DOI: 10.15439/2016F53
ACSIS, Vol. 8. ISSN 2300-5963

# Employing Game Theory and Computational Intelligence to Find the Optimal Strategy of an Autonomous Underwater Vehicle against a Submarine

Bartłomiej Józef Dzieńkowski
Faculty of Computer Science and Management, Wroclaw University of Science and Technology
Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
Email: bartlomiej.dzienkowski@pwr.edu.pl

Christopher Strode
Centre for Maritime Research and Experimentation (CMRE)
La Spezia, Italy
Email: strode@cmre.nato.int

Urszula Markowska-Kaczmar
Faculty of Computer Science and Management, Wroclaw University of Science and Technology
Wyb. Wyspianskiego 27, 50-370 Wroclaw, Poland
Email: urszula.markowska-kaczmar@pwr.edu.pl

***Abstract*—Game theory is a tool that may be used to model a player as an intelligent being – one who seeks to optimize his own performance while taking into account the performance of his opponent. However, it is often challenging to apply the theory in practice. In the naval environment, this approach may be used, for instance, to find the best strategy for an Autonomous Underwater Vehicle (AUV) while considering the intelligence of the submarine opponent. Classic approaches based on Minimax suffer from an explosion of states, and they are difficult to use in real-time. The paper introduces an approach that improves the Minimax algorithm in a complex naval environment. It assumes limited and scalable computational resources. The approach takes advantage of a flexible utility function based on a neural network with parameters tuned by a genetic algorithm.**

## I. Introduction

AN AUTONOMOUS Underwater Vehicle (AUV) is a robot that travels underwater (Fig. 1)[1]. Compared to other Unmanned Underwater Vehicles (UUVs) such as Remote Operating Vehicles (ROVs), it is not guided by an operator. AUVs are mostly employed in the field of oceanography and are beginning to be considered for military use [2]. For instance, they can be used for patrolling and monitoring the vicinity of a naval port, or for supporting a surface platform in its search for a submarine. An AUV has an advantage over an ROV because it does not reveal its location by continuous communication with an operator.

A submarine is a very difficult opponent to detect because of its ability to exploit the complicated nature of underwater sound propagation. As a result, it can stay hidden for a long time owing to its superior endurance and speed. By comparison, an AUV has limited mobility, sonar capabilities, computational power, and battery life. More importantly, a submarine is commanded by qualified personnel making it a deliberate and intelligent competitor. Therefore, the robot controller must meet high requirements. AUV's software should provide as much intelligent behavior as possible to mitigate its limited resources.

In order to maximize the ability of the AUV to detect the submarine, we consider here a bistatic sonar employment. This means that the sonar source and receiver are separated – rather than collocated on a single platform in the more traditional monostatic case. In this study, the sonar source

The project was commissioned and financed by the Visiting Researcher Programme organized by the Centre for Maritime Research and Experimentation (CMRE) located in La Spezia (Italy) – the Centre is an executive body of NATO's Science and Technology Organization (STO) along with the NATO Collaboration Support Office. The research was partially supported by the statutory funds of the Department of Computational Intelligence, Faculty of Computer Science and Management, Wroclaw University of Science and Technology.

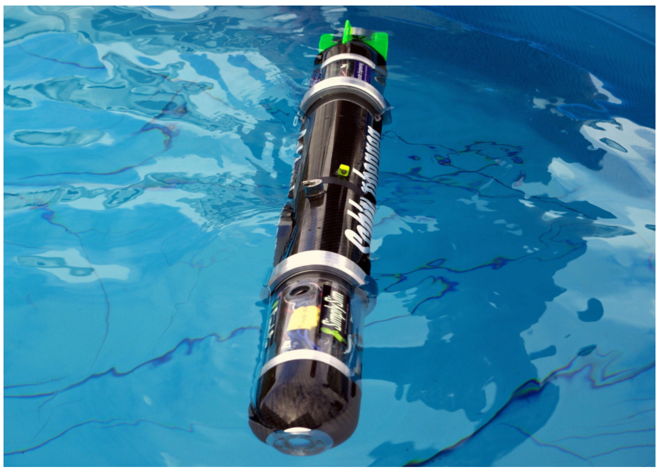
Fig. 1. The picture shows an example of AUV used in military [3]. The Blackghost AUV can attack with no outside control.

is in a fixed location while the receiver is assumed to be a linear array of hydrophones towed by the AUV. Importantly this means that the AUV can receive bistatic sonar contacts – with range and bearing information – without needing to transmit. Sonar transmissions are easily counter detected by a submarine, thereby revealing the location of the asset and allowing the submarine the opportunity to evade.

To achieve an efficient strategy model for an AUV, it must be validated against a challenging foe. Thus, to find the best strategies for both naval units we use game theory [4]. The problem is described as a naval version of a pursuit-evasion game. The players hold different properties; therefore, a skirmish is asymmetrical. It is a multistage non-zero-sum game placed in a complex environment with uncertainty and incomplete information about the game state. The **non**-zero-sum assumption is introduced because the players may have specific and unique objectives. In addition, having a limited access to information about an opponent's state, they might unconsciously cooperate in some cases.

Finding the game equilibrium, which is the problem solution, is a difficult task. In theory, the game should be defined as an extensive-form game. However, adding equivalent state nodes that denote the space of hidden possibilities would greatly increase the problem complexity by extending the game tree, which is already very large. In other words, it is nearly impossible to traverse the entire game tree (extensive or not) in practice. The approach considered here is to employ a pure strategy form of a sequential game with discrete time.

Another complication is that each player receives input that is not readily convertible to a utility value, which expresses how desirable a given state is for a player. There is a significant distance in the state space between actions in the past and their real effect in the future.

In our work, we solve the problem by using a flexible and trained utility function model that is optimized according to specified criteria. A utility function then converts player's input into a utility value of the game state. Both players use the Minimax decision rule to choose the best action. They have their own utility function model that is tuned to gain the best outcome assuming that an opponent is doing the same. In the presented approach, a neural network was chosen as the utility modeling function [5]. Its weights are trained by a genetic algorithm to maximize each player's fitness [6]. The fitness is calculated taking into account the players' objectives.

In the following section, a general overview of the method framework is introduced. Next, a short survey of the existing works related to the stated problem is provided. Subsequently, important properties of the naval environment are described. The following part of the document provides formal foundations and describes the problem as a pursuit-evasion game. Next, naval players are characterized, and their cost functions are defined. A utility function model and its training method are presented in the next part of the document. Finally, results of the experimental study are collated and described.

## II. Method Overview

This section provides a general overview on the proposed approach. It briefly summarizes the method framework and its components (Fig. 2).

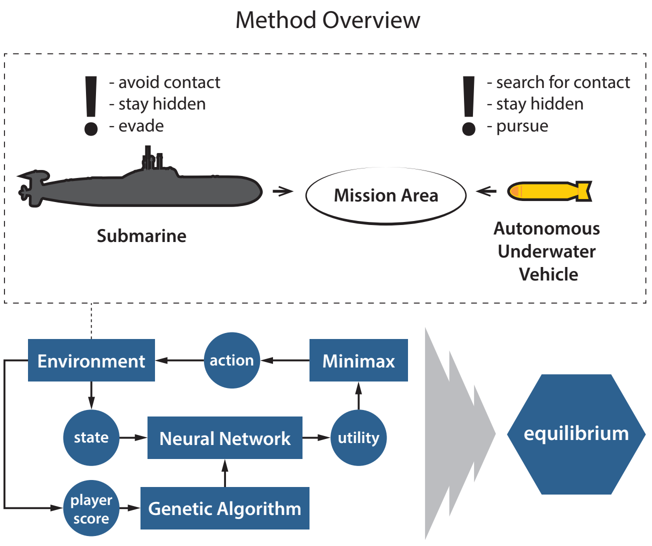

Fig. 2. The picture visualizes the game and the method of solving it.

The proposed method searches for the optimal strategies of players operating in a simulated environment. The problem description uses game-theoretic formalism to model the naval environment as a game. The solution is a saddle-point equilibrium. It is a state in which no player can gain by changing his strategy. In accordance with theoretical foundations, it is assumed that players are rational. The quality of a strategy is represented by its cost value. It is an accumulative cost calculated for a sequence of states from an initial state to a final one.

For the adopted game model, an optimal strategy is expressed by the Minimax decision rule. Application of the algorithm is not straightforward, because a utility value for a given game state is unknown. The utility is a system-wide feedback to the Minimax algorithm. In this approach, a utility value is returned by the output of a multilayer neural network, which is employed as a utility function. Its input is fed with a game state perceived by a player. Each player has a separate neural network, because the players have different capabilities, and they do not share the same view of the environment and its state.

Because the desired (optimal) output of a neural network is unknown, weights inside the network are tuned using an evolutionary approach, rather than the backpropagation learning algorithm. In this method, a genetic algorithm optimizes the cost of the Minimax strategy, which is guided by the output of the neural network. The evaluation procedure requires the game to be simulated over a number of steps. During the optimization process both players improve their strategies until they reach the equilibrium. It is a theoretical state in which the players have found their optimal strategies.

## III. State of the Art

The problem stated in this work is fresh and specific. In the literature, there is a range of works directly addressed to this field but not this particular problem. Most of the papers discuss fundamental issues related to the naval environment [7]. One of the most elementary is modeling of underwater signal propagation and communication [8], [9]. Another is a classic flaming datum problem, in which a fleeing submarine is relocated after momentarily revealing its position [10].

Modern navies are beginning to procure and test multistatic sonar systems at sea. Locations of the sensors are optimized based on a game-theoretic approach, and their efficiency is evaluated in the previous works [11], [12]. Recent technological progress enables us to employ an AUV as a mobile signal receiver instead of using a fixed-position sensor [13]. Despite the fact that an AUV can be considered as a more effective tool against a submarine, the topic is rarely undertaken so far. Related works are focused on the problem of building a probability density map of possible locations of an opponent [14]. Many of the approaches employ simplified behavioral models [15], rather than considering the best strategies for both players and optimizing towards the equilibrium.

Solutions designed for similar problems are often bounded to their domains and cannot be directly applied to this specific case [16]. They are intended for less demanding environments. In the air, aerial vehicles can easily communicate with the command and exchange information. They rely on radar and visual information, which can be processed more efficiently than sonar input. Underwater communication is slow, has a very limited bandwidth, and above all, using it immediately exposes the location of a vehicle to an opponent. Underwater vehicles exploit properties of underwater signal propagation to remain stealthy. They must plan their route very carefully while drones can maneuver more freely, because their environment is mostly uniform. A very limited computational power makes it impossible to run an expensive optimization process on-board while the requirements regarding the method are still high.

The discussed problem is focused on tracking and spying an intelligent enemy unit, rather than patrolling or engaging in combat according to a clearly defined protocol. The task is specific, and it consists of a range of problems that are not addressed by the works related to autonomous vehicles.

The proposed method is based on a known idea of employing an evolutionary algorithm for finding an optimal strategy in game theory [17], [18]. In these works, a population of agents is optimized to achieve the best performance against an opponent who is following a defined strategy. However, that approach cannot be employed in this case, because the behavior of an opponent cannot be easily described by a closed set of rules. Here, both competing sides are optimized simultaneously. To avoid executing an expensive optimization process on-board, the optimization is aimed at tuning a neural network that is used for evaluating the game state and computing utility (reward). The neural network plays the role of a utility function in an on-board decision process. In the light of the above facts, the problem setup and solution are considered as original.

## IV. Naval Environment

The underwater environment in which our scenario takes place is both harsh, from a technological standpoint, and complex in terms of the physics that govern the propagation of sound used to detect a submarine [8]. The performance of a sonar system is a complex function of transmission loss, reverberation levels and noise levels, all affected by various oceanographic, surface, and bottom parameters. Not least of which is the sound speed profile within the water column which causes a bending of the sound propagation paths and can result in large regions of water from which sound may be diverted. These shadow zones may then be exploited by an intelligent submarine greatly decreasing its chance of being detected.

The eventual signal-to-noise ratio (SNR), the primary metric used to assess the probability of detecting a submarine by sonar, depends on water temperature and salinity, surface roughness, depth, and shape of the sea bottom. This study considers the use of multistatic sonar whereby performance is a function of the specific geometry between the separated sonar source and receiver (in this case the AUV) together with the submarine location. As a result, detailed (and time consuming) acoustic propagation models are required to accurately predict multistatic sonar performance within range-dependent environments.

Irrespective of the complexities of actually detecting a submarine, the simple operation of an AUV in the ocean also has its challenges. In addition to basic underwater physics, the AUV has limitations imposed on its motion trajectory due to the stability of its receiving array of hydrophones [19]. The complicated propagation effects already discussed also serve to limit the ability of the AUV to send and receive messages through the water. This is in fact a driving force behind the need for better autonomous decision making – since the vehicle cannot rely on regular intervention by operators.

The goal of this study is to provide a proof-of-concept and consequently, some simplifications are introduced to avoid vast computations, which significantly improves the computation time of the experiments. The system is modeled in such a manner that a general characteristic of the underwater environment is captured. Earlier experiments have shown that the acoustic model is a bottleneck for the calculations. It was replaced by a set of rules that cover only the general features.

The operational area is rectangular, and it contains the players' start positions, signal source position, and mission area, which is circular (Fig. 3). It is assumed that the mission area holds an objective that is important to a submarine. The objective is accomplished if the unit closes to within a defined radius of the goal. In the meantime, the AUV's objective is to keep as close as possible to the submarine. While the AUV is

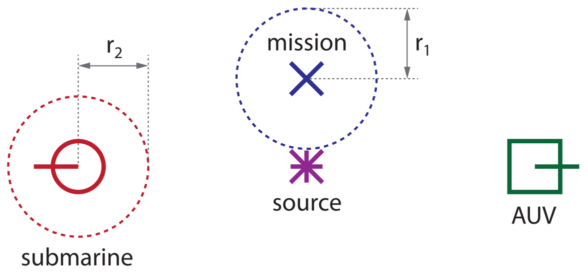

Fig. 3. The image presents an example of game arrangement. The objective of a submarine is to travel to the mission area. The AUV's objective is to be in close surroundings of the opponent. The nearer a unit approaches the signal source, the easier it can be detected.

not intended to prosecute the submarine, thereby preventing it reaching its mission goal, it should nevertheless detect its presence and alert the defenders as soon as possible. Both players benefit from staying undetected. However, the players have different priorities, so the zero-sum property is not valid in this case.

## V. Game Theory

The pursuit-evasion game (PEG) is a well-known problem in the class of differential games in game theory [20]. It is often referred to as a simple lion-zebra or cop-robber case.

Let us define a naval version of PEG as a state-feedback discrete-time dynamic game with two non-cooperating players, where $P_1$ is a pursuer (AUV) and $P_2$ is an evader (a submarine) executing actions sequentially in each game stage $k$, Eq. 1:

$$a(k) = \begin{cases} a_k^1 & \text{if } k \text{ is odd} \\ a_k^2 & \text{if } k \text{ is even} \end{cases}, \tag{1}$$

where $a(k)$ is an action at stage $k$, $a_k^1$ is $P_1$'s action, and $a_k^2$ is $P_2$'s action. The game corresponds to dynamics of the form (Eq. 2):

$$s_{k+1} = \Delta_k\Big(s_k, a(k)\Big),\ \forall k \in \{1, 2, \ldots, K\}, \tag{2}$$

where:

- $s_k$ is an entry state node at stage $k$,
- $\Delta_k$ is a transition function modeling dynamics at stage $k$,
- $K$ is a finite time horizon.

A finite horizon stage additive cost is (Eq. 3):

$$\sum_{k=1}^{K} c_k\Big(s_k, a(k)\Big), \tag{3}$$

that $P_1$ wants it to minimize, and $P_2$ wants it to maximize. A state-feedback information structure corresponds to policies of the form (Eq. 4):

$$a_k^1 = \gamma_k(s_k), \quad a_k^2 = \sigma_k(s_k), \tag{4}$$

where $\gamma$ and $\sigma$ are state-feedback policies for $P_1$ and $P_2$, respectively. The corresponding value of the cost in Eq. 3 for the policies is denoted by $C(\gamma, \sigma)$. A saddle-point pair of equilibrium policies $(\gamma^*, \sigma^*)$ satisfies (Eq. 5):

$$C(\gamma^*, \sigma) \leq C(\gamma^*, \sigma^*) \leq C(\gamma, \sigma^*),\ \forall \sigma, \gamma. \tag{5}$$

For games with a finite state space, an optimal policy cost can be found using Minimax Theorem. It is solved algorithmically. However, the players perceive the state differently, and they do not have strictly opposite objectives. Therefore, the game is not considered as zero-sum. For this reason, Alpha-Beta pruning, which would reduce the complexity, cannot be applied [21].

## VI. Naval Players

Each player has a set of discrete move actions to choose in his turn. For the sake of simplicity, a move action can be executed with a finite number of speeds and headings (Eq. 6):

$$a(k) \in \{\vec{m}_0, \vec{m}_1, \ldots \vec{m}_n\},\ \vec{m} = s * \vec{h}, \tag{6}$$

where $\vec{m}$ is a move vector as a product of scalar speed $s$ and unit heading vector $\vec{h}$. In order to avoid inaccuracy and provide more flexibility, player coordinates are expressed by floating point values rather than grid cells. Units cannot currently change depth.

To imitate sonar features, both detection and counter detection ranges were introduced. A naval player does not have access to information about his opponent as long as the distance between them is bigger than the opponent's detection range. The detection range is not constant, and it changes depending on the distance to the signal source. The closer to the source, the more acoustic energy is reflected by a naval unit and so it is easier to detect. The signal source can be received from a far distance and, therefore, its position is always known to all players.

At some point, the algorithm must estimate the opponent's actions to calculate utilities of the future states. Unfortunately, it cannot be easily done if a player does not know the exact position and heading of the opponent. To deal with the problem an approximate map of possible locations of the opposing unit is generated based on its operational range (Fig. 4). The operational range is a circular area placed in the center of the last revealed position with a radius equal to the maximum distance the unit could travel since the last contact time. The map of possible states of an opponent is limited by the complexity of the algorithm.

## VII. Cost

A cost value describes how well a player performed during the whole game. AUV is referred to as a pursuer whose behavior is characterized by minimizing the distance to its opponent. The pursuer minimizes cost for staying within a specified range to an evader. The robot is penalized if

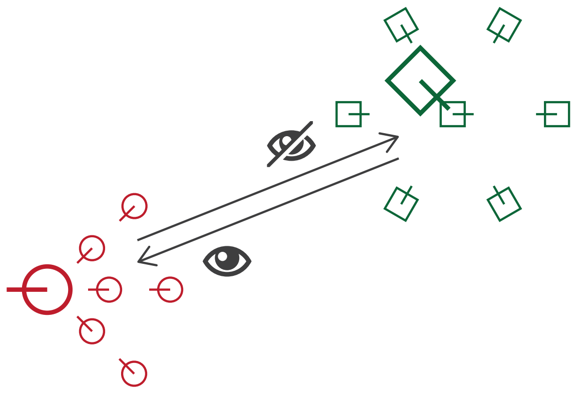

Fig. 4. The picture shows how the algorithm handles limited access to information about the current state of the opponent. The big circle on the left is the actual location of the submarine. The rectangle on the opposite side is the AUV. Small shapes represent their possible positions in the next game step. The state of the player on the left is exposed while the opposite player stays unrevealed, but his last position is known. A map of several possible positions inside the operational range of the hidden player is generated.

its position is revealed. Formally, the cost for the AUV is calculated by the following formula (Eq. 7):

$$C_1(k) = C_1(k-1) + d_o + \\ - \begin{cases} b_1 & \text{if } d_o \le r_1 \\ 0 & \text{otherwise} \end{cases} + \begin{cases} p_1 & \text{if } v_1 = 1 \\ 0 & \text{otherwise} \end{cases}, \quad (7)$$

where:

- $k$ is a game stage number,
- $C_1(\cdot)$ is the pursuer's (AUV) cost function,
- $d_o$ is a distance to the opponent,
- $b_1$ is an award given to the pursuer if distance $d_o$ is smaller than threshold $r_1$,
- $p_1$ is a penalty if the pursuer reveals his position to the opponent,
- $v_1$ is the pursuer's state of visibility to the opponent.

By analogy, the submarine is referred to as an evader who maximizes the distance to his opponent and moves towards the mission area. If the unit is sufficiently close to the mission, a positive cost is awarded. Whenever a player is exposed to his opponent, a penalty cost is applied. In order to clarify, the cost for the submarine is calculated by the following formula Eq. 8:

$$C_2(k) = C_2(k-1) + d_o - d_m + \\ + \begin{cases} b_2 & \text{if } d_m \le r_2 \\ 0 & \text{otherwise} \end{cases} - \begin{cases} p_2 & \text{if } v_2 = 1 \\ 0 & \text{otherwise} \end{cases}, \quad (8)$$

where:

- $C_2(\cdot)$ is the evader's (submarine) cost function,
- $d_m$ is a distance to the mission center,
- $b_2$ is an award given to the evader if distance $d_m$ is smaller than threshold $r_2$,
- $p_2$ is a penalty if the evader reveals his position to the opponent,
- $v_2$ is the evader's state of visibility to the opponent.

Cost parameters $b, r$, and $p$ affect the behavior of players by defining objectives and their relative importance. Distance effects included in the cost functions serve to steer the training towards the desired outcome. Ultimately, award and penalty values drive the tactics of a player, and they can be adjusted to observe different behaviors.

## VIII. Utility

The utility value of the game state is calculated by a neural network based on information held by a player. Among many benefits of neural networks is an application for non-linearly separable problems and generalization of acquired information [22]. In this study, Multi-layer Perceptrons (MLP) were used as a popular model often applied in a broad class of problems [5]. The neural network has three layers. It gives one hidden layer with eight neurons. The activation function is the hyperbolic tangent.

The model is fed by eight inputs provided by a player:

- angle and distance to the mission area,
- angle and distance to the signal source,
- angle and distance to the opponent,
- heading relative to the opponent's heading,
- opponent contact – informs whether the opponent is revealed to a player or one of his possible states should be considered.

The input contains only relative values to ensure that as much general information as possible is acquired by the neural network. Thus, the training process is more efficient and less constrained to a particular case.

### A. *Minimax*

Each player uses a classic Minimax algorithm with a limited depth of the state tree to choose the best action (Alg. VIII-A).

## IX. Evolutionary Training

The utility function model (a neural network) is trained by a classic variant of a genetic algorithm [6]. It is a powerful tool suitable for complex optimization problems characterized by many local extrema. The algorithm optimizes a vector of neural network weights considered as a chromosome without further encoding. To evaluate an individual in a population, it must be decoded to his phenotype level, which means that a neural network is created based on his chromosome.

The optimization process is conducted according to the fitness function that is equal to the cost calculated for each player (Eq. 7 and Eq. 8). Because the players have different goals and fitness functions, the genetic algorithm (Alg. IX) holds two populations – one for each player type. An individual cannot be evaluated without an opponent. Therefore, these two populations periodically pass the copies of their best entities after a defined number of GA iterations. Each population adds a new opponent to a list, and the final fitness is the average cost achieved against a set of foes in the list. This is done to

**Algorithm 1** Minimax ( node, player, depth )

```
    if depth $\leq 0$ then
 2:   ComputeUtilities ( node )
      return $nil$
 4: else if IsRevealed ( player ) then
      actions $\leftarrow$ GetActions ( player )
 6: else
      actions $\leftarrow$ GetPossibleStates ( player )
 8: end if
    nextPlayer $\leftarrow$ (player + 1) % playerCount
10: bestChild $\leftarrow nil$
    for child in Expand ( node, actions ) do
12:   Minimax ( child, nextPlayer, depth - 1 )
      if bestChild = $nil \vee$ GetUtility ( bestChild, player ) $<$ GetUtility ( child, player ) then
14:     bestChild $\leftarrow$ child
      end if
16: end for
    if bestChild = $nil$ then
18:   ComputeUtilities ( node )
    else
20:   CopyUtilities ( node, bestChild )
    end if
22: return bestChild
```

**Algorithm 2** GeneticAlgorithm ( )

```
    populations $\leftarrow \{$ evaders, pursuers $\}$
 2: for pop in populations do
      Evaluate ( pop )
 4: end for
    while g++ $<$ generations do
 6:   if g mod passBestPhase = 0 then
        PassBest ( populations )
 8:   else
        for pop in populations do
10:       Select ( pop )
          Cross ( pop )
12:       Mutate ( pop )
        end for
14:   end if
      for pop in populations do
16:     Evaluate ( pop )
      end for
18: end while
```

optimize against a variety of enemy strategies rather than a single one.

## X. Experiment

The experimental study includes a series of tests carried out to validate the approach. This section describes one of those experiments that was aimed to check the characteristics of a medium-long training process. Thus, it should be emphasized that in the best case scenario obtained results may only represent a near-optimal solution. This study has been preceded by a series of short experiments to select the training parameters. Because of the scale of the problem and the large amount of data, this section includes only selected results regarding the optimization process of the players' strategies.

Taking into account the nature of the research, realistic units of measure have not been preserved. They were adjusted to obtain easily observable behaviors and test the approach. In the experiment, a game arrangement is the same as presented in Fig. 3. However, the picture does not show an additional free space below and above the signal source placed in the center of the stage. A single game was simulated for a limited number of turns that was sufficient for both players to fulfill their objectives. A realistic degree of asymmetry between the players was set by giving to the submarine a better speed and the AUV greater stealth. Another difference is that the submarine is more interested in being undetected than the AUV. More detailed parameters of the experiment setup are provided in Tab. I.

Training results are shown in Fig. 5 and Fig. 6. Surprisingly, training of both populations begins from a relatively high level of fitness. Notwithstanding, an initial population is random, and there is always a chance that one of population entities will accidentally advance to the goal. It is, however, a chaotic behavior, and it changes drastically depending on the input. Thus, any opponent reaction can cause that change.

Further fitness decreases are caused by filling the list of opponents with better enemies. New opponents are added to the list every 20 generations, which is referred to as the pass best phase in the algorithm. Sometimes the period between the transfers of the best entities is too short to invent a better opponent. Nevertheless, it prevents from overtraining to a current set of opponents, which evolve quickly. Forcing an entity to achieve the best average cost against a number of opponents allows it to acquire a more general strategy but may also cause conservative and protective behaviors.

While observing behaviors of both players at the end of the training, it turned out that they acquired interesting strategies. The submarine does not immediately proceed to the mission. First, it glides far below the source, where it has a good cover. Next, it quickly travels toward the mission. It can be said that the submarine takes advantage of its speed and lures the robot. The AUV is not able to catch up with the submarine nor detect the opponent near the mission, because the submarine visits the mission only for a short moment at the end of the game. On the other hand, the AUV tries to follow the trail of the submarine. The robot minimizes its distance to the opponent but never reaches a sufficient distance to receive an award.

During the experimental study, a number of long training runs have been conducted. However, the simulation progress and outcome turned out to be very sensitive to the random nature of the genetic algorithm. Therefore, it was difficult to observe regularity, and averaging results did not lead to clear conclusions. The results from long training runs are placed in Fig. 7 and Fig. 8. For some of the experiments, fitness tend to stabilize in time (Fig. 7) while in other cases, the balance was lost and the fitness of both populations fluctuated (Fig. 8).

**Minimax**

| | | | |
|---|---|---|---|
| tree depth: 3 | max. actions: 7 | hidden opp. states: 7 | game turns: 25 |

**Genetic Algorithm**

| | | | |
|---|---|---|---|
| pop. size: 100 | passBestPhase: 20 | selection: tourn. | tourn. size: 10% |
| elitist model | cross: uniform | cross prob.: 0.6 | cross factor: 0.5 |
| mut. prob.: 0.1 | mut. factor: 0.01 | opp. list: 10 | |

**Evader**

| | | | |
|---|---|---|---|
| max. velocity: 2 | max. turn angle: $\frac{\Pi}{4}$ | detection range: 15 | bistatic |

**Pursuer**

| | | | |
|---|---|---|---|
| max. velocity: 1 | max. turn angle: $\frac{\Pi}{4}$ | detection range: 5 | bistatic |

**Cost parameters**

| | | | | | |
|---|---|---|---|---|---|
| $r_1 = 4$ | $b_1 = 100$ | $p_1 = 50$ | $r_2 = 4$ | $b_2 = 100$ | $p_2 = 10$ |

TABLE I
THE TABLE SHOWS A DETAILED PARAMETER SETUP OF THE EXPERIMENT.

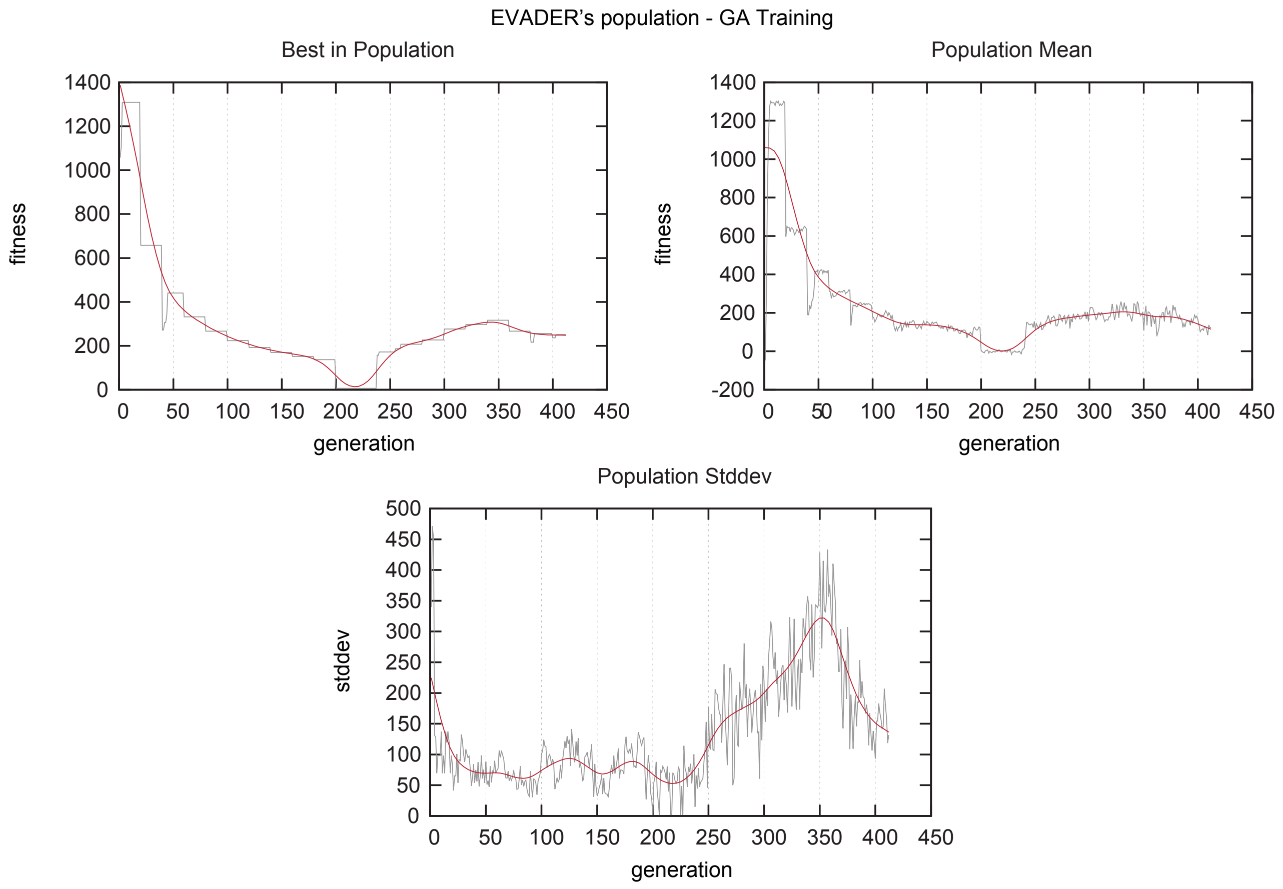

Fig. 5. The plots show the best fitness, average population fitness, and fitness standard deviation over generations of evader's population training.

The solution space is vast, and it has may local extrema. Therefore, it is difficult to clearly state if players could perform any better. Surely, training results may vary depending on the initial parameter setup. However, an important lesson is the training can be adjusted until a satisfactory outcome is achieved.

## XI. Summary

Through the training process, the players acquire knowledge that is encoded in the utility model. The knowledge can be used easily, requiring only reduced calculation. However, the information stored in a neural network cannot simply be exported to a human-readable form, unless it is a rule-based model. The study showed that the method can be successfully

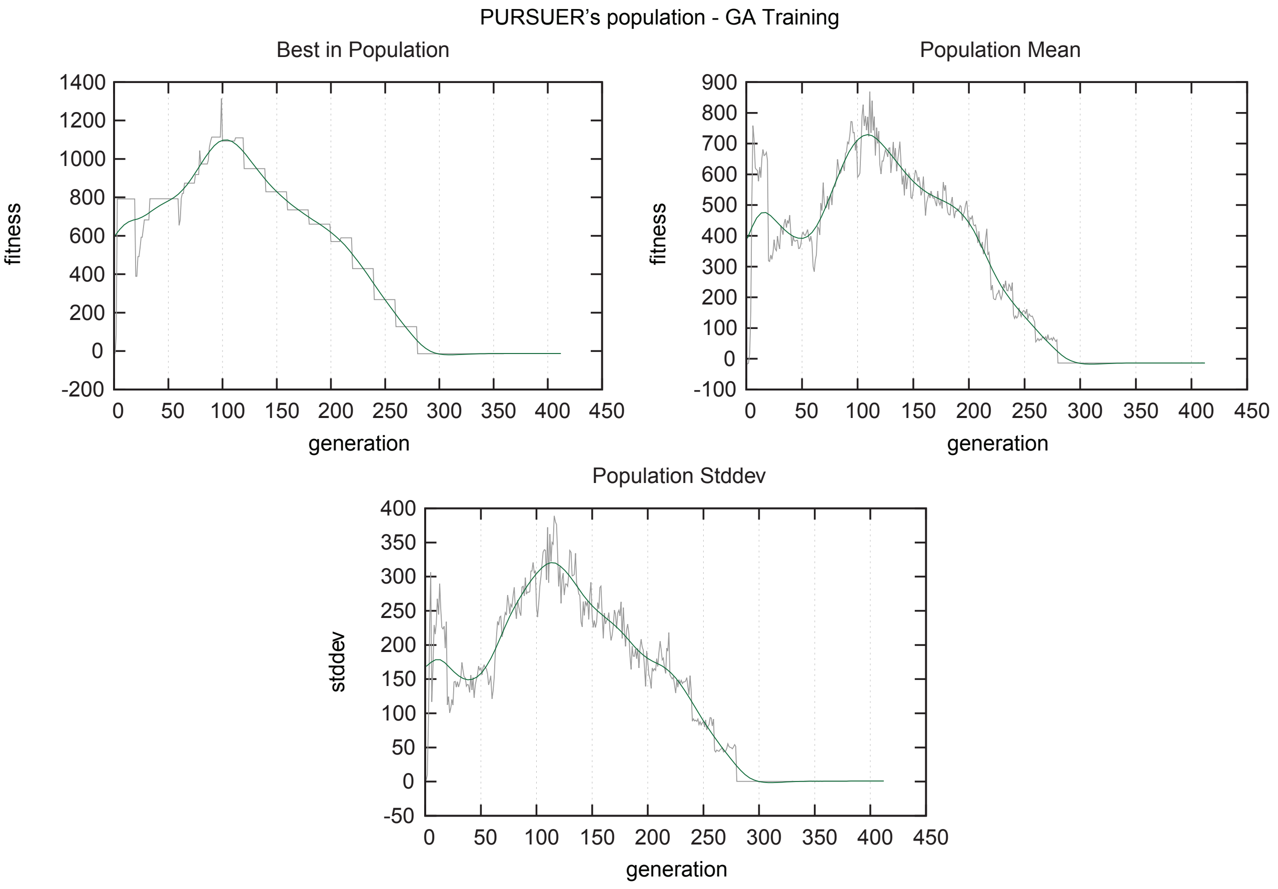

Fig. 6. The plots show the best fitness, average population fitness, and fitness standard deviation over generations of pursuer's population training.

used for observing how the system behaves depending on the game configuration.

Earlier studies have shown that the deeper Minimax algorithm penetrated a game tree, a more intelligent behavior of a player was observed, which is consistent with the theory. Also, it should be noted that the number of possible states of an unrevealed opponent taken into account by the algorithm has a congruent impact.

Another practical observation from the experiment is that the training phase is computationally expensive. Regardless of the fact that the implementation was using C++11 standard and calculations were efficiently distributed over several threads [23], the study was strongly impeded by time-consuming experiments. Accompanying tests proved that the process cannot be easily accelerated by GPU computing [24], [25]. The bottleneck is the Minimax algorithm that involves multiple calculations of the neural network's output. Despite the fact that computations in each network layer can be parallelized by GPU, the gain compared to CPU calculations was hard to observe.

An important achievement in the experimental study is that the players' strategies stabilize at some point. Strategies of competing players often oscillate when a pure-strategy equilibrium cannot be found. Although the stabilization is only one of the conditions that have to be satisfied to obtain a good solution, this should be considered as a significant success.

## XII. Future Work

From a theoretical point of view, adopting a pure strategy may not be the most suitable model for this game. Nonetheless, it is one of the simplest approaches, and proved to be sufficient. One of the interesting directions is to check a mixed strategy and stochastic decision rules. Subsequently, player positions and environment parameters should be randomized. Undoubtedly, it will substantially increase the training process since additional evaluation repetitions are required to obtain an acceptable statistical significance level. However, the new model addresses environments with uncertain information, and it should give better results.

Because of the overall problem difficulty, the initial study has employed very basic tools that are commonly used in the field of computational intelligence. In the next step, it would be beneficial to use Minimax hybrids to increase the performance of the tree search [26]. Deep Learning methods could be applied to improve the training process and the generalization capabilities of a neural network [27].

At the current stage of development, the system cannot be used for building a universal model of AUV strategy that could

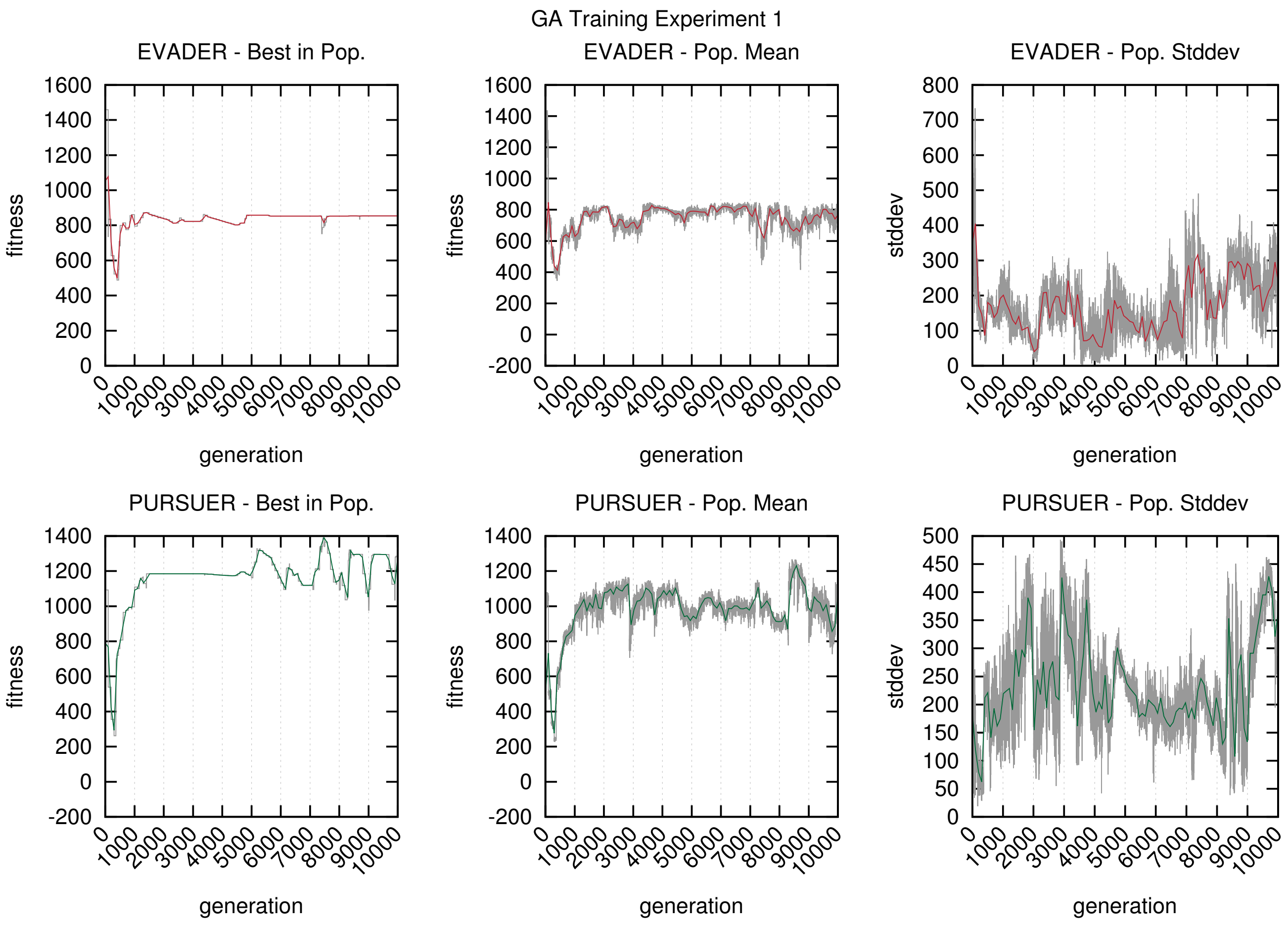

Fig. 7. A long training run 1.

be tested on actual hardware. Preparation of a reliable strategy requires a well-organized training process. The training should therefore introduce more realistic environmental parameters and cover various game scenarios including different player and mission arrangements. It should also simulate noisy and misleading information. Therefore, one of the primary goals is employing an accurate model of underwater signal propagation. It should be emphasized that the quality of the system must be evaluated according to the military knowledge and real-life scenarios.

## References

[1] P. Kachroo, *Autonomous Underwater Vehicles: Modeling, Control Design and Simulation*. CRC Press, 2010. ISBN 978-1-4398-1831-2

[2] G. Griffiths, Ed., *Technology and Applications of Autonomous Underwater Vehicles*. CRC Press, 2002. ISBN 978-0-415-30154-1

[3] Wikipedia. Blackghost photo. [Online]. Available: http://en.wikipedia.org/wiki/Autonomous_underwater_vehicle

[4] T. Basar and G. Olsder, *Dynamic Noncooperative Game Theory*, 2nd ed. Society for Industrial and Applied Mathematics, 1999. ISBN 978-0898714296

[5] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Prentice Hall PTR, 1998. ISBN 0-13-273350-1

[6] D. E. Goldberg, *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 2002. ISBN 1402070985

[7] D. Wagner, W. Mylander, and T. Sanders, *Naval Operations Analysis*. Naval Institute Press, 1999. ISBN 1-557-50956-5

[8] C. Harrison, "Closed form bistatic reverberation and target echoes with variable bathymetry and sound speed," *Oceanic Engineering, IEEE Journal of*, vol. 30, no. 4, pp. 660–675, Oct 2005. doi: 10.1109/JOE.2005.862095

[9] A. Sehgal and D. Cernea, "A multi-auv missions simulation framework for the usarsim robotics simulator," in *Control Automation (MED), 2010 18th Mediterranean Conference on*, June 2010. doi: 10.1109/MED.2010.5547632 pp. 1188–1193.

[10] A. Washburn and R. Hohzaki, "The diesel submarine flaming datum problem," *Military Operations Research*, vol. 6, no. 4, pp. 19–30, September 2001. doi: 10.5711/morj.6.4.19

[11] C. Strode, "Optimising multistatic sensor locations using path planning and game theory," in *Computational Intelligence for Security and Defense Applications (CISDA), 2011 IEEE Symposium on*, April 2011. doi: 10.1109/CISDA.2011.5945938 pp. 9–16.

[12] C. Strode, B. Mourre, and M. Rixen, "Decision support using the Multistatic Tactical Planning Aid (MSTPA)," *Ocean Dynamics*, vol. 62, no. 1, pp. 161–175, 2012. doi: 10.1007/s10236-011-0483-7

[13] J. Borges de Sousa, K. H. Johansson, A. Speranzon, and J. Silva, "A control architecture for multiple submarines in coordinated search missions," in *Proceedings of the 16th IFAC world congress*. IFAC, 2005. doi: 10.1.1.65.3030

[14] A. Antoniades, H. Kim, and S. Sastry, "Pursuit-evasion strategies for

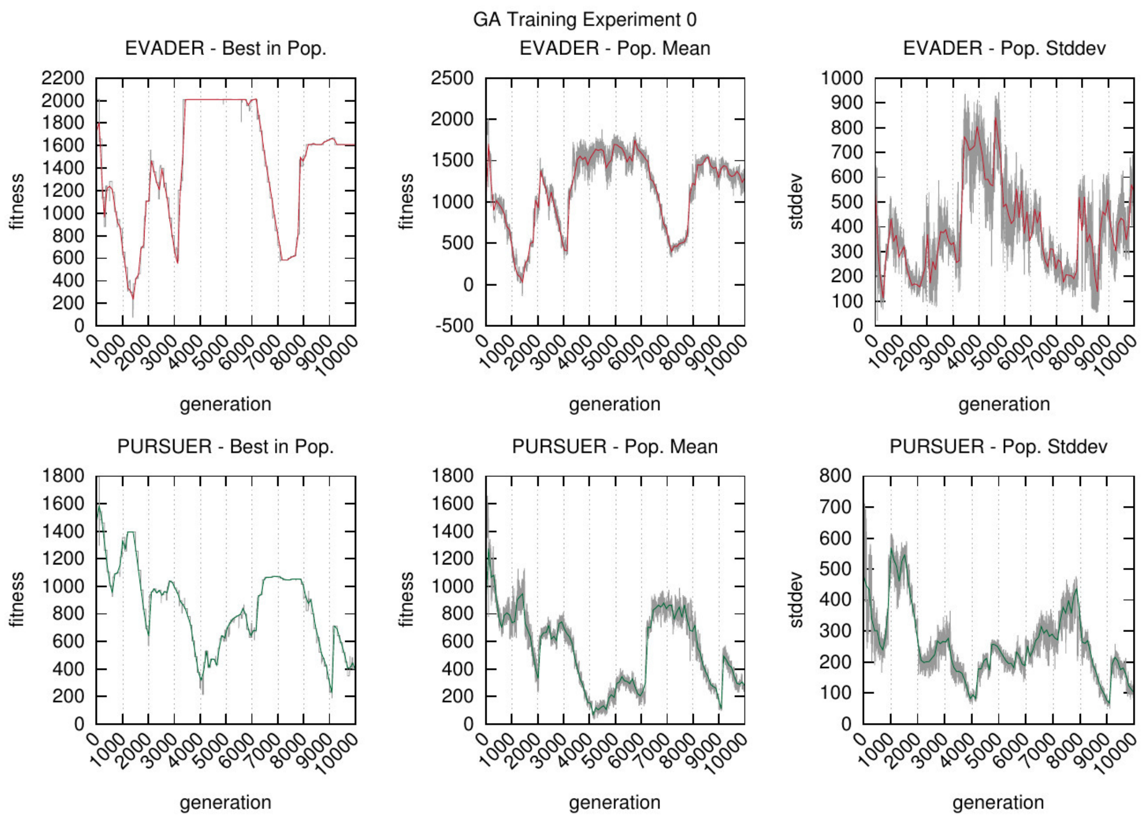

Fig. 8. A long training run 2.

teams of multiple agents with incomplete information," in *Decision and Control,* 2003. Proceedings. 42nd IEEE Conference on, vol. 1, Dec 2003. doi: 10.1109/CDC.2003.1272656. ISSN 0191-2216 pp. 756–761.

[15] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," *Auton. Robots,* vol. 31, no. 4, pp. 299–316, Nov. 2011. doi: 10.1007/s10514-011-9241-4

[16] R. Vidal, O. Shakernia, H. Kim, D. Shim, and S. Sastry, "Probabilistic pursuit-evasion games: theory, implementation, and experimental evaluation," *Robotics and Automation, IEEE Transactions on,* vol. 18, no. 5, pp. 662–669, Oct 2002. doi: 10.1109/TRA.2002.804040

[17] K.-B. Sim, D.-W. Lee, and J.-Y. Kim, "Game theory based coevolutionary algorithm: A new computational coevolutionary approach," *International Journal of Control, Automation, and Systems,* vol. 2, pp. 463–474, 2004. doi: 10.1.1.132.3826

[18] H. Sally and M. Rafie, "A survey of game theory using evolutionary algorithms," in *Information Technology (ITSim),* 2010 International Symposium in, vol. 3, June 2010. doi: 10.1109/ITSIM.2010.5561648. ISSN 2155-897 pp. 1319–1325.

[19] S. Kemna, M. J. Hamilton, D. T. Hughes, and K. LePage, "Adaptive autonomous underwater vehicles for littoral surveillance: the GLINT10 field trial results," *Intelligent Service Robotics,* vol. 4, no. 4, pp. 245–258, 2011. doi: 10.1007/s11370-011-0097-4

[20] R. Isaacs, *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization.* Courier Dover Publications, 1999. ISBN 0-486-40682-2

[21] J. Pearl, "The solution for the branching factor of the Alpha-beta pruning algorithm and its optimality," *Commun. ACM,* vol. 25, no. 8, pp. 559–564, Aug. 1982. doi: 10.1145/358589.358616. [Online]. Available: http://doi.acm.org/10.1145/358589.358616

[22] K. Chellapilla and D. Fogel, "Evolution, neural networks, games, and intelligence," *Proceedings of the IEEE,* vol. 87, no. 9, pp. 1471–1496, Sep 1999. doi: 10.1109/5.784222

[23] E. Alba and J. M. Troya, "A survey of parallel distributed genetic algorithms," *Complexity,* vol. 4, no. 4, pp. 31–52, 1999. doi:10.1002/(SICI)1099-0526(199903/04)4:4<31::AID-CPLX5>3.0.CO;2-4

[24] J. Sanders. (2010) *Introduction to CUDA C. GPU Technology Conference,* NVIDIA. [Online]. Available: http://www.nvidia.com/content/GTC-2010/pdfs/2131_GTC2010.pdf

[25] S. Rennich. (2011) *CUDA C/C++ streams and concurrency.* NVIDIA. [Online]. Available: http://on-demand.gputechconf.com/gtc-express/2011/presentations/StreamsAndConcurrencyWebinar.pdf

[26] H. Baier and M. H. M. Winands, "Monte-carlo tree search and minimax hybrids," in *Computational Intelligence in Games (CIG)*, 2013 IEEE Conference on, Aug 2013. doi: 10.1109/CIG.2013.6633630. ISSN 2325-4270 pp. 1–8.

[27] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks,* vol. 61, pp. 85–117, 2015. doi: 10.1016/j.neunet.2014.09.003