Anchorage, 2010), pp. 4755–4762.
[18] J. Das, K. Rajan, S. Frolov, F. Pyy, J. Ryany, D. Caron and G. Sukhatme, Towards marine bloom trajectory prediction for AUV mission planning, *Int. Conf. Robot. Autom. (ICRA) 2010* (2010), pp. 4784-4790.
[19] Z. Yan and Y. Zhao and T. Chen and L. Jiang, Fault recovery based mission scheduling of AUV for oceanographic survey, *World Congr. Intelligent Control Autom. (WCICA) 2012* (2012), pp. 4071-4076.
[20] M. Morin, I. Abi-Zeid, Y. Petillot and C. Quimper, A hybrid algorithm for coverage path planning with imperfect sensors, emphIEEE Int. Conf. Int. Robot (IROS) 2013, (2013), pp. 5988–5993.
[21] J. Baylog and T. Wettergren, Online determination of the potential benefit of path adaptation in undersea search, *IEEE J. Oceanic Eng.* **39**(1) (2014) 165–178.
[22] L. Paull, S. Saeedi, M. Seto and H. Li, Sensor driven online coverage planning for autonomous underwater vehicles, *IEEE Int. Conf. Int. Robot (IROS) 2012* (2012), pp. 2875-2880.
[23] S. Land and H. Choset, Coverage path planning for landmine location, in *Proc. of 3rd Int. Symp. on Technology and the Mine Problem* (USA, Monterey, 1998).
[24] B. O. Koopman, *Search and Screening: General Principles with Historical Applications* (The Military Operations Research Society, Alexandria, USA, 1999).
[25] M. Prior, *Simple sonar modelling for diver detection*, Internal Memo (NATO Undersea Research Centre, 2006).
[26] C. R. Rollins, A Cumulative Detection Probability Method, in *Proc. of 28th Navy Symp. on Underwater Acoustics* (1970).
[27] L. Nash, G. L. Hover and R. E. Burns, *Additional Analyses of Probability of Detection (POD) in Search and Rescue (SAR) Project Data*, report (United States Coast Guard, 1982).
[28] J. Carnes, *Effective Area Searching for Ground Team Members: POD, Critical Spacing, and Smart Search Techniques*, report (California OES and MRA, 1993).
[29] K. Deb, *Multi-objective optimization using evolutionary algorithms* (Wiley, Chichester, UK, 2001).
[30] M. Ehrgott, *Multicriteria optimization* (Springer, Berlin, Germany, 2000).
[31] C. Darwin, *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life* (John Murray, London, UK, 1859).
[32] E. Zitzler, M. Laumanns, and L. Thiele, *SPEA2: Improving the strength pareto evolutionary algorithm*, Technical report (Swiss Federal Institute of Technology, 2001).
[33] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, *IEEE Trans. Evol. Comput.* **6**(2) (2002) 182–197.
[34] J. Canny, *The Complexity of Robot Motion Planning* (MIT Press, Cambridge, MA, USA, 1988).
[35] G. Huang, Learning capability and storage capacity of two-hidden-layer feed-forward networks, *IEEE Trans. Neural Networks* **14**(1) (2003) 274-281.
[36] T. Goel and N. Stander, A Study on the Convergence of Multiobjective Evolutionary Algorithms, in *Proc. of 13th AIAA/ISSMO MA&O Conf.* (USA, Fort Worth, Texas, 2010), pp. 1-18.
[37] S. Kirkpatrick, C. Gelatt and M. Vecchi, Optimization by Simulated Annealing, *Science* **220** (1983) 671-680.
[38] N. Metropolis, A. W. Rosenbluth, M. Rosenbluth, A. H. Teller and E. Teller, Equation of State Calculations by Fast Computing Machines,*J. Chem. Phys.* **21** (1983) 1087-1092.
[39] D. Shepard, A two-dimensional interpolation function for irregularly-spaced data, in *Proc. of the 23rd ACM National Conf. (ACM '68)* (USA, NY, 1968), pp. 517-524.
[40] Y. Yu and Z. H. Zhou, On the usefulness of infeasible solutions in evolutionary search: A theoretical study, in *Proc. of the IEEE Congress on Evol. Comput.* (Hong Kong, 2008), pp. 835–840.
[41] D. Coit and A. Smith, Penalty guided genetic search for reliability design optimization, *Comput. Ind. Eng.* **31**(4) (1996) 895–904.
[42] P. Diego and B. Baran, Solving Multiobjective Multicast Routing Problem with a new Ant Colony Optimization Approach, in *Proc. of the 3rd Int. IFIP/ACM Latin American Conf. on Networking*, eds. ACM (USA, NY, 2005), pp. 11-19.
[43] T. Back, D. Fogel and Z. Michalewicz, *Handbook of Evolutionary Computation*, 1st edn. (IOP Publ. Ltd., Bristol, UK, 1997).

Nuno Abreu is a Ph.D. student at the

Electrical and Computer Engineering Department of the Faculty of Engineering of Porto University (Portugal) and researcher at the Robotics Unit at INESC TEC. His thesis focuses in mine countermeasures mission planning for autonomous underwater vehicles. In 2008 he received the degree of MSc in Electrical and Computer Engineering from the University of Porto. His research activity is mainly focused on robotics in research topics such as path planning and artificial intelligence.

Aníbal Matos received a Ph.D. in Electrical and Computer Engineering form Porto University in 2001. He is currently a researcher in the Robotics Unit at INESC TEC and also an assistant professor at the Faculty of Engineering of Porto University. His main research interests are related to marine robotics and control systems. Aníbal Matos has been and is currently involved in several research and development projects related to marine robotics and its application to several fields, funded by national and international agencies. He has also participated in the development of modular autonomous underwater vehicles for environmental monitoring and underwater inspection.