# Online Direct Policy Search for Thruster Failure Recovery in Autonomous Underwater Vehicles

**Seyed Reza Ahmadzadeh, Matteo Leonetti, Petar Kormushev** [1]

**Abstract.** Autonomous underwater vehicles are prone to various factors that may lead a mission to fail and cause unrecoverable damages. Even robust controllers cannot make sure that the robot is able to navigate to a safe location in such situations. In this paper we propose an online learning method for reconfiguring the controller, which tries to recover the robot and survive the mission using the current asset of the system. The proposed method is framed in the reinforcement learning setting, and in particular as a model-based direct policy search approach. Since learning on a damaged vehicle would be impossible owing to time and energy constraints, learning is performed on a model which is identified and kept updated online. We evaluate the applicability of our method with different policy representations and learning algorithms, on the model of the Girona500 autonomous underwater vehicle.

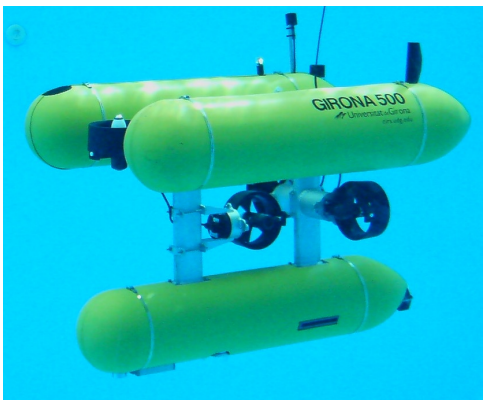


**Figure 1.** The Girona500 AUV with 5 thrusters

## 1 INTRODUCTION

Autonomous Underwater Vehicles (AUVs) have to deal with long missions in unknown environments. The risk of damage is of severe concern, owing to many factors, e.g. extreme pressure, corrosive effects of sea water, the risk of damage due to waves while on the surface, and collisions during launch or later. Improving the reliability of the robot is therefore substantial, especially for autonomous vehicles. Fault Detection, Isolation, and Reconfiguration (FDIR) are important and challenging problems for autonomous vehicles. If the fault is detected and identified, appropriate reconfiguration control actions may be taken [1]. The first part of the FDIR process, the Fault Detection and Identification (FDI) problem, consists of making a binary decision - either something has gone wrong or not - and of determining the location as well as nature of the fault. The developed and implemented FDI methods for Remotely Operated underwater Vehicles (ROVs) and AUVs aim at generating robust error residuals that are insensitive to noise and uncertainties, while sensitive to faults [1]. These methods can be grouped into a few basic approaches as: observer-based methods [27], parity relation approaches [31], optimization-based approaches, Kalman filter-based approaches [15], stochastic approaches, system identification approaches, nonlinear approaches, hybrid system approaches, and artificial intelligence methods.

The reconfiguration step of the FDIR process, which is the focus of this paper, involves changing the controller in response to the detected fault, in order to ensure safe or satisfactory operation of the system. There are various methods of control reconfiguration, such as those based on online learning or system identification. Many control reconfiguration methods are based on techniques for fault detection and isolation. Multiple-model approaches [18] and adaptive control approaches [21] are two examples. In multiple-model approaches, a bank of parallel models is used to describe the subsystem under normal operating mode and under various fault conditions, such as thruster failure. A corresponding controller is designed for each of these models. A suitably chosen switching mechanism is designed to determine the mode of the system at each time step, and to select the corresponding controller designed for that mode. Another common approach in reconfigurable control is to utilize an adaptive controller to ensure robust or acceptable level of performance under abrupt changes in system parameters. In adaptive controllers, some parameters of the system are tuned. In this paper, we take a different stand and devise a new controller by reconfiguring the functional resource of the system. We use online learning approaches based on evolution strategy and reinforcement learning techniques [29].

Thruster failure recovery is one of the most challenging tasks which is defined in PANDORA project [20]. The main goal of PANDORA is to respond to system faults persistently and act appropriately under unexpected environmental changes and extreme uncertainties.

In this paper, we illustrate the capabilities of our approach on Girona500 [25]. Girona500 is a reconfigurable AUV equipped with typical navigation sensors (DVL, AHRS, pressure gauge and USBL) and basic survey equipment (profiler sonar, side scan sonar, video camera and sound velocity sensor) which is used in PANDORA. In the layout we used, the AUV is equipped with 5 thrusters (Figure 1): two vertical thrusters to actuate the heave, one lateral thruster for the sway, and two horizontal thrusters for the surge and the yaw.

[1] Authors are with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego, 30, 16163, Genova, Italy, E-mail: {reza.ahmadzadeh, matteo.leonetti, petar.kormushev }@iit.it

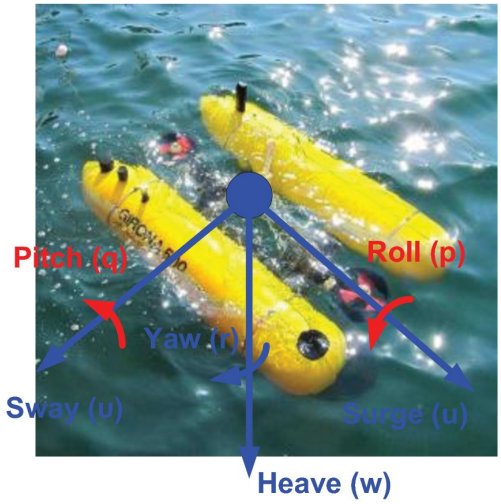


**Figure 2.** The Girona500 AUV. The blue color indicates actuated DoFs and the red color indicates underactuated DoFs.

## 2 AUV MODEL

According to the standard modeling procedure of underwater vehicles [5], the Girona500 AUV can be modeled as a rigid body subject to external forces and torques. Also we have to consider the actuated and underactuated DoFs of the AUV that can be seen in Figure 2.

$$\begin{aligned} M\dot{\mathbf{v}} + C(\mathbf{v})\mathbf{v} + D(\mathbf{v})\mathbf{v} + g(\eta) &= \tau \\ \dot{\eta} &= J(\eta)\mathbf{v} \\ \tau &= Bu \end{aligned} \tag{1}$$

where:

- $M = M_{RB} + M_A$, where $M_{RB}$ and $M_A$ are the inertia matrix of a rigid body and the added mass respectively;
- $C(\mathbf{v}) = C_{RB}(\mathbf{v}) + C_A(\mathbf{v})$ , where $C_{RB}(\mathbf{v})$ and $C_A(\mathbf{v})$ are the Coriolis and centripetal matrix for a rigid body and the added mass respectively;
- $D(\mathbf{v}) = D_{quad}(\mathbf{v}) + D_{lin}(\mathbf{v})$ , where $D_{quad}(\mathbf{v})$ and $D_{lin}(\mathbf{v})$ are the quadratic and linear drag matrix respectively;
- $g(n)$ is the hydrostatic restoring force vector;
- $J(\eta)$ is the Jacobian matrix transforming the velocities from the body-fixed to the earth-fixed frame;
- $\eta = [x\ y\ z\ \phi\ \theta\ \psi]^T$ is the pose (position and orientation) vector;
- $\mathbf{v} = [u\ v\ w\ p\ q\ r]^T$ is the body velocity vector;
- $\tau$ is the force/torque vector.
- $u$ is the input vector.
- $B$ is the thruster control matrix.

While the vehicle is not underactuated, most approaches operate on the matrix $B$, to obtain the required forces/torques by reallocating the command on the functional thrusters. The simplest way to modify $B$ is ignoring the columns correspondent to the faulty thrusters. We proposed a different approach by computing a new command function $u$ to reach a given target without modifying the matrix $B$. Our approach is feasible in underactuated vehicles as well.

## 3 METHODOLOGY

We frame our method in the context of model-based policy search for reinforcement learning [29]. This framework comprises a dynamic model of the vehicle.1, a parameterized representation for the control law, a cost function, and an optimization algorithm. The control law is represented as a function $\pi(t|\theta)$ of time (called *policy*) depending on a parameter vector $\theta$. In this work the policy is represented with a linear function approximator, that is a function of the form $\pi(t|\theta) = \theta^T \phi(t)$, using the Fourier basis functions for $\phi$ [11]. The output of the policy is the voltage for the active thrusters. The performance of the vehicle is measured through a cost function:

$$J(\theta) = \sum_{t=0}^{T} c_t(s_t)\Big|_{\pi(t|\theta)} \tag{2}$$

where $c_t$ is the immediate cost, and depends on the current state, which in turn is determined by the policy and its parameters. Therefore, the aim of the agent is to tune the policy's parameters in order to minimize the cumulative cost $J$ over a horizon $T$. We implemented two global derivative-free optimization algorithms to find an optimal policy for this task. Policy gradient approaches can be used as an alternative solution, because they estimate the derivative of the policy with respect to the parameters of the model. The main issue is that the estimation procedure of these approaches is expensive, so derivative-free methods are chosen to be applied in this particular case. The first algorithm is the well-known Simulated Annealing (SA), which solve unconstrained and bound-constrained [10]. The second algorithm which is a modified version of the Price algorithm [23] combines a global controlled stochastic search and a deterministic local search [12]. Since it is a global algorithm, it is particularly suitable for decision problems created on-the-fly and solved online, without the possibility to engineer a good initial solution. In addition, in [9], the modified Price's algorithm was used for online identification of the Girona500 AUV, and we use the identified parameters of the AUV in our dynamics model of the system.

In our scenario, when a thruster is deemed faulty, a function $J$ is created to represent the cost of a path to the target location. The optimization algorithm is then used to compute the minimal policy, in the given policy parametrization, that takes the AUV as close as possible to the target location using only the functional thrusters. The function $\pi$ computed, substitutes the AUV's controller that would work under normal operating conditions. It is also possible to use the target location as a waypoint, by adding a secondary optimization objective (appropriately weighed) to $J$. As will be seen subsequently, the secondary objective enforces the robot to reach the desired point with a given velocity.

## 4 LEARNING ALGORITHMS

Since, in this case, the cost function is not available in closed form; we implement derivative-free optimization algorithms. Also, we do not use policy gradient approaches, because estimating the derivatives of the policy with respect to the parameters of the model is an expensive process. We define the immediate costs, but the total cost, which is what we aim at minimizing, depends on both the policy and the environment (in this case the model). We implement a well-known optimization algorithm, Simulated Annealing [10], to find an optimum policy to the problem of thruster failure recovery for the Girona500 AUV. In addition, we utilize another derivative-free optimization algorithm which is introduced recently by Leonetti et al. [12], to try different derivative-free methods, to evaluate the quality of the solutions found and the time required, to verify their

usability as online methods. This algorithm was used for online identification of the Girona500 AUV [9].

## 4.1 Simulated Annealing

Simulated Annealing (SA) is a probabilistic meta-heuristic which has been proposed in the area of combinatorial optimization [10] that is, when the objective function is defined in a discrete domain. SA is proposed for finding the global minimum of a cost function that may possess several local minima [3]. The method is reported to perform well in the presence of a very high number of variables (even tens of thousands). The SA heuristic was modified in order to apply to the optimization of multi-modal functions defined in continuous domain [4].

SA comes from the Metropolis algorithm, a simulation of the recrystallization of atoms in metal during its annealing (gradual and controlled cooling). During annealing, atoms migrate naturally to configurations that minimize the total energy of the system, even if during this migration the system undergoes high-energy configurations. The observation of this behavior suggested the application of the simulation of such a process to combinatorial optimization problems [30].

The choice of the temperature or cooling scheduling and the next candidate distribution are the most important decisions in the definition of an SA algorithm [19]. The temperature schedule defines how the temperature in SA is decreased. There exists a wide range of methods to determine this temperature schedule [22]. Corana et al. [4] proposed a self-tuning SA algorithm the step size of which is configured in order to maintain a number of accepted solutions. The probability distribution used by Corana et al. is flat. Ingber [6], on the other hand, proposed to use a Cauchy distribution and a faster temperature decrease. The very fast simulated re-annealing proposed by Ingber and Rosen [8] searches for different sensitivities in parameters by processing the first derivatives of the objective function. In this paper, we use Boltzman temperature scheduling.

The SA algorithm proposed in [30] uses a Gaussian probability distribution with a self-controlled standard deviation in order to maintain the number of accepted solutions. The starting temperature must be hot enough to allow a move to almost any neighborhood state. If this is not done, the ending solution will be very close to the starting solution. However, if the temperature starts at a too high value then the search can move to any neighbor and thus transform the search into a random search. Effectively, the search will be random until the temperature is cool enough to start acting as an SA algorithm.

Several areas of application of SA can be enumerated, some of which are: chemistry and chemical engineering [13], image processing [28], economics and finance [7], electrical engineering and circuit design [26], geometry and physics [17], machine learning [16], networking and communication [24], etc.

The well-known SA algorithm is not described in detail here, just a pseudocode listing of the main Simulated Annealing algorithm is provided in Algorithm 1.

**Algorithm 1** Pseudocode for the main Simulated Annealing.

1: Input: Problem Size: $n$, $iterations_{max}$, $temp_{max}$
2: Output: $S_{best}$ *//the best solution*
3: create $n$ initial random solutions : $S_{current}$
4: $S_{best} = S_{current}$
5: **for** $i = 1$ **to** $iterations_{max}$ **do**
6: create neighbor solution from $S_{current} : S(i)$
7: calculate current temperature $temp_{max} : temp_{current}$
8: **if** $cost(S(i)) \leq cost(S_{current})$ **then**
9: $S_{current} = S(i)$
10: **if** $\text{cost}(S(i)) \leq \text{cost}(S_{best})$ **then**
11: $S_{best} = S(i)$
12: **end if**
13: **else if** $\exp(\frac{cost(S_{current}) - cost(S(i))}{temp(S(i))} > rand())$ **then**
14: $S_{current} = S(i)$
15: **end if**
16: **end for**
17: Return $S_{best}$

## 4.2 Modified Price's algorithm

While Simulated Annealing provides our baseline, we perform our experiments additionally with another black-box optimization algorithm [12]. We believe this algorithm has great potential in its application to policy search for robotic reinforcement learning tasks. It also can provide the task of this paper with an evaluation plan. The Price's algorithm is a combination of a global and a local derivative-free method, designed for the optimization of non-linear, multi-modal, multivariate functions. This algorithm is global, derivative-free, and iterative.

The algorithm is divided into two phases: a controlled global random-search phase, and a deterministic local line-search phase. The algorithm used in the global phase has been introduced by Brachetti et al. [2], and we report it in Algorithm 2. The global phase is population-based, and the initial population is drawn at random over D (line 3). It is also possible to add to the initial population any good point known in advance by the designer.

The population at any time is composed by the best $m$ points ever sampled, where $m$ is a parameter of the algorithm. The bigger $m$, the more likely the algorithm is to avoid non-global minima. The algorithm terminates when the difference between the best and the worst point of the population is less than the parameter $\varepsilon$. The population evolves by sampling a random family of $n+1$ points from the population itself, where $n$ is the dimensionality of the domain, and computing the weighted centroid (lines 9–10). The next trial point is computed as a weighted reflection of the worst point of the population with respect to the weighted centroid (line 11).

This algorithm has been proved to converge to the global minimum if uniform random sampling is performed together with the weighted centroid reflection [2]. Since this step guarantees the convergence in the limit by assigning a non-zero probability to the neighborhood of any point on the domain, it often compromises the performance on most functions in practice. Therefore, Leonetti et al. [12] chose not to perform the uniform sampling, and to rely only on the heuristic provided by the centroid reflection. This approach has been extensively numerically evaluated in the literature [12, 2, 23]. Leonetti et al. [12] followed the approach presented by [2] and combined this global search with a deterministic local search. Therefore, instead of employing Algorithm 2 with a very small $\varepsilon$ (typically in the order of $10^{-6}$) they let $\varepsilon = 10^{-2}$, and perform a deterministic local search in the neighborhood represented by the population at the time the global search is terminated. Although, global stochastic search is a powerful method to avoid being trapped in local minima, the random nature of its sampling makes it less effective when the region of the global minimizer has been identified. Thus, the best-point from the first global phase is used as the starting point of the following local search, which employs a coordinate-search algorithm with line-search expansions [14] reported in Algorithm 3.

The algorithm uses positive step sizes $\alpha_j, j = 1, \ldots, 2n$ along the

**Algorithm 2** Controlled Random Search Phase

1: Input: a positive integer $m \geq n+1$, $\varepsilon > 0$
2: $k = 0$
3: compute the initial set: $S^k = \{\theta_1^k, \ldots, \theta_m^k\}$ where the points $\theta_i^k$, $i = 1, \ldots, m$ are chosen at random over a box $D$
4: evaluate $J$ at each point $\theta_i^k$, $i = 1, \ldots, m$.
5: determine the points $\theta_{max}^k$, $\theta_{min}^k$ and the values $J_{max}^k$, $J_{max}^k$ such that: $J_{max}^k = J(\theta_{max}^k) = \max_{\theta \in S^k} J(\theta)$ and $J_{min}^k = J(\theta_{min}^k) = \min_{\theta \in S^k} J(\theta)$
6: **if** $J_{max}^k - J_{min}^k \leq \varepsilon$ **then**
7: STOP
8: **end if**
9: choose at random $n+1$ points $\theta_{i_0}^k, \theta_{i_1}^k, \ldots, \theta_{i_n}^k$ over $S^k$, where $J(\theta_{i_0}^k) \geq J(\theta_{i_j}^k), \quad j = 1, \ldots, n$
10: determine the centroid $c^k = \sum_{j=0}^n w_j^k \theta_{i_j}^k$
11: determine the trial point $\bar{\theta}^k$ given by

$$\bar{\theta}^k = c^k - \alpha^k (\theta_{i_0}^k - c^k)$$

where

$$w_j^k = \frac{\eta_j^k}{\sum_{r=0}^n \eta_r^k}, \qquad \eta_j^k = \frac{1}{J(\theta_{i_j}^k) - J_{min}^k + \phi^k},$$
$$\alpha^k = 1 - \frac{J(\theta_{i_0}^k) - \sum_{j=0}^n w_j^k J(\theta_{i_j}^k)}{J_{max}^k - J_{min}^k + \phi^k}$$

and

$$\phi^k = n \frac{(J_{max}^k - J_{min}^k)^2}{J_{max}^0 - J_{min}^0};$$

12: **if** $\bar{\theta}^k \notin D$ **then**
13: go to 9
14: **else**
15: compute $J(\bar{\theta}^k)$
16: **end if**
17: **if** $J(\bar{\theta}^k) \geq J_{max}^k$ **then**
18: $S^{k+1} = S^k$
19: $k = k+1$
20: go to 9
21: **else**
22: $S^{k+1} = S^k \cup \{\bar{\theta}^k\} - \{\theta_{max}^k\}$
23: $k = k+1$
24: go to 5
25: **end if**

cardinal directions $\{e_1, \ldots, e_n, -e_1, \ldots, -e_n\}$ to search for a point that improves the current best-point starting from $\theta^0$. The parameters are initially set to:

$$\alpha_j = \frac{\text{median}\{\|x^i - x_{\min}^S\|, x^i \in S\}}{2}, j = 1, \ldots, 2n \tag{3}$$

where $S$ is the final population from the previous algorithm. Thus, the parameters $\alpha_j$, that are the initial steps in each direction, are set to half the median of the distance between the points in the population and the current best-point ($x_{\min}^S$). This seamless integration between the global and local phases provides an improvement over the original formulation of this method [12], so that the parameter of the latter can be computed from the population of the former. We considered other measures for the radius of the population, for instance, the mean or the maximum distance, but the median proved to be the most effective in practice. Finally, if the largest step $\max_{i=1,\ldots,2n}\{\tilde{\alpha}_i^k\}$ is smaller than the parameter $\varepsilon$ the algorithm terminates (line 3).

**Algorithm 3** Line Search Phase

1: Input: $\theta^0 \in \mathbb{R}^n$, $\tilde{\alpha}_1^0, \ldots, \tilde{\alpha}_{2n}^0 > 0$, $\sigma \in (0,1)$, $\gamma \in (0,1)$, $\delta > 1$, $\varepsilon > 0$
2: $k = 0$
3: **if** $\max_{i=1,\ldots,2n} \{\tilde{\alpha}_i^k\} \leq \varepsilon$ **then**
4: STOP
5: **end if**
6: $i = 1$, $y_1^k = \theta^k$, $x^k = \theta^k$
7: **if** $J(y_i^k + \tilde{\alpha}_i^k e_i) \leq J(y_i^k) - \gamma(\tilde{\alpha}_i^k)^2$ and $J(y_i^k + \tilde{\alpha}_i^k e_i) < J(x^k)$ **then**
8: $\alpha_i^k = \tilde{\alpha}_i^k$
9: $x^k = y_i^k + \alpha_i^k e_i$
10: **while** $J(y_i^k + \delta\alpha_i^k e_i) \leq J(y_i^k) - \gamma(\delta\alpha_i^k)^2$ and $J(y_i^k + \delta\alpha_i^k e_i) < J(x^k)$ **do**
11: $\alpha_i^k = \delta\alpha_i^k$
12: $x^k = y_i^k + \alpha_i^k e_i$
13: **end while**
14: $\tilde{\alpha}_i^{k+1} = \alpha_i^k$
15: **else**
16: $\alpha_i^k = 0$
17: $\tilde{\alpha}_i^{k+1} = \sigma\tilde{\alpha}_i^k$
18: **end if**
19: $y_{i+1}^k = y_i^k + \alpha_i^k e_i$
20: **if** $i < 2n$ **then**
21: $i = i+1$
22: go to 7
23: **end if**
24: $\theta^{k+1} = x^k$, $k = k+1$, go to 3

Trial points are subsequently generated, along the direction $e_i$ and with steps $\alpha_i$. If a point along a direction $e_i$ starting at $y_i^k$ improves $y_i^k$ of at least $\gamma \cdot (\alpha_i^k)^2$, and also improves the current best-point $x_i^k$, then $\alpha_i^k$ is increased by a factor $\delta$, and a farther point along $e_i$ is tried (lines 7–14). This is called the expansion phase. If the trial point is not sufficiently improving, $\alpha_i^k$ is reduced by a factor $\sigma$ in a contraction phase (lines 16–17), and other directions are polled. Typically $\delta = 2$ and $\sigma = 1/2$. When all the directions have been contracted up to $\varepsilon$ the algorithm terminates. This line search algorithm draws its inspiration from gradient-based methods, and performs optimization along given lines, in this case the coordinate axes. While none of the directions is as fast descending as the gradient, about half of them will be improving directions. The expansion phase, in which the step is increased by a factor $\delta$, allows to proceed along a successful line at an increasing pace, proving very effective in practice. Coordinate-search algorithms poll the function on a finite number of points, and are guaranteed to provide a locally optimum value at the required precision $\varepsilon$. No point closer to the current best-point than $\varepsilon$ is polled.

## 5 EXPERIMENTAL RESULTS

We performed our experiments on the dynamics model of Girona500, whose parameters have been identified in [9]. We limited the trials to the horizontal plane, where we supposed the left-hand forward thruster to be broken. Therefore, the AUV can navigate only with the right-hand forward thruster and the lateral one. Any attempt to change the allocation matrix of the thrusters would be impossible, since the vehicle is underactuated. We used the following definition of the immediate cost:

$$c_t(\langle p_t, v_t \rangle) = \begin{cases} \| p_t - g_p \| & \text{if } t < T \\ w \| v_t - g_v \| & \text{if } t = T \end{cases} \tag{4}$$

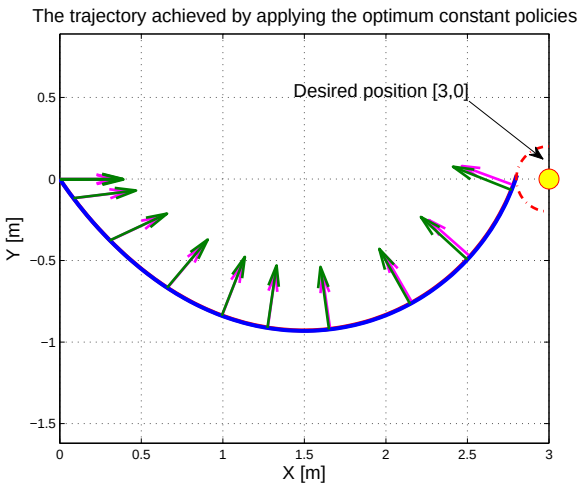


**Figure 3.** Acquired trajectories by the Simulated Annealing (SA) and Modified Price (MP) algorithms for the first experiment with constant policy representation. The area inside the red arc is the acceptable area, $dist < 0.2$. (see Section 5.1 for more details).

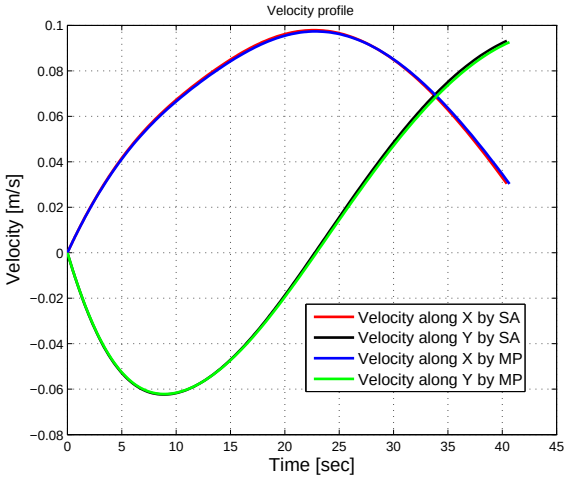


**Figure 4.** Acquired velocities by the Simulated Annealing (SA) and Modified Price (MP) algorithms for the first experiment with constant policy representation (see Section 5.1 for more details).

where the state $s_t = \langle p_t, v_t \rangle$ is composed by position and velocity at time $t$, $g_p$ is the goal location, $g_v$ is the goal velocity and $w$ weighs the velocity objective with respect to the positional one. The first part of the defined immediate cost states that we want to find the shortest-path to the global location $g_p$. So, the algorithm tries to minimize the position of all the points along the path, not only the last state. On the other hand, the second part, which is a secondary objective, is considered at the end state ($t = T$) to minimize the velocity. Thus we provide the second part with a weight coefficient $w$.

For all our experiments we use $T = 60$s, since all the target destinations are reachable in 60 seconds. For the policy representation, we start with constant values and make it more complicated in each experiment. For example, we use an order 3 Fourier expansion, in the third experiment. This way, we can find out the complexity limit of the policy representation that is still operational as an online process. In all the policy representation experiments, we defined a desired position in the vicinity of the robot (at $\langle 3,0 \rangle$), and a target velocity of $\langle 0,0 \rangle$. We also designed the cost function somehow that, when the AUV reaches to an area practically near to the target position ($distance < 0.2meter$), the optimization algorithm will terminate.

## 5.1 Constant Policy

In the first experiment, a constant policy vector is considered, which operates like a constant voltage function, applied on the undamaged thrusters, to move the AUV towards the desired target position. We repeated the optimizing process 50 times for each optimization algorithm.

The trajectories resulting by the application of the optimum constant policies computed by the two algorithms can be seen in Figure 3. The figure shows that the solutions found by the two optimization algorithms are very similar. The acquired velocity profiles can be seen in the Fig. 4. The resulting velocity profiles are also very similar in both cases. Since both algorithms are stochastic, the results may converge to different solutions in different runs. In Table 1 we report the mean, median, standard deviation, and interquartile range for the distributions of the values of the optimal policies over the 50 experiments, together with the number of iterations to compute them. In addition,

**Table 1.** Performance of the two optimization algorithms in the first experiment with constant policy representation. FcnVal represents the best value found for the objective function and the numIter shows the number of Iterations.

| | Modified Price | | Simulated Annealing | |
|---|---|---|---|---|
| | numIter | fcnVal | numIter | fcnVal |
| mean | 139 | 772.25 | 399 | 798.32 |
| median | 132 | 766.97 | 373 | 776.04 |
| std | 41 | 24.23 | 135 | 55.19 |
| iqr | 45 | 1.78 | 139 | 25.63 |

the box plot of the statistical results is depicted in Figure 5. The central mark is the median, the edges of the box are the 25th and 75th percentiles, the whiskers extend to the most extreme data points not considered outliers, and outliers are plotted individually. As it can be seen from the results, it takes longer for Simulated Annealing to achieve the same quality of the solutions as for modified Price. Gen-

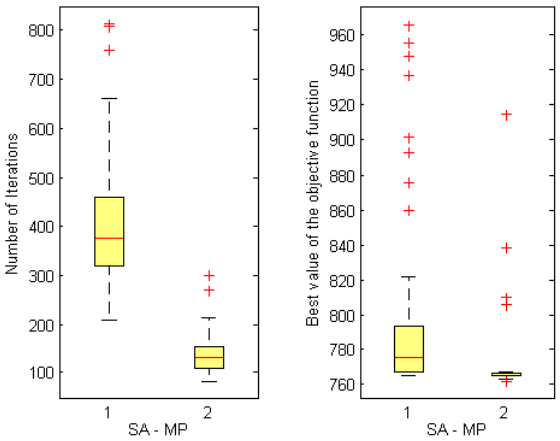


**Figure 5.** Comparing Simulated Annealing (SA) vs. Modified Price (MP) algorithms with constant policy values over 50 experiments (see Section 5.1 for more details).

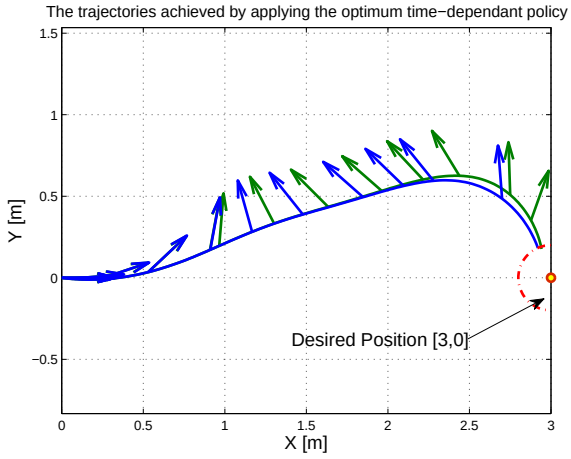


**Figure 6.** Acquired trajectories by the Simulated Annealing (SA) and Modified Price (MP) algorithms for the second experiment with time-dependant policy representation. The area inside the red arc is the acceptable area, $dist < 0.2$. (see Section 5.2 for more details).

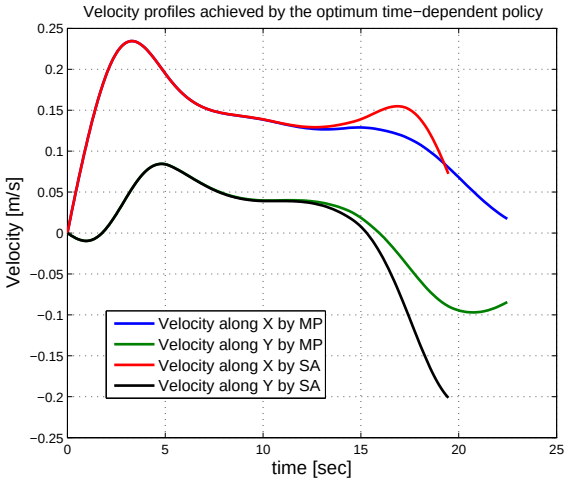


**Figure 7.** Acquired velocities by the Simulated Annealing (SA) and Modified Price (MP) algorithms for the second experiment with time-dependant policy representation (see Section 5.2 for more details).

erally, a constant policy can get the AUV close enough to the target but has little control on the velocity. In order to have the velocity close to a target as a secondary objective we need a more flexible control law.

## 5.2 Time-dependent Policy

In this experiment, we employ an order 3 approximator using the Fourier bases [11] to represent the policy. The policy is dependent on time only, and has 8 parameters to optimize. On the one hand, the target velocity of $\langle 0, 0 \rangle$ becomes more relevant here, as the control law can be more flexible than the constant policy we previously used. On the other hand, while the optimization was only on 2 parameters in the previous experiments (a constant command per each functional thruster), it depends on four times more variables here. In this experiment, we repeated the optimizing process 50 times for each optimization algorithm again. As can be seen in Figure. 6 the acquired trajectories from both algorithms are similar. The velocity profiles, however, are different. As it is shown in Figure. 7, the final part of the velocity profiles acquired by the modified Price algorithm converges to the target velocity more than the Simulated Annealing algorithm. In Table 2 we report the mean, median, standard deviation, and interquartile range for the distributions of the values of the optimal policies over the 50 experiments for the time-dependent policy experiment, together with the number of iterations used to compute them. In addition, the box plot of the statistical results is depicted in

**Table 2.** Performance of the two optimization algorithms in the second experiment using time-dependant policy representation. FcnVal represents the best value found for the objective function and the numIter shows the number of Iterations.

| | Modified Price | | Simulated Annealing | |
|---|---|---|---|---|
| | numIter | fcnVal | numIter | fcnVal |
| mean | 1609 | 412.2 | 496 | 435.24 |
| median | 984 | 384.31 | 421 | 406.18 |
| std | 1822 | 56.1 | 248 | 86.75 |
| iqr | 841 | 115.48 | 351 | 70.65 |

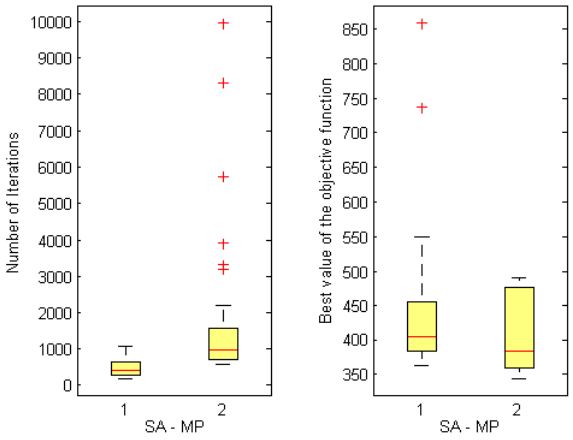


**Figure 8.** Comparing Simulated Annealing (SA) vs. Modified Price (MP) algorithms with time-dependant policy representation over 50 experiments (see Section 5.2 for more details).

Figure 8.

## 5.3 State-dependent Policy

In the last policy representation experiment, we increase the complexity of the representation by including position, orientation and velocities in the observations. An order-3 Fourier policy with 5 state variables (2 positions, 2 velocities, and 1 orientation), produces an optimization problem with 32 variables, which is considered quite significant for policy-search methods. The robustness provided by the reactivity to more information is traded with computational complexity. We repeated the optimizing process 50 times for each optimization algorithm.

As can be seen in Figure. 9, the acquired trajectories from both algorithms are similar. The velocity profiles, however, are different again. As it is shown in Figure. 10, the final part of the velocity profiles acquired by the modified Price algorithm converges to the target velocity more than the Simulated Annealing algorithm. In Table 3 we

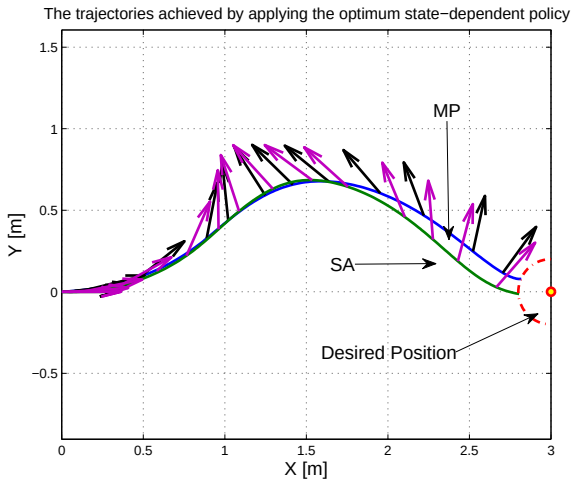


**Figure 9.** Acquired trajectories by the Simulated Annealing (SA) and Modified Price (MP) algorithms for the third experiment with state-dependant policy representation. The area inside the red arc is the acceptable area, $dist < 0.2$. (see Section 5.3 for more details).

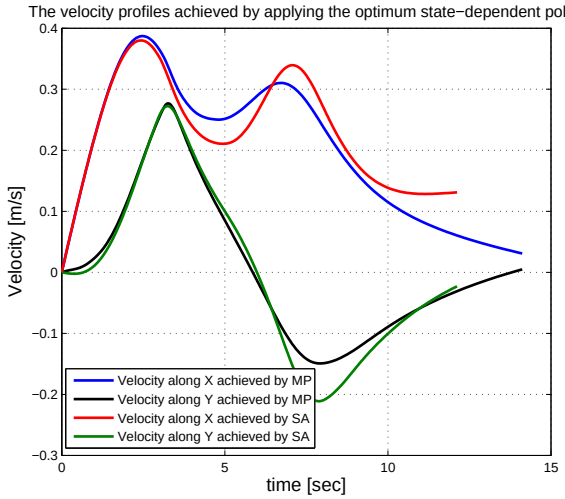


**Figure 10.** Acquired velocities by the Simulated Annealing (SA) and Modified Price (MP) algorithms for the third experiment with state-dependant policy representation (see Section 5.3 for more details).

report the mean, median, standard deviation, and interquartile range for the distributions of the values of the optimal policies over the 50 experiments for the state-dependent policy experiment, together with the number of iterations to compute them. In addition, the box plot of the statistical results is depicted in Figure 11.

## 5.4 Point-to-Point Navigation

In the previous experiments, we focused on the complexity of the policy and the corresponding accuracy in reaching a target position and velocities. In this experiment, we show the point-to-point navigation capability of the approach, using a time-dependent order-3 Fourier policy, which is also valid for other target locations. We generate a grid of target points with coordinates in $[-10, 10]$ at 1 meter distance with respect to each other, and null target velocity. We show

**Table 3.** Performance of the two optimization algorithms in the third experiment using state-dependant policy representation. FcnVal represents the best value found for the objective function and the numIter shows the number of Iterations.

| | Modified Price | | Simulated Annealing | |
|---|---|---|---|---|
| | numIter | fcnVal | numIter | fcnVal |
| mean | 5287 | 217.6 | 511 | 265.89 |
| median | 4214 | 219.5 | 456 | 254.01 |
| std | 2658 | 8.5 | 240 | 39.05 |
| iqr | 1947 | 14.7 | 222 | 16.67 |

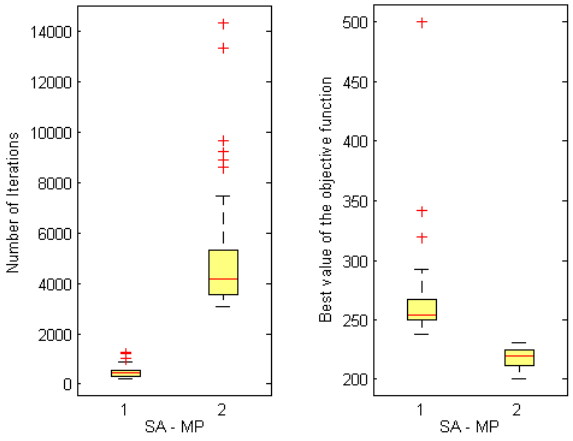


**Figure 11.** Comparing Simulated Annealing (SA) vs. Modified Price (MP) algorithms with state-dependant policy representation over 50 experiments (see Section 5.3 for more details).

some of the generated trajectories in Figure 12. The AUV was able to reach each point at a distance of less than 0.2m and stop there, since the target velocity was the null vector.

## 5.5 Navigating Through Waypoints

Point-to-point navigation is the most common when the objective of the AUV is just to reach a safe location, and possibly float to the surface. The presented method, however, can also be used to follow a trajectory with waypoints. We generate two trajectories to reach a point 50m far, one on a straight line and the other one on an arc of circumference. Along each trajectory, we generate a waypoint every 5m. We iteratively pose the problem of reaching the next waypoint from the current state (position and velocity), with a target velocity pointing towards the subsequent waypoint and norm equal to 0.7, the highest linear velocity for Girona500. The trajectories and the orientation of the AUV are shown in Figure 13. The AUV learns to proceed laterally, using the forward thruster to control the orientation. Sometimes the AUV happens to turn around, but it is always able to recover towards the next waypoint.

## 5.6 Computational Cost

For the feasibility of the presented method we need to assure that the optimization of the policy can be performed on-board in a short time. Figure 14 shows the quality of the solution (distance from the target)computed over time for a waypoint 5m away. The plot is averaged over 20 runs. We used an Intel core 2 Duo P8700 processor,

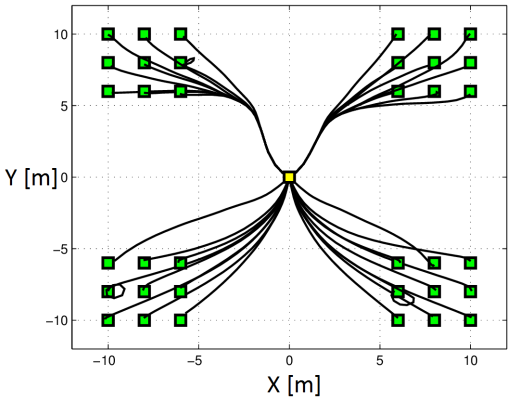


**Figure 12.** The result trajectories for the point-to-point navigation experiment in (XY) plane. The yellow and green points show the initial and final position of the AUV respectively. (see Section.5.4 for more details).

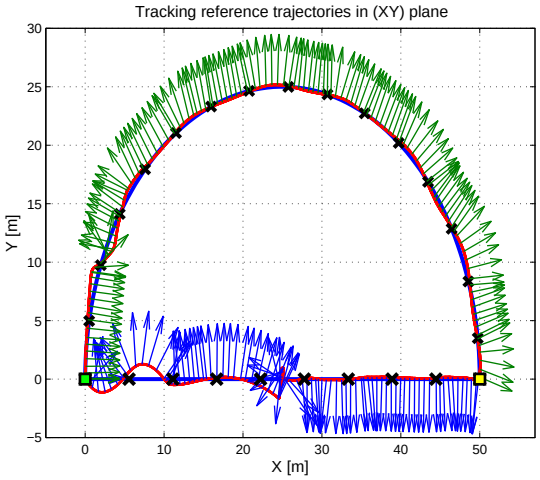


**Figure 13.** The desired and result trajectories in the case that the AUV navigates through waypoints. The blue trajectory is the desired and the red is the acquired trajectory (see Section.5.5 for more details).

where the optimization ran on a single thread. It took 12 seconds to find a solution able to take the AUV only 0.5m from the target. The autonomous vehicle can stop the optimization process at any time after this point, having guaranteed a solution good enough to reach the target. In Figure 15 the green trajectory shows the behavior of the control module of the AUV while one of surge thrusters is failed. The controller cannot recover the AUV to reach the desired position. The result from a fast solution from each of the optimization algorithms are shown in blue and black trajectories. As you can see, the solutions are not optimum but still can recover the AUV to reach the desired position.

## 6 CONCLUSION

In this paper, an online learning method is proposed for the failure thruster recovery task in AUVs. Even robust controllers are unable to recover the robot in such situations, because any attempt to change the allocation matrix of the thrusters for an underactuated vehicle is impossible. The proposed learning method, which is a model-based direct policy search approach, tries to recover the robot and survive

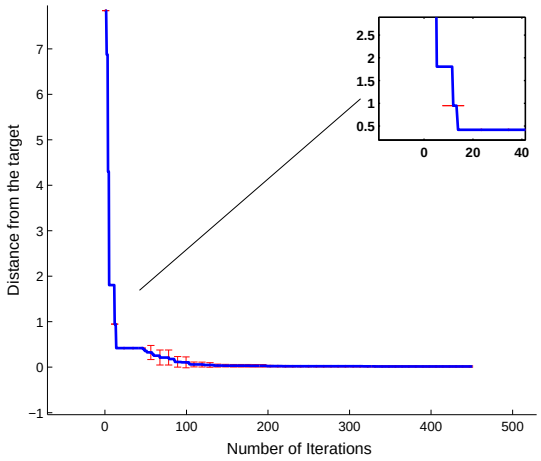


**Figure 14.** Computational cost of the learning method to find the (global) optimum solution (see Section.5.6 for more details).

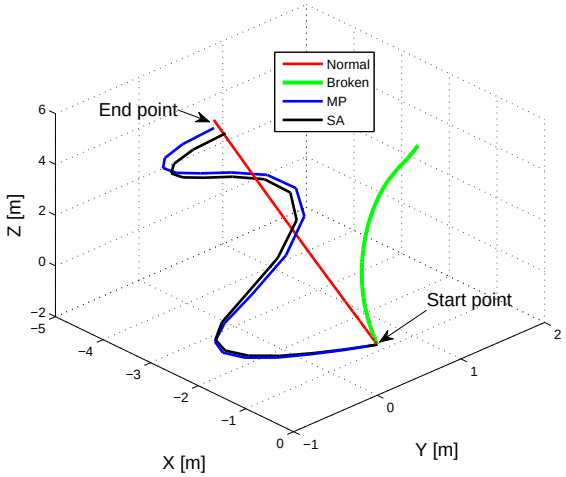


**Figure 15.** Finding a fast solution to reach the desired position for the AUV with a broken thruster (see Section.5.6 for more details).

the mission using the current undamaged thrusters of the system. We evaluated the applicability of our method with different policy representations and learning algorithms. We successfully implemented our approach on Girona500 AUV in simulation.

## ACKNOWLEDGEMENTS

This research was supported by the PANDORA [20] EU FP7 project under the grant agreement No. ICT-288273. Project website: http://persistentautonomy.com/.

## REFERENCES

[1] Gianluca Antonelli, 'A survey of fault detection/tolerance strategies for auvs and rovs', in *Fault diagnosis and fault tolerance for mechatronic systems: Recent advances*, 109–127, Springer, (2003).

[2] P Brachetti, M De Felice Ciccoli, G Di Pillo, and S Lucidi, 'A new version of the price's algorithm for global optimization', *Journal of Global Optimization*, **10**(2), 165–184, (1997).

[3] VLADIMÍR Černỳ, 'Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm', *Journal of optimization theory and applications*, **45**(1), 41–51, (1985).

[4] Angelo Corana, Michele Marchesi, Claudio Martini, and Sandro Ridella, 'Minimizing multimodal functions of continuous variables with the simulated annealing algorithm corrigenda for this article is available here', *ACM Transactions on Mathematical Software (TOMS)*, **13**(3), 262–280, (1987).

[5] T. Fossen, 'Guidance and control of ocean vehicles', *Wiley, New York*, (1994).

[6] Lester Ingber, 'Very fast simulated re-annealing', *Mathematical and computer modelling*, **12**(8), 967–973, (1989).

[7] Lester Ingber, 'Trading markets with canonical momenta and adaptive simulated annealing', Technical report, Lester Ingber, (1996).

[8] Lester Ingber and Bruce Rosen, 'Genetic algorithms and very fast simulated reannealing: A comparison', *Mathematical and Computer Modelling*, **16**(11), 87–100, (1992).

[9] George C. Karras, Charalampos P. Bechlioulis, Matteo Leonetti, Narcs Palomeras, Petar Kormushev, Kostas J. Kyriakopoulos, and Darwin G. Caldwell, 'On-line identification of autonomous underwater vehicles through global derivative-free optimization', in *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, (2013).

[10] Scott Kirkpatrick, D. Gelatt Jr., and Mario P Vecchi, 'Optimization by simulated annealing', *science*, **220**(4598), 671–680, (1983).

[11] George Konidaris, Sarah Osentoski, and Philip S. Thomas, 'Value function approximation in reinforcement learning using the fourier basis', in *AAAI*, (2011).

[12] Matteo Leonetti, Petar Kormushev, and Simone Sagratella, 'Combining local and global direct derivative-free optimization for reinforcement learning', *Cybernetics and Information Technologies*, **12**(3), 53–65, (2012).

[13] Daniel A Liotard, 'Algorithmic tools in the study of semiempirical potential surfaces', *International journal of quantum chemistry*, **44**(5), 723–741, (1992).

[14] Stefano Lucidi and Marco Sciandrone, 'On the global convergence of derivative-free methods for unconstrained optimization', *SIAM Journal on Optimization*, **13**(1), 97–116, (2002).

[15] Rami S Mangoubi, Brent D Appleby, George C Verghese, and Wallace E Vander Velde, 'A robust failure detection and isolation algorithm', in *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 3, pp. 2377–2382. IEEE, (1995).

[16] Horacio Martinez-Alfaro and Donald R Flugrad, 'Collision-free path planning for mobile robots and/or agvs using simulated annealing', in *Systems, Man, and Cybernetics, 1994.'Humans, Information and Technology'., 1994 IEEE International Conference on*, volume 1, pp. 270–275. IEEE, (1994).

[17] Horacio Martínez-Alfaro, Homero Valdez, and Jaime Ortega, 'Linkage synthesis of a four-bar mechanism for n precision points using simulated annealing', in *Asme Design Engineering Technical Conferences/25th Biennial Mecanisms and Robotics Conference. Atlanta.[Links]*, (1998).

[18] Conor McGann, Frédéric Py, Kanna Rajan, John Ryan, Hans Thomas, Richard Henthorn, and Rob McEwen, 'Preliminary results for model-based adaptive control of an autonomous underwater vehicle', in *Experimental Robotics*, pp. 395–405. Springer, (2009).

[19] Mitsunori Miki, Tomoyuki Hiroyasu, and Keiko Ono, 'Simulated annealing with advanced adaptive neighborhood', in *Second international workshop on Intelligent systems design and application*, pp. 113–118. Citeseer, (2002).

[20] PANDORA. Persistent autonomy through learning, adaptation, observation and re-planning. http://persistentautonomy.com/, 2012. PANDORA Europian Project.

[21] Pedro Patrón, Emilio Miguelanez, Yvan R Petillot, David M Lane, and Joaquim Salvi, 'Adaptive mission plan diagnosis and repair for fault recovery in autonomous underwater vehicles', in *OCEANS 2008*, pp. 1–9. IEEE, (2008).

[22] William H Press, Saul A Teukolsky, William T Vetterling, and Brian P Flannery, *Numerical recipes in C+: the art of scientific computing*, volume 994, Cambridge University Press Cambridge, 2009.

[23] WL Price, 'Global optimization by controlled random search', *Journal of Optimization Theory and Applications*, **40**(3), 333–348, (1983).

[24] Alejandro Quintero and Samuel Pierre, 'Assigning cells to switches in cellular mobile networks: a comparative study', *Computer Communications*, **26**(9), 950–960, (2003).

[25] David Ribas, Narcıs Palomeras, Pere Ridao, Marc Carreras, and Angelos Mallios, 'Girona 500 auv: From survey to intervention', *Mechatronics, IEEE/ASME Transactions on*, **17**(1), 46–53, (2012).

[26] Rob A Rutenbar, 'Simulated annealing algorithms: An overview', *Circuits and Devices Magazine, IEEE*, **5**(1), 19–26, (1989).

[27] Alexey Shumsky, Alexey Zhirabok, and Chingiz Hajiyev, 'Observer based fault diagnosis in thrusters of autonomous underwater vehicle', in *Control and Fault-Tolerant Systems (SysTol), 2010 Conference on*, pp. 11–16. IEEE, (2010).

[28] Erik Sundermann and Ignace Lemahieu, 'Pet image reconstruction using simulated annealing', in *Proceedings of the SPIE Medical Imaging95 (Image Processing)*, pp. 378–386, (1995).

[29] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[30] RS Tavares, TC Martins, and MSG Tsuzuki, 'Simulated annealing with adaptive neighborhood: A case study in off-line robot path planning', *Expert Systems with Applications*, **38**(4), 2951–2965, (2011).

[31] KP Venugopal, R Sudhakar, and AS Pandya, 'On-line learning control of autonomous underwater vehicles using feedforward neural networks', *Oceanic Engineering, IEEE Journal of*, **17**(4), 308–319, (1992).