2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Search-Evasion Path Planning for Submarines Using the Artificial Bee Colony Algorithm

Bai Li
School of Control Science and Engineering
Zhejiang University
Hangzhou, China
libai@zju.edu.cn

Raymond Chiong
School of Design, Communication and Information Technology
The University of Newcastle
Callaghan, Australia
Raymond.Chiong@newcastle.edu.au

Li-gang Gong
School of Automation Science and Electrical Engineering
Beihang University
Beijing, China
glgbh@aspe.buaa.edu.cn

***Abstract*—Submarine search-evasion path planning aims to acquire an evading route for a submarine so as to avoid the detection of hostile anti-submarine searchers such as helicopters, aircraft and surface ships. In this paper, we propose a numerical optimization model of search-evasion path planning for invading submarines. We use the Artificial Bee Colony (ABC) algorithm, which has been confirmed to be competitive compared to many other nature-inspired algorithms, to solve this numerical optimization problem. In this work, several search-evasion cases in the two-dimensional plane have been carefully studied, in which the anti-submarine vehicles are equipped with sensors with circular footprints that allow them to detect invading submarines within certain radii. An invading submarine is assumed to be able to acquire the real-time locations of all the anti-submarine searchers in the combat field. Our simulation results show the efficacy of our proposed dynamic route optimization model for the submarine search-evasion path planning mission.**

Keywords—artificial bee colony; numerical optimization; search-evasion path planning; submarines

I. INTRODUCTION

A submarine is a watercraft capable of independent operation underwater [1]. Submarines first appeared in large scale during World War I for military usages [2], but now they are widely used for many civilian purposes too. Military missions of submarines include attacking the hostile submarines or surface ships, aircraft carrier protection, blockade running and underwater reconnaissance, among others [3]. Due to their outstanding capability to lurk in the deep sea and to launch a sudden attack, submarines have been taken as a vital part of a nation's military strength in the modern world.

At the same time, anti-submarine technologies are well developed to prevent the invasion of hostile submarines. In order to prevent submarine invasion, the first step is to detect the invading submarines successfully. To that end, three kinds of methods are commonly used. The first is to lay passive sonar in the sea [4]. However, such sensors will inevitably drift with ocean streams. Moreover, the sea clutter (i.e., background noise in the sea) places severe limits on detecting and tracking the underlying submarines [5]. Also, this method is not practical when the search scope becomes large. The second method is to arrange anti-submarine aircraft or surface ships bounded with sensors (e.g., radars, magnetic detectors or infrared detectors) for the purpose of patrolling [6]. It is currently a prevailing method due to the flexibility and maneuverability of anti-submarine vehicles. The third way is to dispatch autonomous underwater vehicles (AUVs) to patrol under the sea [7], aiming to reduce the interference of sea clutter. This may be a well-adopted approach in the future when AUVs are becoming sufficiently intelligent. Many research studies have been carried out on efficient search strategies in this area (e.g., see [8-12]). Interested readers can refer to [13] for a thorough review.

In contrast to the work done on the detection and search of invading submarines, limited research has been devoted to search-evasion path planning for the invading submarines. This paper focuses on the route planning scheme for invading submarines, which aims to avoid being detected by the anti-submarine sensors. We assume that the searchers are equipped with sensors with circular footprints that allow them to detect invading submarines on the condition that the invading submarines fall within the sensor's radii [14]. In addition, we assume that an invading submarine is able to acquire the real-time locations of all anti-submarine searchers in the combat field with the help of cooperative reconnaissance satellites or intelligence aircraft. During the manoeuvre from the initial location to a given destination, the invading submarine can update its route dynamically according to the latest intelligent information received regarding the anti-submarine searchers. In this work, we attempt to transform the original search-evasion route programming scheme into a numerical optimization problem.

Finding the optimum for this kind of transformed numerical optimization problem has been confirmed to be NP-hard [15]. Consequently, nature-inspired stochastic search and optimization methods such as genetic algorithms, genetic programming, differential evolution, ant colony optimization and particle swarm optimization have to be used [16-17]. Although these stochastic approaches do not guarantee global convergence, they are able to find quasi-optimal solutions within a reasonable amount of computational time [18]. As a matter of fact, stochastic search and optimization methods have been well studied and investigated in various route planning schemes [19-28].

Artificial Bee Colony (ABC) is a relatively new swarm intelligence algorithm inspired by the foraging behavior of honey bees [29]. Various studies using different numerical benchmark tests have confirmed that the ABC algorithm possesses competitive advantages compared to some other swarm intelligence and evolutionary algorithms [30-33] (in regard to the question of what is an "evolutionary algorithm", see [34] for some interesting discussion). In addition, the algorithm framework of ABC is relatively simple, thus making it possible to acquire good results at a low computational cost [35]. This brings about a range of modifications/improvements made for the conventional ABC algorithm in recent years. From the authors' viewpoint, broadly speaking there are three prevailing ways to improve the conventional ABC. The first is to adopt some strategies or theories from the "outside world". The second is mainly about modifying the solution search equations. While the third focuses on changing the framework of the ABC algorithm. An incomplete collection of references [28, 35-48] is used in Figure 1 to provide a quick overview of possible modifications made to the conventional ABC algorithm.

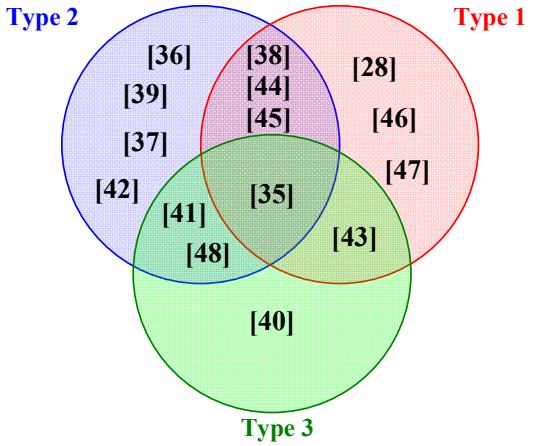

Fig. 1. An incomplete review of modifications made for the ABC algorithm.

In this work, we adopt the ABC algorithm to solve the transformed numerical optimization problem of search-evasion path planning for invading submarines. The remainder of this paper is organized as follows. In the next section (i.e., Section II), we briefly review a static route planning model that is commonly used in many path programming schemes. Then, we introduce our proposed dynamic search-evasion path planning model, which is inspired by the conventional model introduced in Section II, in Section III. The procedures of the ABC algorithm are described in Section IV, followed by Section V where experimental setups and results have been presented to show the efficacy of our proposed dynamic model. Finally, the strengths and limitations of this work are highlighted in the last section.

II. A Brief Review of the Conventional Path Planning Model with Static Threats

In this section, we briefly introduce the conventional route planning model where only static anti-submarine settings (e.g., anti-submarine ships berthed in the sea) will be considered. The detection scope of these static vehicles is presented by circles of different radii. Besides that, a threat weight index is associated with each of the static anti-submarine vehicles thanks to the reliability of the equipped sensor. When an invading submarine's path falls within the scope of such a static detection sensor, the invasion will be flagged.

Let us denote the starting point as S and the terminal point as T. The search-evasion mission is to calculate an optimal path from S to T, with all the fixed threat regions as well as fuel consumption considered. First, we draw a segment ST connecting S and T. After that, we divide ST into $(D+1)$ equal portions by D vertical dash lines L_k $(k=1,2,..,D)$ as illustrated in Figure 2. These lines are taken as the new axes. Then, as many D points (see the small rectangles in Figure 2) along these axes will be connected in sequence to form a feasible path from S to T [28]. In this way, the vector $\mathbf{z}=[z_1,z_2,\ldots,z_D]$ determines the shape of a feasible path from S to T.

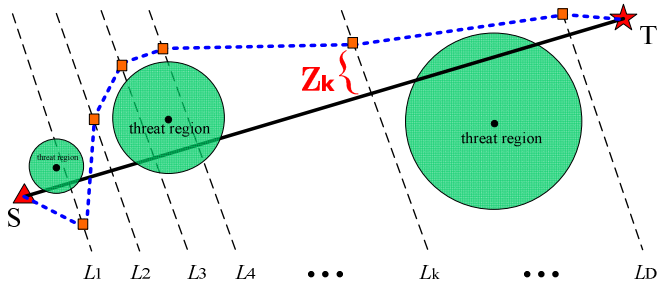

Fig. 2. A schematic diagram of the conventional path planning model with fixed threats only.

Regarding the quality of a candidate search-evasion path, the threat cost J_{threat} as well as the fuel consumption J_{fuel} are taken into consideration as shown in (1) below:

$$\begin{aligned} J &= \lambda \cdot J_{fuel} + (1-\lambda)\cdot J_{threat} \\ &= \lambda \cdot \int_0^{length} w_{threat}\mathrm{d}l + (1-\lambda)\cdot \int_0^{length} w_{fuel}\mathrm{d}l, \end{aligned} \tag{1}$$

where J is the weighted sum cost of this path, $\lambda \in [0,1]$ is a weighting parameter, w_{threat} and w_{fuel} are variables related to every instantaneous position on the path, and *length* denotes the total length of this candidate path.

If a search-evasion path falls into a threat region during the sub-route from point PL_i on L_i to point PL_{i+1} on L_{i+1}, some form of detection penalty will be calculated. Equation (2) shows a general function of detection probability, assumed to be supplied through the radars on the anti-submarine searchers [5], where *Prob* denotes the detection probability, d refers to the distance between the detecting radar and the invading submarine, C is a performance parameter of the radar, and σ_s denotes the radar cross section of the target. Since C and σ_s are typically taken as constant parameters [13], we have

$Prob \propto 1/d^4$, which also holds when the surface anti-submarine ships are equipped with other kinds of sensors for detection in general.

$$Prob(d) = \frac{C \cdot \sigma_s}{d^4} \tag{2}$$

To simplify the integral operation in (1), we calculate $\left(\sum_{i=1}^{D-1} w_{threat,L_i \to L_{i+1}}\right) + w_{threat,S \to L_1} + w_{threat,L_1 \to D}$ instead of $\int_0^{length} w_{threat} dl$ in practice. The detection cost $w_{threat,L_i \to L_{i+1}}$ from PL_i to PL_{i+1} is calculated at five sample points in-between, as illustrated in Figure 3, using (3):

$$w_{threat,L_i \to L_{i+1}} = \frac{length_i}{5} \cdot \sum_{k=1}^{N_t} \left[t_k \cdot \left(\frac{1}{d_{0.1,i,k}^4} + \frac{1}{d_{0.3,i,k}^4} + \frac{1}{d_{0.5,i,k}^4} + \frac{1}{d_{0.7,i,k}^4} + \frac{1}{d_{0.9,i,k}^4} \right) \right], \tag{3}$$

where N_t denotes the number of threatening circles the sub-route falls into, $length_i$ refers to the length of the ith sub-path, $d_{0.1,i,k}$ stands for the distance between the $1/10$ point on the path and the kth threat center, and t_k is regarded as the threat grade of the kth threat. It is a common practice to assume that the fuel consumption J_{fuel} is proportional to $length$, which means w_{fuel} will be a constant [28]. There is no problem if we set $w_{fuel} \equiv 1$ because the polynomial $(1-\lambda)$ in (1) is always attached with w_{fuel}.

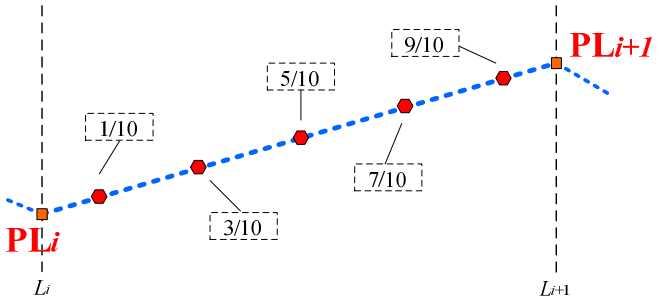

Fig. 3. A schematic diagram of route cost calculation in a simplified way to avoid the integral operation.

As a brief summary, adjusting the elements in the vector $\mathbf{z} = [z_1, z_2, \ldots, z_D]$ will lead to different paths from S to T. Each feasible path corresponds with a path cost value calculated using (1). We seek for the very $\mathbf{z}^*$ that makes $J(\mathbf{z}^*)$ minimized. In this way, it is notable that the original search-evasion path planning scheme is transformed into a numerical optimization problem. The optimal vector $\mathbf{z}^*$ can be achieved through the ABC algorithm, which will be presented in Section IV.

III. The Novel Dynamic Search-evasion Path Planning Model

We have reviewed the route planning model with only fixed anti-submarine threats considered in the previous section. In this section, we will take mobile anti-submarine searchers into consideration. Our assumption is that the invading submarine can get the real-time locations of all the anti-submarine vehicles in the combat field by means of cooperative reconnaissance satellites or intelligence aircraft. Such assumption is considered to be feasible and practical [49, 50]. In this work, we propose a locally look-ahead strategy for the real-time route planning scheme.

Assuming that the invading submarine is currently located at point PL_i that is on the line L_i. Different from the static model in Section II, here we will program as many N_{ahead} points PL_{i+1}, …, $\mathrm{PL}_{i+N_{ahead}}$, which are on the coordinate axes L_{i+1}, …, $L_{i+N_{ahead}}$ respectively. Thereafter, we will calculate the cost of the sub-route $\overset{\frown}{\mathrm{PL}_i \ \ldots \ \mathrm{PL}_{i+N_{ahead}}}$ rather than that of the whole route. An example is shown in Figure 4. We assume that the invading submarine is now located at PL_2, and that $N_{ahead} = 2$. At this moment, the submarine concerns only of the sub-route $\overset{\frown}{\mathrm{PL}_2\ \mathrm{PL}_3\ \mathrm{PL}_4}$, making it an N_{ahead}-dimensional numerical optimization sub-problem. This sub-problem is solved by optimizing only the elements $z_{i+1}, \ldots, z_{i+N_{ahead}}$ in the vector $\mathbf{z}$. Now that the submarine is located at PL_i, the elements $z_1, \ldots, z_i$ will always remain as historical data. Similarly, the elements $z_{i+N_{ahead}+1}, \ldots, z_D$ will be temporarily fixed because we do not worry about them at this moment.

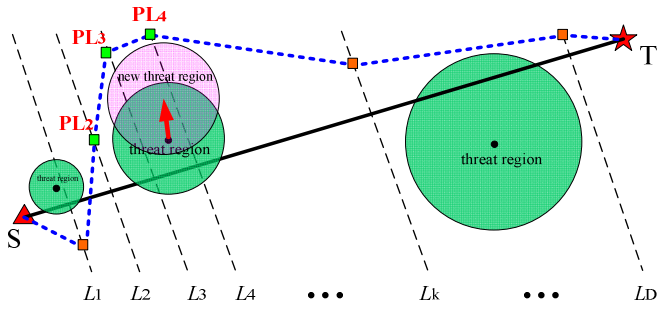

Fig. 4. A schematic diagram of the dynamic path planning model with fixed and mobile threats considered. According to our proposed locally look-ahead strategy, the invading submarine concerns only about a sub-route from PL_2 to PL_4.

A question that would arise is, why is it making sense to look merely several steps locally ahead? Since there may be some drastic changes in the combat field as time goes by, the global route computed in an earlier step can be of no use in a subsequent step when the situation differs. On the contrary, if we only solve a sub-problem at each step the optimization dimension will be reduced significantly (because $N_{ahead} < D$

in general). This eases the real-time computation burden as well.

IV. The Artificial Bee Colony Algorithm

In the preceding section, a description of the route planning model with both fixed and mobile anti-submarine threats considered has been presented. This section focuses on how to find an optimal vector $\mathbf{z}^*$ with the minimum path cost function value using the ABC algorithm.

The conventional ABC algorithm employs three kinds of "bees": scout bees searching for nectar sources randomly, employed bees associated with specific nectar sources, and onlooker bees who keep watch on the employed bees. Half of a bee colony would consist of the employed bees, and the other half the onlooker bees [29]. At an initial stage, scout bees are set out to randomly search for nectar sources. Soon after, they become the employed bees responsible for sharing information (e.g., nectar source quality and their current locations) with other employed bees as well as the onlooker bees by means of "dancing". The onlooker bees will then randomly select the locations of the employed bees to exploit. It is worth pointing out that the locations with relatively higher quality nectar sources are more likely to be chosen by the onlooker bees for exploitation. The employed bees, on the other hand, will also randomly share their locations with others to explore possible new locations. If an employed bee finds no better nectar source than one that it has previously discovered within a certain time length, it turns into a scout bee again. Its position will be a randomly initialized location in the search space.

A location of a nectar source represents a feasible solution to the problem, and the nectar quantity is reflected by the objective function value [30]. Let $\mathbf{X} = (X^1, X^2, \cdots, X^D)$ represent a solution in the feasible solution space, $fun(\cdot)$ be the objective function that needs to be minimized, $rand(m,n)$ be a random number between m and n obeying the uniform distribution, and SN be the population size of a bee swarm. As aforementioned, the number of onlooker bees in a bee colony is $SN/2$, equaling that of the employed bees.

At first, as many as $SN/2$ scout bees are randomly initialized in the feasible solution space. Equation (5) shows how the jth element of the ith scout bee's location $\mathbf{X}_i$ is calculated:

$$X_i^j \leftarrow X_{\min}^j + rand(0,1)\cdot(X_{\max}^j - X_{\min}^j), \quad i = 1,2,...,SN/2,\ j = 1,2,...,D, \tag{5}$$

where $X_{\min}^j$ and $X_{\max}^j$ denote the lower and upper boundaries of this jth element, and D denotes the dimension of a feasible solution. Thereafter, the $SN/2$ scout bees will become the employed bees and an iterated process begins from here.

In each cycle of iteration, an employed bee will share information with a randomly chosen companion and change one randomly chosen element of its location vector from X_i^j to X^{*j}_i using the following equation:

$$X^{*j}_i \leftarrow X_i^j + rand(-1,1)\cdot(X_k^j - X_i^j), \quad k \in \{1,2,\ldots,SN/2\},\ j \in \{1,2,\ldots,D\},\ k \neq i. \tag{6}$$

It is necessary to note that j and k are both randomly selected integers. When all the employed bees arrive at their new nectar sources $\{\mathbf{X}_i^*,\ i = 1,2,\ldots,SN/2\}$, they evaluate the quality of these new nectars and then decide whether to stay at the new location or the previous one by means of a greedy selection strategy. Specifically, if the ith employed bee finds that $fun(\mathbf{X}_i^*) < fun(\mathbf{X}_i)$, it will go to the new location $\mathbf{X}_i^*$, i.e., $\mathbf{X}_i \leftarrow \mathbf{X}_i^*$; otherwise, it remains at the previous location $\mathbf{X}_i$.

When all the employed bees have decided on their locations, a roulette selection strategy will direct the onlooker bees to select "qualified" employed bees to follow. A probability index P is calculated according to (7) and (8) to reflect the relative qualification of nectar sources at which the employed bees are located.

$$P(i) = \frac{fitness(i)}{\sum_{j=1}^{SN/2} fitness(j)},\ i = 1,2,\ldots,SN/2, \tag{7}$$

$$fitness(i) = \begin{cases} \dfrac{1}{1+fun(\mathbf{X}_i)} & \text{if } fun(\mathbf{X}_i) \geq 0 \\ 1+\left|fun(\mathbf{X}_i)\right| & \text{if } fun(\mathbf{X}_i) < 0 \end{cases}. \tag{8}$$

Each onlooker bee will search locally around an employed bee. For some ith onlooker bee, a comparison is made between a random number $rand(0,1)$ and $P(1)$. If $P(1) \geq rand(0,1)$, this onlooker bee will search around the 1st employed bee; otherwise, a comparison between $rand(0,1)$ and $P(2)$ will be made. If any $P(j),\ j \in \{1,2,\ldots,SN/2\}$ happened to be smaller than $rand(0,1)$, such process is repeated until a larger $P(j)$ is found. Then, the corresponding jth employed bee will be chosen. The following equation (i.e., Equation (9)) shows the location of the ith onlooker bee $\mathbf{Y}_i = \left(X_j^1,\ldots,X_j^{k-1},Y_i^k,X_j^{k+1},\ldots,X_j^D\right)$ that searches locally around the selected j-th employed bee.

$$Y_i^k \leftarrow X_j^k + rand(-1,1)\cdot(X_m^k - X_j^k), \quad m \in \{1,2,\ldots,SN/2\},\ k \in \{1,2,\ldots,D\},\ m \neq j. \tag{9}$$

Note that in this equation m and k are randomly selected integers as well. When all the $SN/2$ onlooker bees have

determined their locations, a greedy selection strategy is implemented. This time, however, a comparison is made between $fun(\mathbf{X}_j)$ and $fun(\mathbf{Y}_i)$ $i = 1, 2, \ldots, SN/2$. If $fun(\mathbf{Y}_i)$ is smaller than $fun(\mathbf{X}_j)$, the jth employed bee will abandon the current location $\mathbf{X}_j$ and go to $\mathbf{Y}_i$, i.e., $\mathbf{X}_j \leftarrow \mathbf{Y}_i$; otherwise, the jth employed bee remains at $\mathbf{X}_j$.

It is interesting to note that every time the greedy selection is implemented, it involves one central employed bee. In the ABC algorithm, besides the probability index P there is another index that is associated with each of the employed bees, namely *trial*, which memorizes inefficient information that is relevant to the employed bees. Specifically, $trial(i)$ records the number of times an inefficient search is performed by the ith employed bee or any onlooker bee that searches around the ith employed bee. That is to say, $trial(i)$ is incremented by one each time when the condition $fun(\mathbf{X}_i^*) \geq fun(\mathbf{X}_i)$ or $fun(\mathbf{Y}_j) \geq fun(\mathbf{X}_i)$ is satisfied. At the beginning, each $trial(i)$ is set to zero. As the iteration process goes on, when $trial(i)$ reaches a predefined threshold *Limit* the ith employed bee will turn into a scout bee again with a randomly initialized location in the search space (based on (5)).

The pseudo-code of the ABC algorithm is given as follows:

Algorithm 1 The Artificial Bee Colony Algorithm

1. Set the population size SN, and maximum cycle number MCN. Set the inefficient trial time counter $trial(i) \leftarrow 0$ $(i = 1, 2, \ldots, SN/2)$.
2. Randomly initialize locations of $SN/2$ scout bees using Eq. (5)
3. **For** $iter = 1$ to MCN, **do**
4. **For** $item = 1$ to $SN/2$, **do** *% employed bee phase*
5. Generate $\mathbf{X}_{item}^*$ for the *item*-th employed bee to search according to Eq. (6)
6. **If** $fun(\mathbf{X}_{item}^*) < fun(\mathbf{X}_{item})$, **then** *% implementation of the greedy selection*
7. $\mathbf{X}_{item} \leftarrow \mathbf{X}_{item}^*$, and set $trial(item) \leftarrow 0$
8. **Else**
9. $trial(item) \leftarrow trial(item) + 1$
10. **End if**
11. **End for**
12. **For** $i = 1$ to $SN/2$, **do** *% preparation for the roulette selection*
13. Calculate $P(i)$ using Eq. (7) and Eq. (8)
14. **End for**
15. Set $j = 1$ *% implementation of the roulette selection*
16. **For** $item = 1$ to $SN/2$, **do**
17. **If** $P(j) > rand(0,1)$, **then** *% onlooker bee phase*
18. Choose the jth employed bee to follow, and then generate $\mathbf{Y}_{item}$ using Eq. (9)
19. **If** $fun(\mathbf{Y}_{item}) < fun(\mathbf{X}_j)$, **then** *% implementation of the greedy selection*
20. $\mathbf{X}_j \leftarrow \mathbf{Y}_{item}$, and set $trial(j) \leftarrow 0$
21. **Else**
22. $trial(j) \leftarrow trial(j) + 1$
23. **End if**
24. **End if**
25. $j \leftarrow j + 1$
26. **If** $j > SN/2$, **then**
27. Set $j \leftarrow 1$
28. **End if**
29. **End for**
30. Collect *item* that satisfy $trial(item) > Limit$ in an index set Ω *% scout bee phase*
31. **If** $\Omega \neq \varnothing$, **then**
32. Randomly choose $k \in \Omega$
33. Re-initialize the location of the kth employed bee using Eq. (5)
34. Set $trial(k) \leftarrow 0$
35. **End if**
36. Memorize the best-ever solution
37. **End for**
38. Output the best-ever solution

V. Experiments and Results

We have conducted a number of simulation experiments to verify the efficacy of our proposed dynamic search-evasion path planning model. All the experiments were done in a MATLAB R2010a environment and executed on an Intel Core 2 Duo CPU with 2 GB RAM running at 2.53 GHz under Windows XP. The population size was set constant at $SN = 40$ and $Limit = MCN/10$ for the ABC algorithm.

In the first experiment, we investigated the route programming ability of our proposed method to handle static anti-submarine threats (see Figure 5). The starting location was set to (11, 11), and the destination was pre-defined as (75, 75). Some other parameters were set as: $MCN = 30$, $D = 15$, $N_{ahead} = 2$ and $\lambda = 0.7$.

The second experiment focused on the existence of moving anti-submarine aircraft in the combat field (see Figure 6), while other settings remained the same as in the previous static case. In this experiment, an anti-submarine aircraft flew from (63, 56) to (58.50, 73.25).

Figures 5 and 6 confirm that our proposed approach is capable of handling both static and dynamic anti-submarine threats. In both cases, we repeated the simulations for as many as 30 times. From Figure 5, we see that most of the optimized

routes share similar trends, i.e., they are able to avoid the static threat regions. In Figure 6, when some of the anti-submarine vehicles can move, we observe that most of the programmed routes are still able to avoid the moving threat efficiently. Nevertheless, we acknowledge that there exist a few imperfect routes among the 30 runs in both cases, as can be seen in Figures 5 and 6. This means there is still room for improvement in the optimization process of this ABC-based dynamic path planning model.

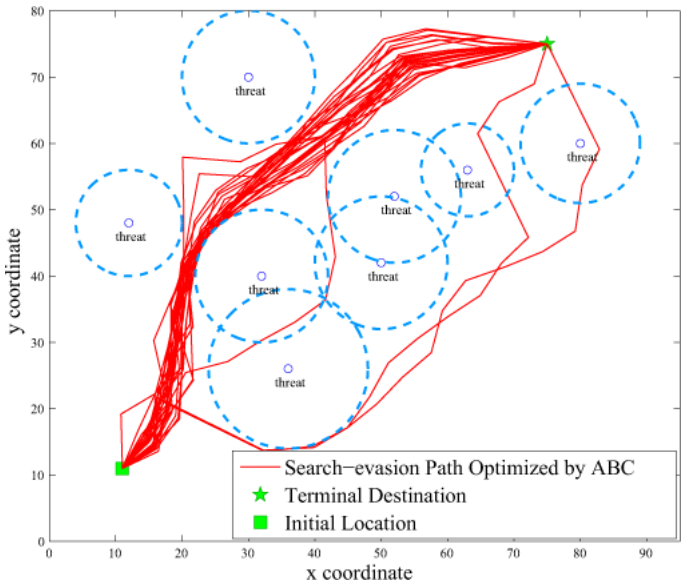

Fig. 5. Simulation results of search-evasion paths (30 runs) optimized by the ABC algorithm when only static anti-submarine threats have been considered.

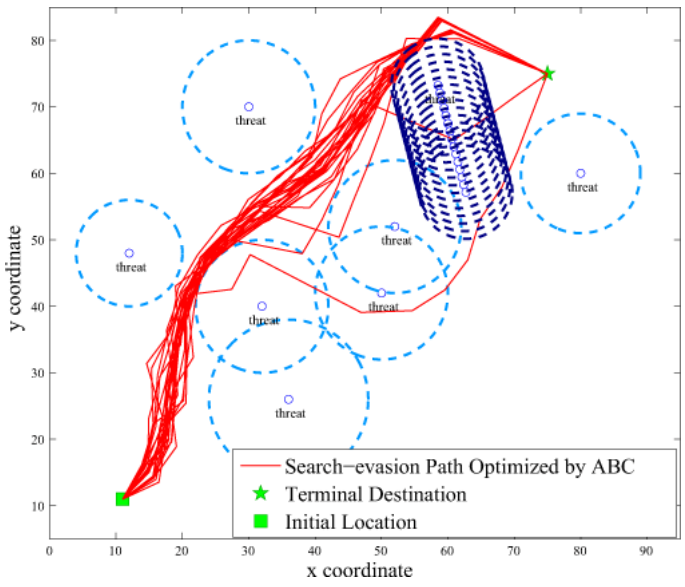

Fig. 6. Simulation results of search-evasion paths (30 runs) optimized by the ABC algorithm when both static and dynamic anti-submarine threats/searchers have been considered.

In addition to the above two experiments, we have also carried out a sensitivity analysis on the parameters of the proposed model. Due to space constraints, only the results for the parameters N_{ahead}, λ and D are presented.

First, we investigated how the selection of N_{ahead} would affect the search-evasion route in a complicated case (see Figure 7). In this case, one anti-submarine aircraft flew from (63, 56) to (58.50, 73.25), another anti-submarine aircraft flew from (12, 48) to (10.5, 28.5), while the third anti-submarine aircraft flew from (30, 70) to (28.95, 65.20) when the invading submarine attempted to move from the starting point to the terminal destination. Figure 7 shows a number of optimized routes obtained when N_{ahead} = 1, 2, 3 and 5 respectively, with $\lambda \equiv 0.7$ and $D \equiv 15$.

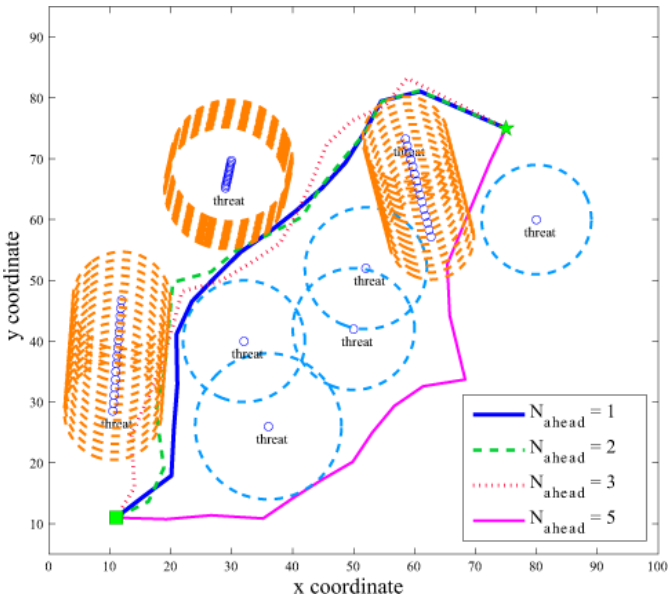

Fig. 7. Simulation results concerning the sensitivity analysis of N_{ahead}.

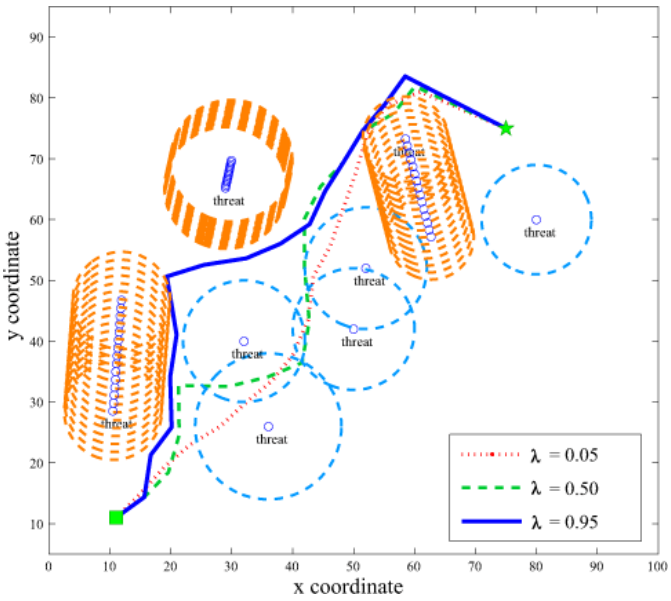

Fig. 8. Simulation results concerning the sensitivity analysis of λ.

As can be seen from Figure 7, when the value of the parameter N_{ahead} is relatively small, the three routes obtained for N_{ahead} = 1, 2 and 3 are very close to each other. When $N_{ahead} = 5$, however, the route goes from S to T in a completely different direction to the right. These results indicate that our proposed model is robust with different selection of N_{ahead}, although one may ask why the route with a larger N_{ahead} value turns out to be so different? Here, it is interesting to point out that when the invading submarine is set to "look" 5 steps ahead, it has the tendency to veer right in an attempt to keep itself away from the detectors. However, since the situation of the combat field changes all the time, to look far ahead makes no sense. Therefore, we suggest that the selected N_{ahead} value should never be too large.

Second, we investigated how the selection of λ would affect the search-evasion route. Figure 8 shows the routes when λ = 0.05, 0.5 and 0.95 respectively, with $N_{ahead} \equiv 2$ and $D \equiv 15$. As we can see from the figure, the larger the value of λ, the better the optimized route is in avoiding the threat zones. When the value of λ is smaller, it concerns more about the length of the planned route from S to T.

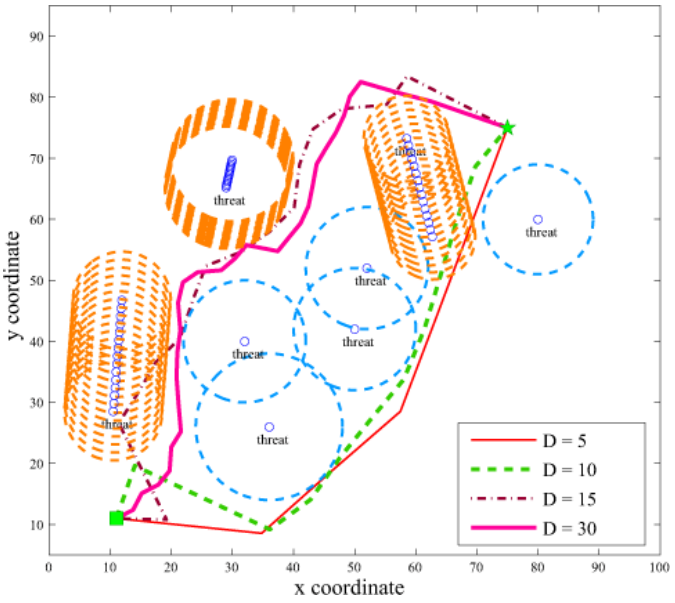

Fig. 9. Simulation results concerning the sensitivity analysis of D.

Finally, we also investigated how the selection of D would affect the search-evasion route. Figure 9 shows the routes when D = 5, 10 and 30 respectively, with $N_{ahead} \equiv 3$ and $\lambda \equiv 0.95$. In general, an optimized route turns out to be more exquisite with a larger D value. Again, one may find that the route goes to the destination from the right side when D is small in Figure 9. It is necessary to point out here that the submarine will go as many D steps as possible from S to T. When D is small, each step will generally be bigger. Besides that, it also looks N_{ahead} steps ahead. Therefore, when D is smaller, the submarine looks farther in advance, making it a preferable choice to veer to the right side. In this regard, a relatively small D value has somewhat a similar effect to a large N_{ahead} value as discussed previously. Some readers may point out that the invading submarine can only plan a subsequent route when it reaches one coordinate axis L_i. This is true, however, when D is set to be sufficiently large we believe that this will not be a critical problem.

VI. Conclusion and Future Work

In this paper, we have applied the ABC algorithm for optimizing the route of an invading submarine based on a proposed dynamic path planning model. Our simulation results clearly demonstrated that the proposed method is a viable approach for submarine search-evasion path planning.

Despite the positive results, this route planning model still has its limitations. For example, the model is formulated in a two-dimensional plane, which basically assumes that all the underwater vehicles lay at a same depth below the sea level. This is obviously not realistic when a real-world application of this kind is pursued.

For future work, we may consider the formulation of a model in a three-dimension space. Besides that, given that our optimized paths are composed of line segments we may also investigate ways to smooth the final paths in order to remove discontinuities in the velocity.

Acknowledgment

We would like to thank the anonymous referees for their valuable comments and suggestions. The second author would like to acknowledge support from the University of Newcastle's PVC Conference Assistance Grant Scheme. This work is also supported in part by the 5th National College Students' Innovative and Entrepreneurial Training Program in China.

References

[1] R. A. Geyer, Submersibles and their use in oceanography and ocean engineering. Elsevier, 2011.

[2] G. Hardach, The First World War, 1914-1918. University of California Press, 1981.

[3] G. M. Sandal, I. M. Endresen, R. Vaernes, and H. Ursin, "Personality and coping strategies during submarine missions," Military Psychology, vol. 11, no. 4, pp. 381, 1999.

[4] P. C. Etter, Underwater acoustic modeling and simulation. CRC Press, 2013.

[5] M. I. Skolnik, Radar handbook. New York: Academic, 1970.

[6] Y. Qiu, W. Zhang, P. Zhao, and X. Liu, "Sea wave filter design for cable-height control system of anti-submarine helicopter," Emerging Technologies for Information Systems, Computing, and Management, vol. 236, pp. 433–441, 2013.

[7] J. Das, F. Py, T. Maughan, T. O'Reilly, M. Messie, J. Ryan, and G. S. Sukhatme, "Simultaneous tracking and sampling of dynamic oceanographic features with autonomous underwater vehicles and lagrangian drifters," Experimental Robotics, vol. 79, pp. 541–555, 2014.

[8] J. M. Danskin, "A helicopter versus submarine search game," Operations Research, vol. 16, no. 3, pp. 509–517, 1968.

[9] M. F. Shlesinger, "Mathematical physics: Search research," Nature, vol. 443, pp. 281–282, 2006.

[10] H. Jin and J. Li, "Submarine searching strategies for dipping sonar on antisubmarine helicopter," Electronics Optics & Control, vol. 18, no. 8,

pp. 26–29, 2011 (in Chinese).

[11] T. H. Chung, G. A. Hollinger, and V. Isler, "Search and pursuit-evasion in mobile robotics," Autonomous Robots, vol. 31, no. 4, pp. 299–316, 2011.

[12] C. Pan, J. Hu, and Z. Yin, "Quasi-optimal method for multiple UUVs cooperate to search static target," Fire Control & Command Control, vol. 38, no. 4, pp. 53–56, 2013 (in Chinese).

[13] Y. Que, Effectiveness evaluation and decision modeling on searching submarine of antisubmarine aircraft. National Defense Industry Press, 2011 (in Chinese).

[14] T. G. McGee and J. K. Hedrick, "Guaranteed strategies to search for mobile evaders in the plane," Proceedings of the 2006 American Control Conference, IEEE, pp. 2819–2824, 2006.

[15] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," Proceedings of the 28th Annual Symposium on Foundations of Computer Science, pp. 49-60, IEEE, 1987.

[16] R. Chiong, Ed. Nature-inspired algorithms for optimisation. Berlin: Springer-Verlag, 2009.

[17] R. Chiong, T. Weise, and Z. Michalewicz, Eds. Variants of evolutionary algorithms for real-world applications. Berlin: Springer-Verlag, 2012.

[18] T. Weise, M. Zapf, R. Chiong, and A. J. Nebro, "Why is optimization difficult?," in Nature-Inspired Algorithms for Optimisation, R. Chiong, Ed. Berlin: Springer-Verlag, 2009, pp. 1–50.

[19] H. T. Hsieh and C. H. Chu, "Improving optimization of tool path planning in 5-axis flank milling using advanced PSO algorithms," Robotics and Computer-Integrated Manufacturing, vol. 29, no. 3, pp. 3-11, 2013.

[20] N. Geng, D. Gong, and Y. Zhang, "Robot path planning in an environment with many terrains based on interval multi-objective PSO," Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2013), IEEE, pp. 813-820, 2013.

[21] E. Masehian and D. Sedighizadeh, "Multi-objective PSO-and NPSO-based algorithms for robot path planning," Advances in electrical and computer engineering, vol. 10, no. 4, pp. 69-76, 2010.

[22] W. Jiao, G. Liu, J. Zhang, and B. Zhang, "Geomagnetic matching path planning based on PSO algorithm," Systems Engineering–Theory & Practice, vol. 30, no. 11, pp. 2106-2111, 2010 (in Chinese).

[23] I. Chaari, A. Koubaa, H. Bennaceur, S. Trigui, and K. Al-Shalfan, "smartPATH: A hybrid ACO-GA algorithm for robot path planning," Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2012), IEEE, pp. 1-8, 2012.

[24] Y. Sun and R. Zhang, "Research on global path planning for AUV based on GA," in Mechanical Engineering and Technology, T. Zhang, Ed. Berlin: Springer-Verlag, 2012, pp. 311–318.

[25] C. T. Cheng, K. Fallahi, H. Leung, and C. K. Tse, "A genetic algorithm-inspired UUV path planner based on dynamic programming," IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, vol. 42, no. 6, pp. 1128-1134, 2012.

[26] C. Sun and H. Duan, "Artificial bee colony optimized controller for unmanned rotorcraft pendulum," Aircraft Engineering and Aerospace Technology, vol. 85, no. 2, pp. 104-114, 2013.

[27] B. Li, L. Gong, and C. Zhao, "Unmanned combat aerial vehicles path planning using a novel probability density model based on artificial bee colony algorithm," Proceedings of the Fourth International Conference on Intelligent Control and Information Processing (ICICIP 2013), IEEE, pp. 620-625, 2013.

[28] C. Xu, H. Duan, and F. Liu, "Chaotic artificial bee colony approach to Uninhabited Combat Air Vehicle (UCAV) path planning," Aerospace Science and Technology, vol. 14, no. 8, pp. 535-541, 2010.

[29] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," Journal of Global optimization, vol. 39, no. 3, pp. 459-471, 2007.

[30] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," Applied Soft Computing, vol. 8, no. 1, pp. 687-697, January 2008.

[31] H. Li, K. Liu, and X. Li, "A comparative study of artificial bee colony, bees algorithms and differential evolution on numerical benchmark problems," in Computational Intelligence and Intelligent Systems, Z. Cai, H. Tong, Z. Kang, and Y. Liu, Eds. Berlin: Springer-Verlag, 2010, pp. 198-207.

[32] D. Karaboga and B. Akay, "Artificial bee colony (ABC), harmony search and bees algorithms on numerical optimization," Innovative Production Machines and Systems Virtual Conference, 2009.

[33] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," Applied Mathematics and Computation, vol. 214, no. 1, pp. 108-132, 2009.

[34] C. Blum, R. Chiong, M. Clerc, K. De Jong, Z. Michalewicz, F. Neri, and T. Weise, "Evolutionary Optimization," in Variants of Evolutionary Algorithms for Real-World Applications, R. Chiong, T. Weise and Z. Michalewicz, Eds. Berlin: Springer-Verlag, 2012 pp. 1–29.

[35] B. Li and R. Chiong, "A novel artificial bee colony algorithm with a balance-evolution strategy for numerical optimization," unpublished.

[36] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," Applied Mathematics and Computation, vol. 217, no. 7, pp. 3166–3173, 2010.

[37] S. Biswas, S. Das, S. Debchoudhury, and S. Kundu, "Co-evolving bee colonies by forager migration: A multi-swarm based Artificial Bee Colony algorithm for global search space," Applied Mathematics and Computation, vol. 232, pp. 216-234, 2014.

[38] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," Information Processing Letters, vol. 111, no. 17, pp. 871-882, 2011.

[39] W. Gao, S. Liu, and L. Huang, "Enhancing artificial bee colony algorithm using more information-based search equations, Information Sciences, vol. 270, pp. 112–133, 2014.

[40] A. Alizadegan, B. Asady, and M. Ahmadpour, "Two modified versions of artificial bee colony algorithm," Applied Mathematics and Computation, vol. 225, pp. 601-609, 2013.

[41] D. Aydin, S. Özyön, C. Yaşar, and T. Liao, "Artificial bee colony algorithm with dynamic population size to combined economic and emission dispatch problem," International Journal of Electrical Power & Energy Systems, vol. 54, pp. 144-153, 2014.

[42] A. Rajasekhar, R. K. Jatoth, and A. Abraham, "Design of intelligent PID/PI λ D μ speed controller for chopper fed DC motor drive using opposition based artificial bee colony algorithm," Engineering Applications of Artificial Intelligence, vol. 29, no. pp. 13-32, 2014.

[43] F. Kang, J. Li, and H. Li, "Artificial bee colony algorithm and pattern search hybridized for global optimization," Applied Soft Computing, vol. 13, no. 4, pp. 1781-1791, 2013.

[44] B. Li, "Research on WNN modeling for gold price forecasting based on improved artificial bee colony algorithm," Computational Intelligence and Neuroscience, vol. 2014, pp. 1-10, 2014.

[45] W. Gao, S. Liu, and L .Huang, "A novel artificial bee colony algorithm with Powell's method," Applied Soft Computing, vol. 13, no. 9, pp. 3763-3775, 2013.

[46] F. Kang, J. Li, and Z. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions," Information Sciences, vol. 181, no. 16, pp. 3508-3531, 2011.

[47] H. Ding and Q. Feng, "Artificial bee colony algorithm based on Boltzmann selection policy," Computer Engineering and Applications, vol. 31, pp. 53-55, 2009 (in Chinese).

[48] B. Li, Y. Li, and L. Gong, "Protein secondary structure optimization using an improved artificial bee colony algorithm based on AB off-lattice model," Engineering Applications of Artificial Intelligence, vol. 27, no. 1, pp. 70-79, 2014.

[49] J. C. Baker and D. G. Wiencek, Cooperative monitoring in the South China Sea: satellite imagery, confidence-building measures, and the Spratly Islands disputes. Greenwood Publishing Group, 2012.

[50] G. Mei and S. Wu, "Development, Application and Challenge Confronted with Electronic Reconnaissance Satellite," vol. 28, no. 4, pp. 28-31, 2005 (in Chinese).