2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

# An AUVs Path Planner using Genetic Algorithms with a Deterministic Crossover Operator

Chi-Tsun Cheng†, Kia Fallahi‡, Henry Leung‡, and Chi K. Tse†
†Department of Electronic and Information Engineering,
The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong.
‡Department of Electrical and Computer Engineering,
The University of Calgary, Calgary, AB, Canada.

*Abstract*— **Path planning is an optimization process in which a path between two points is to be found that results in a user-defined optimum satisfaction of a given set of requirements. For small scale path planning, exact algorithms such as linear programming and dynamic programming are usually adopted which are able to give optimum solutions in short time. However, due to their memory intensive nature and computational complexity, exact algorithms are not applicable for medium to large scale path planning. Meta-heuristic algorithms such as evolutionary algorithms can provide sub-optimum solution without the full understanding of the search space and are widely used in large-scaled path planning. However, extra precautions are needed to avoid meta-heuristic algorithms from being trapped in local optimum points. In this paper, a path planner combining genetic algorithms (GA) with dynamic programming (DP) is proposed to solve an autonomous underwater vehicles (AUVs) path planning problem. The proposed path planner inherits the speed of exact algorithms and the scalable nature of meta-heuristic algorithms. Simulation results show that when comparing with conventional GA-based path planners, the proposed path planner can greatly improve the convergence rate and solution quality.**

## I. Introduction

Path planning (or trajectory planning) techniques have been widely applied in guiding robots to travel through different spaces. In general, path planning refers to the process of finding a feasible path between two points in a bounded environment, provided that some user-defined constraints are being satisfied. Path planning is usually carried out off-line with the help of existing knowledge about the environment. A classical example of path planning problems is the *Piano Mover's Problem* [1]. In recent years, applications of path planning have been further extended to navigate autonomous underwater vehicles (AUVs) [2], [3] and unmanned aerial vehicles (UAVs) [4], [5]. Trajectories of the autonomous vehicles are optimized with respect to time, distance, fuel consumption, and other interested aspects provided that they are collision free [6].

Path planning problems have been attempted by using different computational algorithms. Basically, two major types of path planning exist. They are *combinatorial* and *sampling-based* path planning [7], [8]. Exact algorithms such as Linear Programming (LP) and Dynamic Programming (DP) are often adopted in combinatorial path planning. Exact algorithms are complete algorithms, which mean they can always provide a globally optimum solution to an optimization problem. Linear programming has been well studied and the implementation is straightforward [9], [10]. However, because of its computational complexity, linear programming is not applicable in applications with large search spaces. Dynamic programming is capable of partitioning an optimization problem into stages, reducing the computational complexity [11]. Dynamic programming is widely used in solving combinatorial optimization problems. However, high computing power is still required especially for multi-dimensional problems.

Meta-heuristic algorithms, such as evolutionary algorithms (EA), have extensively been used in sampling-based path planning [12], [13], [14], [15].They are global optimization algorithms which are able to provide decent results even when the problem dimension is high. However, due to their meta-heuristic properties, these methods are incomplete and require careful selection of parameters in order to avoid being trapped in local optimum points.

In this paper, a path planner which combines genetic algorithms (GA) with dynamic programming is proposed for AUVs navigation. Simulation results show that the proposed path planner can achieve a higher convergence rate and provide solutions with better fitness than conventional GA-based path planners. The rest of the paper is organized as follows. Section II defines the path planning problem being tackled. Brief descriptions of GA and DP are given in Section III. Section IV describes the proposed path planner. Simulation results are shown in Section V, where the proposed path planner is compared against a conventional GA-based path planner. Finally, Section VI concludes this paper.

## II. Problem Formulation

The path planning problem in this paper is concerned with finding a path in a bounded terrain between an arbitrary starting point $p_s$ and an arbitrary ending point $p_e$, provided that the path is optimized according to a fitness function.

### A. Scenario Under Study

To model different underwater scenarios, two seabed landscape models are considered in this paper. The landscape models under study are meshed 3-D models mimicking hilly landscapes. The two models, model 1 and model 2, can be expressed by the following mathematical functions

TABLE I

PARAMETERS USED IN LANDSCAPE MODELS.

| Model 1 | | Model 2 | |
|---|---|---|---|
| Parameters | Values | Parameters | Values |
| $a_1$ | 0.5 | $a_2$ | 1.2 |
| $b_1$ | 1.3 | $b_2$ | 0.8 |
| $c_1$ | -0.3 | $c_2$ | -1.7 |
| $d_1$ | 3.0 | $d_2$ | 1.0 |
| $e_1$ | -2.0 | $e_2$ | -1.0 |

respectively:

$$z_1(x,y) = \sigma_1\big(\sin(4\pi y + a_1) + b_1\cos(4\pi x) + c_1\sin(d_1\sqrt{(4\pi y)^2 + (4\pi x)^2}) + e_1\sin(4\pi y)\big)^2 \quad (1)$$

$$z_2(x,y) = \sigma_2\big(\cos(4\pi y + a_2) + b_2\sin(4\pi x) + c_2\sin(d_2\sqrt{(4\pi y)^2 + (4\pi x)^2}) + e_2\cos(4\pi y)\big)^2 \quad (2)$$

where $a_1$, $b_1$, $c_1$, $d_1$, $e_1$, $a_2$, $b_2$, $c_2$, $d_2$, and $e_2$ are arbitrary constants. Their values are shown in Table I. Parameters $\sigma_1$ and $\sigma_2$ are the normalizing factors such that $z_1(x,y)$ and $z_2(x,y)$ will lie within the range [0,1]. Illustrations of the model 1 and 2 are shown in Figs 1 and 2, respectively. The base of each model is a square of 1 unit$^2$ located on the $x$-$y$ plane. The center of the base is located at (0,0).

### B. B-spline Curves

In this paper, paths of AUVs are represented by B-spline curves. The idea of B-spline curves was given by Schoenberg in the 1940's. A B-spline curve is a piecewise polynomial curve comprising a number of polynomial segments. The continuity nature of B-spline curves makes them most suitable for the representation of aircraft and watercraft trajectories. A B-spline curve can be constructed using de Boor's algorithm. Details on constructing a B-spline curve can be obtained in the literature on spline studies [16], [17]. Basically, the trajectory of a spline model can be controlled by adjusting the locations of its control points. In this paper, the goal of a path planner is to optimize the fitness of a path by adjusting the control points' locations.

### C. Fitness of a Path Segment

The fitness of a path segment is expressed as the inverse of its cost value $\delta$. The cost value is the weighted sum of three constraint values $v_i$, where $i = 1, 2, 3$. The cost value is expressed as:

$$\delta = \frac{\sum_{i=1}^{3} w_i v_i}{\sum_{i=1}^{3} w_i} \quad (3)$$

where $w_i$ are the weightings of the constraints values. The optimization problem is to minimize the cost value and thus maximize the fitness of the paths. To save time and energy, a shorter path segment is more desirable than a longer path segment. Constraint value $v_1$ is equal to the length of the path which is expressed as

$$v_1 = d \quad (4)$$

where $d$ represents the length of the path segment. Paths consisting of sharp turnings are undesirable for the navigation of AUVs. In this paper, an AUV trajectory is formed using a B-spline curve with a number of control points $p_i$. A turning angle $\theta$ is defined as the acute angle formed by any three consecutive control points as illustrated in Fig. 3. Constraint value $v_2$ is defined as the complement of the normalized minimum turning angle $\theta_{\min}$ in a path segment, which is expressed as

$$v_2 = 1 - \frac{\theta_{\min}}{\pi} \quad (5)$$

In most applications, an AUV is required to sail up and down according to the landscape of the seabed. Paths with frequent climbing and descending motions are undesirable in the sense of fuel consumption. Constraint value $v_3$ is proportional to

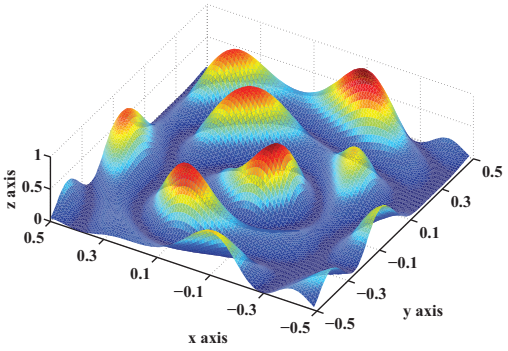


Fig. 2. Landscape model 2

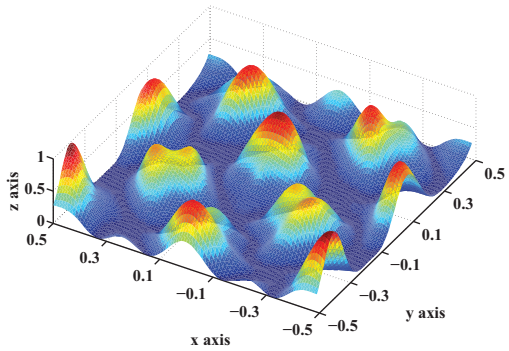


Fig. 1. Landscape model 1

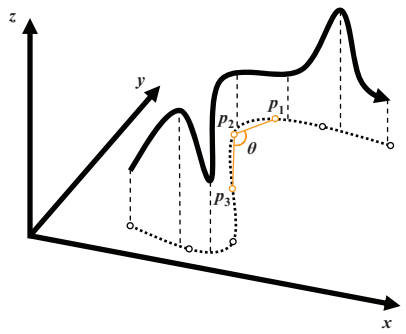


Fig. 3. Illustration of a turning angle formed by three consecutive control points $p_1$, $p_2$, and $p_3$.

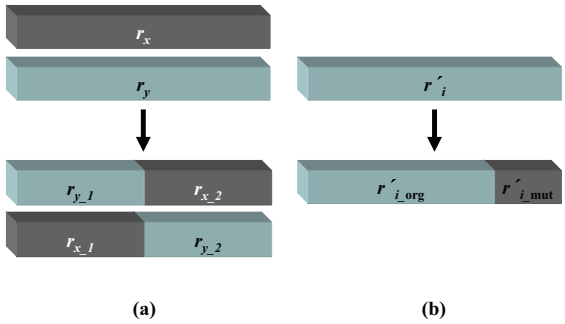


Fig. 4. Two main processes in genetic algorithm: (a) crossover; (b) mutation.

the maximum elevation rate, which is defined as the greatest magnitude of the change in the altitude of an AUV over its displacement in the $x$-$y$ plane. Here, $v_3$ is expressed as

$$v_3 = \underset{t\in[0,t_{\max}-\Delta t]}{\arg\max} \left(\frac{z_{t+\Delta t} - z_t}{\sqrt{(x_{t+\Delta t} - x_t)^2 + (y_{t+\Delta t} - y_t)^2}}\right) \quad (6)$$

where $t_{\max}$ is the total time needed by an AUV to get through the path segment.

## III. Background Knowledge

### A. Genetic Algorithm

According to the principle of *natural selection*, the more adaptable an individual is to its environment, the higher the chance for it to survive and produce its offspring. The adaptability of an individual to its environment is measured by its *fitness*. By means of exchanging genes with other parties of better fitness, an individual can produce offspring with better fitness and higher probability of survival. Genetic algorithms (GA) are a class of evolutionary algorithms which mimic the natural selection in biological populations [18]. With the crossover and mutation operators, GA can provide promising results for large-scale optimization problems even without the full understanding of the problem being tackled. The operations of generic GA can be briefly summarized as follows.

1) The algorithm begins with the generation of $b$ potential solutions $r_i$ (where $i = 1, 2, \cdots b$) which are called *chromosomes* and form the initial population $\mathcal{P}$. This set of chromosomes is generated in a random manner.
2) Each individual chromosome $r_i$ will then pass through a fitness evaluation function $f_{\text{fitness}}(r_i)$. The fitness evaluation function is a function which measures how feasible a chromosome is to the optimization problem. The chromosomes are then ranked according to their fitness values.
3) A pair of chromosomes, $r_x$ and $r_y$, are selected to perform crossover operation. Different selection methods are available such as *roulette wheel* and *rank selection.* The aim is to increase the chance for a chromosome of high fitness to be selected.
4) A pair of selected chromosomes, $r_x$ and $r_y$, are regarded as *parents* and are fed into a crossover operator. The crossover operator is an operator to exchange information carried by the two parents. The operator will select crossover points between them. The method for selecting crossover points may vary from application to application. For instance, if a single crossover point $cp_1$ is found, the operator will then cut $r_x$ and $r_y$ at $cp_1$ and form 4 chromosome segments $r_{x\_1}$, $r_{x\_2}$, $r_{y\_1}$ and $r_{y\_2}$. These 4 segments will recombine as $r_{x\_1}$–$r_{y\_2}$ and $r_{y\_1}$–$r_{x\_2}$ and become the children of the parents. An illustration of this process is shown in Fig. 4 (a).
5) Repeat procedures 3 to 4 for $h$ times, where $h = b/2$. Thus, $b$ new chromosomes $r'_i$ (where $i = 1, 2, \cdots b$) are generated and a new population $\mathcal{P}'$ is formed. Within the new population, a small portion of chromosomes are randonly chosen for the mutation process. The mutation process will remove a segment of the selected chromosome and replace it with a new segment which is usually generated in a random manner. The idea behind the mutation process is to prevent the solutions from being trapped in local optimum points. After mutation, these mutated chromosomes will be put back into the population $\mathcal{P}'$. An illustration of this process is shown in Fig. 4 (b).
6) Set $\mathcal{P} \leftarrow \mathcal{P}'$. Repeat procedures 2 to 5 for several times. By repeating the above processes, sub-optimum chromosomes with lower fitness will hopefully be eliminated and reduce the variety in the population. The above procedures repeat until the population has converged to a predefined value. The most frequently appeared chromosome in the final population is the result of the optimization.

### B. Dynamic Programming

The idea of dynamic programming was proposed by Richard Bellman in the 1940s. Dynamic programming is an optimization method for making a sequence of interrelated decisions such that the overall outcome is optimized [19], [20]. The idea of dynamic programming is based on the *principle of optimality* which states that for a sequence of decisions to be optimum, decision made in each stage has to be optimized. The gist of dynamic programming is in breaking down an optimization problem into a sequence of sub-problems and tackling each sub-problem separately. Comparing with linear programming, dynamic programming is much less computationally exhaustive.

The operation of dynamic programming can be explained as follows. Consider an optimization problem with $k$ intermediate stages $(s_1, s_2, \cdots, s_k)$, with $n$ decisions $(x_1, x_2, \cdots, x_n)$ available in each stage (Fig. 5). Suppose the current stage is $s_i$. Then, there exists a vector $F(s_i)$ such that

$$F(s_i) = [f_c(s_0, s_i, x_1) \cdots f_c(s_0, s_i, x_j) \cdots f_c(s_0, s_i, x_n)]^{\mathrm{T}} \quad (7)$$

where $f_c(s_0, s_i, x_j)$ is the total cost of the sequence of optimum decisions made from stage $s_0$ to $s_i$, provided that the decision made in stage $s_i$ is $x_j$. In general, the

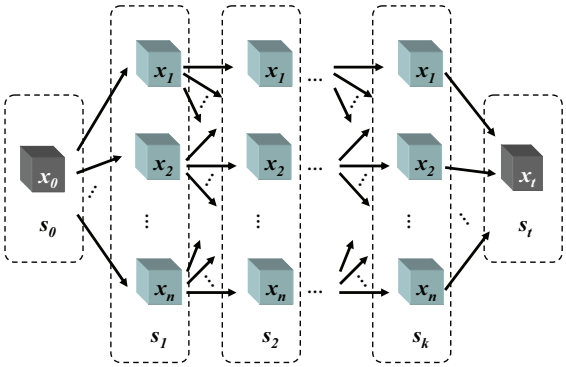


Fig. 5. Dynamic programming. Stage $s_0$ is the initial stage and stage $s_t$ is the target stage.

optimization problem for an intermediate stages $s_{i+1}$ can be written as

$$f_c(s_0, s_{i+1}, x_p) = \min_j (f_c(s_0, s_i, x_j) + f_c(s_i, s_{i+1}, x_p)) \tag{8}$$

where $p = 1, 2, \cdots, n$, $j = 1, 2, \cdots, n$ and decision $x_j$ is the decision made in stage $s_i$. The optimization problem for the target stage $s_t$ is expressed as

$$f_c(s_0, s_t, x_t) = \min_j (f_c(s_0, s_k, x_j) + f_c(s_k, s_t, x_t)) \tag{9}$$

where $j = 1, 2, \cdots, n$ and $x_t$ represents the decision leading to the goal of the optimization. Using (8) and (9), the optimization procedure moves forward stage by stage, and each time it finds the optimum solution for that stage and eventually arrives at the optimum solution when it finishes at the final stage $s_t$.

## IV. The Proposed AUVs Path Planner

In generic GA, the populations are generated in a random manner. The path generator repeats for $d$ times to generate a population of $d$ chromosomes, where $d \in \mathbb{Z}^+$. After the population is generated or updated, two chromosomes are selected randomly at a time to perform crossover. When there is more than one possible crossover point, the crossover point will be randomly selected. Although a chromosome with better fitness will have a higher chance of being selected in the selection process, offspring of the next generation may not necessarily have better fitness than those in previous generations due to the randomized process mentioned above. This problem can be alleviated by comparing the fitness of the chromosomes across two generations and allow only those chromosomes with better fitness to survive. The drawback of such policy is a higher probability of having an immature convergence in the optimization.

Another way to solve this problem is to break chromosomes at some common crossover points, decompose the chromosomes into segments and construct the best solution out of these chromosome segments. In this way, the operation becomes a combinatorial optimization problem which can be solved by dynamic programming [21], [22].

In the first generation, the population generator will generate $n$ crossover points, i.e. $\mathrm{cop}_1$, $\mathrm{cop}_2$, $\cdots$ , $\mathrm{cop}_n$. Each crossover point is located randomly inside the terrain under study. Here, $p_s$ and $p_e$ are also considered as crossover points which are relabeled as $\mathrm{cop}_s$ and $\mathrm{cop}_e$, respectively. Among these $n + 2$ crossover points, any two crossover points ($\mathrm{cop}_i$,$\mathrm{cop}_j$) are connected by a path segment $\kappa_{i,j}$, which is constructed using B-spline. The number of control points used to construct each B-spline path segment is a random positive integer. Similar to the crossover points, the positions of the control points are randomly placed inside the terrain under study. The path segments are then fed into a deterministic crossover operator.

The deterministic crossover operator is used to replace the random crossover operator in conventional genetic algorithms. The working principle of the deterministic crossover operator is based on the operation of forward dynamic programming. Instead of selecting two chromosomes at a time, the supervised crossover operator will put the whole population under consideration. The operation of the supervised crossover operator can be illustrated using the following example.

Suppose, including $\mathrm{cop}_s$ and $\mathrm{cop}_e$, there exists $4$ crossover points, i.e., $\mathrm{cop}_s$, $\mathrm{cop}_1$, $\mathrm{cop}_2$, and $\mathrm{cop}_e$. The operator will evaluate the fitness $\delta$ of all the path segments associated with the crossover points. To avoid loops in the crossover outcomes,

1) paths leading toward $\mathrm{cop}_s$ will be regarded as invalid and have zero fitness values.
2) paths with zero displacement, except $\kappa_{\mathrm{cop}_e,\mathrm{cop}_e}$, will have zero fitness values.
3) at the end of the crossover process, except $\mathrm{cop}_e$, a crossover point can only be appeared once in a resultant path.

To keep all the solutions ended at the target location, all paths going out from $\mathrm{cop}_e$ will again be regarded as invalid and have zero fitness values. The fitness values of all the path segments will be stored into the fitness matrix $\mathcal{D}$ which is shown as follow:

$$\begin{bmatrix} \delta(\mathrm{cop}_s, \mathrm{cop}_s) & \delta(\mathrm{cop}_s, \mathrm{cop}_1) & \cdots & \delta(\mathrm{cop}_s, \mathrm{cop}_e) \\ \delta(\mathrm{cop}_1, \mathrm{cop}_s) & \delta(\mathrm{cop}_1, \mathrm{cop}_1) & \cdots & \delta(\mathrm{cop}_1, \mathrm{cop}_e) \\ \vdots & \vdots & \ddots & \vdots \\ \delta(\mathrm{cop}_e, \mathrm{cop}_s) & \delta(\mathrm{cop}_e, \mathrm{cop}_1) & \cdots & \delta(\mathrm{cop}_e, \mathrm{cop}_e) \end{bmatrix}$$

The fitness matrix $\mathcal{D}$ will be used in the dynamic programming shown in Fig. 6. Beginning from the starting location $\mathrm{cop}_s$, one can either choose $\mathrm{cop}_1$, $\mathrm{cop}_2$ or $\mathrm{cop}_e$ as the crossover point in stage 1. Therefore, a graph can be constructed as shown in step I. In step II, suppose $\mathrm{cop}_1$ is the selected crossover point in stage 2, the crossover point in the stage 1 can either be $\mathrm{cop}_2$ or $\mathrm{cop}_e$. It can be shown that if $\mathrm{cop}_1$ is selected as the crossover point in stage $s_2$, route $\mathrm{cop}_s \rightarrow \mathrm{cop}_2 \rightarrow \mathrm{cop}_1$ is the one with the highest fitness. This intermediate best route is saved for later use. Step III to step IV execute the same procedures, with $\mathrm{cop}_2$

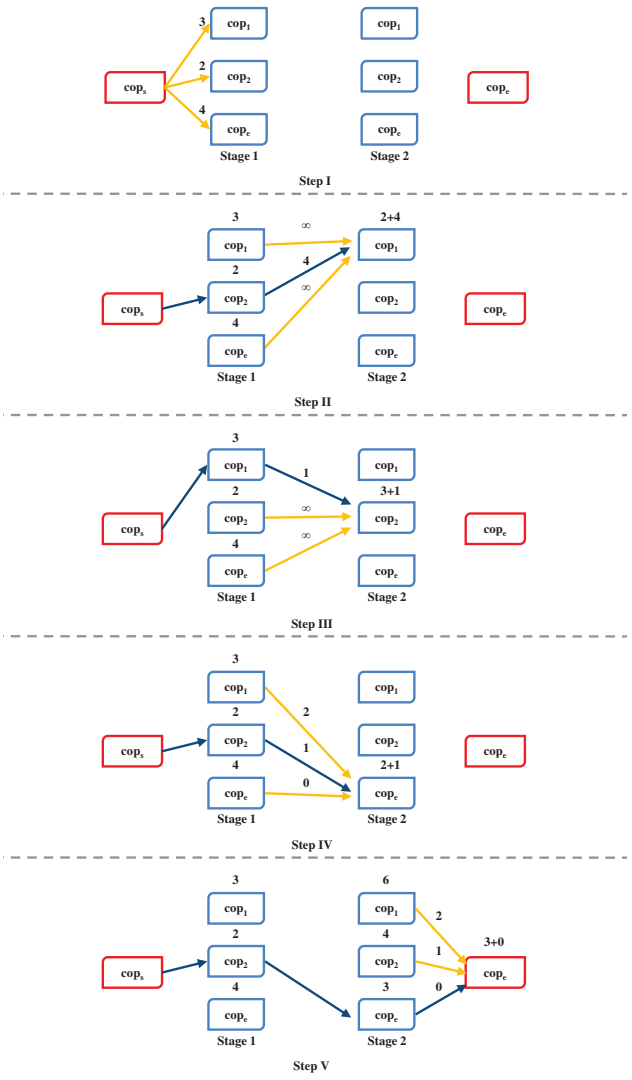


Fig. 6. Operation of the proposed crossover operator.

and $\mathrm{cop}_e$ tentatively selected as the crossover point in stage $s_2$, respectively. The final step is to direct all intermediate routes toward the target location. In this example, the route $\mathrm{cop}_s \rightarrow \mathrm{cop}_2 \rightarrow \mathrm{cop}_e$ is having the highest fitness and it is indeed the optimum path of the current population.

All path segments and crossover points that are not associated with the optimum path will be discarded at the end of the generation. In the previous example, only $\mathrm{cop}_s$, $\mathrm{cop}_2$, and $\mathrm{cop}_e$ will be passed to the next generation. In the next generation, another $n$ crossover points will be generated. Path segments will again be generated to connect any two crossover points. Notice that if a path segment between two points has been generated in the previous generations, the proposed generator will compare the fitness of the old path segment with the new one. If the new one has higher fitness than the old one, the new path segment will be used to replace the old one, and vice versa. Such feature can be regarded as the mutation operator in conventional genetic algorithms. The above procedures keep on repeating until the maximum generation number has reached.

TABLE II

PARAMETERS USED IN SIMULATIONS.

| Parameters | Values |
|---|---|
| Crossover points per path | 2–10 |
| B-spline construction method | de Boor's algorithm |
| B-spline order | 4 |
| Control points per path segment | 4–10 |
| $w_1$ | 1 |
| $w_2$ | 2 |
| $w_3$ | 4 |
| Maximum no. of generation | 100 |

TABLE III

PARAMETERS USED BY THE PATH PLANNERS.

| Parameters | GA-based | Proposed |
|---|---|---|
| Population generation method | Random | N/A |
| Crossover points added per generation | 8 | 8 |
| Chromosomes per generation | 20 | N/A |
| Selection method | Roulette wheel | N/A |
| Crossover method | Random | Deterministic |
| Mutation probability | 20 | N/A |

## V. SIMULATIONS

In this section, the performance of proposed path planner is evaluated using computer simulations. For comparison purposes, a conventional GA-based path planner is introduced into the simulation and serves as the reference. At the beginning of each simulation, the conventional GA-based path planner generates a number of crossover points. Chromosome segments (incomplete chromosomes) are then formed to connect each pair of crossover points. The population generator constructs complete chromosomes by concatenating crossover points and their associated chromosome segments together in a random manner. The complete chromosomes are then fed into the selection, crossover, and mutation operators as mentioned in Section III-A. At the end of a generation, a chromosome with the highest fitness will be the only "survivor" of the population. Crossover points associated with this chromosome are preserved for the next generation. The remaining crossover points are discarded. In the next generation, new crossover points are added. The operation continues until the maximum generation number is reached.

The performance of the proposed path planner is judged by comparing the constraint variables of the paths generated by the two path planners. Simulations are performed in Matlab. In the simulations, a path planner is required to navigate an AUV to sail across the landscape models defined in Section II. The starting point $p_s$ and the ending point $p_e$ of the AUVs are located at (-0.25,-0.50) and (0.25,0.50), respectively. The parameters used in the simulations are shown in Table II. The parameters used by the conventional GA-based and the proposed path planners are shown in Table III. A simulation lasts for 100 generations. At each

TABLE IV

SIMULATION RESULTS OF LANDSCAPE MODELS 1 AND 2, WITH $w_1 = 1$,$w_2 = 2$, AND $w_3 = 4$.

| | Landscape Model 1 | | | | | | Landscape Model 2 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Criteria | PL | | ER | | MT | | PL | | ER | | MT | |
| Gen. | GADPBS | GABS | GADPBS | GABS | GADPBS | GABS | GADPBS | GABS | GADPBS | GABS | GADPBS | GABS |
| 10 | 1.637 | 2.664 | 2.891 | 3.515 | 56.470 | 38.253 | 1.764 | 3.985 | 2.349 | 3.097 | 59.546 | 23.052 |
| 20 | 1.541 | 2.621 | 2.013 | 2.692 | 55.765 | 37.363 | 1.675 | 3.493 | 2.228 | 2.606 | 80.029 | 26.349 |
| 30 | 1.441 | 2.332 | 1.694 | 3.617 | 59.816 | 44.806 | 1.670 | 3.167 | 2.197 | 2.219 | 79.020 | 32.983 |
| 40 | 1.441 | 2.806 | 1.694 | 3.391 | 59.816 | 31.171 | 1.578 | 3.405 | 2.416 | 2.121 | 90.361 | 27.928 |
| 50 | 1.522 | 2.591 | 1.416 | 2.498 | 57.824 | 44.102 | 1.605 | 2.364 | 2.483 | 1.902 | 85.344 | 53.974 |
| 60 | 1.480 | 2.572 | 1.488 | 1.572 | 62.850 | 41.884 | 1.589 | 2.380 | 2.041 | 2.184 | 83.765 | 57.008 |
| 70 | 1.480 | 2.391 | 1.488 | 1.116 | 62.850 | 42.687 | 1.560 | 2.380 | 1.647 | 2.184 | 76.252 | 57.008 |
| 80 | 1.496 | 2.317 | 1.454 | 1.487 | 61.721 | 43.914 | 1.547 | 2.380 | 1.454 | 2.184 | 81.924 | 57.008 |
| 90 | 1.508 | 2.301 | 1.229 | 0.837 | 59.806 | 38.722 | 1.556 | 2.640 | 1.659 | 2.543 | 80.411 | 52.728 |
| 100 | 1.486 | 2.342 | 1.116 | 0.868 | 50.284 | 38.354 | 1.472 | 2.460 | 1.672 | 2.225 | 88.268 | 52.728 |

generation, the path with the highest fitness value is selected. The path length, maximum elevation rate, and minimum turning angle of the selected path at every 10 generations are recorded and shown in table IV. Results presented in this paper are taken from the average of 50 individual simulations.

According to the simulation results, the proposed path planner can achieve a much higher convergence rate than the conventional GA-based path planner. Nevertheless, the proposed path planner can optimize all constraint variables much better than the conventional GA-based path planner.

## VI. CONCLUSION

In this paper, an AUVs path planner using genetic algorithms with a deterministic crossover operator is proposed. The path of each AUV is represented using a B-spline curve with a number of control points. In the proposed path planner, the random crossover operator in conventional genetic algorithms is replaced by a deterministic crossover operator which is based on the operation of dynamic programming. The path length, minimum turning angle, and maximum elevation rate have been optimized simultaneously by using the proposed path planner. With the deterministic crossover operator, the proposed path planner can always construct the best solution out of the population given. Comparing with conventional genetic algorithm based path planner, the proposed path planner has a much higher convergence rate and provides solutions with better fitness.

## REFERENCES


[1] J. H. Reif, "Complexity of the mover's problem and generalizations," in *20th Annual Symp. Foundations of Computer Science*, San Juan, USA, October 1979, pp. 421–427.

[2] K. P. Carroll, S. R. McClaran, E. L. Nelson, D. M. Barnett, D. K. Friesen, and G. N. William, "AUV path planning: an A* approach to path planning with consideration of variable vehicle speeds and multiple, overlapping,time-dependent exclusion zones," in *Proc. Symp. Autonomous Underwater Vehicle Technology, (AUV '92)*, Washington, USA, June 1992, pp. 79–84.

[3] H.-J. Wang, X.-Q. Bian, X. Zhang, M.-Y. Fu, and J. Li, "Two approaches for autonomous underwater vehicle global path planning in large range ocean environment," in *Proc. Int. Conf. Intelligent Mechatronics and Automation, (ICIMA 2004)*, Chengdu, China, August 2004, pp. 224–227.

[4] S. A. Bortoff, "Path planning for UAVs," in *Proc. American Control Conf., (ACC2000)*, vol. 1, Chicago, USA, June 2000, pp. 364–368.

[5] Y.-H. Qu, Q. Pan, and J.-G. Yan, "Flight path planning of UAV based on heuristically search and genetic algorithms," in *Proc. Annual Conf. IEEE Industrial Electronics Society, (IECON'05)*, Raleigh, USA, November 2005, pp. 45–49.

[6] N. Shahidi, H. Esmaeilzadeh, M. Abdollahi, and C. Lucas, "Memetic algorithm based path planning for a mobile robot," *Int. Jour. Information Technology*, vol. 1, no. 3, pp. 100–103, 2005.

[7] J. C. Latombe, *Robot motion planning*. Norwell, USA: Kluwer, 2005.

[8] S. M. LaValle, *Planning Algorithm*. New York, USA: Cambridge University Press, 2006.

[9] B. Cetin, M. Bikdash, and F. Y. Hadaegh, "Hybrid mixed-logical linear programming algorithm for collision-free optimal path planning," *IET Control Theory and Applications*, vol. 1, no. 2, pp. 522–531, March 2007.

[10] G. C. Chasparis and J. S. Shamma, "Linear-programming-based multi-vehicle path planning with adversaries," in *Proc. American Control Conf., (ACC2005)*, vol. 2, Portland, USA, June 2005, pp. 1072–1077.

[11] H. Hu and M. Brady, "Dynamic global path planning with uncertainty for mobile robots in manufacturing," *IEEE Trans. Robotics and Automation*, vol. 13, no. 5, pp. 760–767, October 1997.

[12] J. Xiao, Z. Michalewicz, L. Zhang, and K. Trojanowski, "Adaptive evolutionary planner/navigator for mobile robots," *IEEE Trans. Evolutionary Computation*, vol. 1, no. 1, pp. 18–28, April 1997.

[13] I. K. Nikolos, K. P. Valavanis, N. C. Tsourveloudis, and A. N. Kostaras, "Evolutionary algorithm based offline/online path planner for uav navigation," *IEEE Trans. Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 33, no. 6, pp. 898–912, December 2003.

[14] T. W. Manikas, K. Ashenayi, and R. L. Wainwright, "Genetic algorithms for autonomous robot navigation," *IEEE Instrumentation and Measurement Mag.*, vol. 10, no. 6, pp. 26–31, December 2007.

[15] A. Alvarez, A. Caiti, and R. Onken, "Evolutionary path planning for autonomous underwater vehicles in a variable ocean," *IEEE Jour. Oceanic Engineering*, vol. 29, no. 2, pp. 418–429, April 2004.

[16] H. Späth, *One Dimensional Spline Interpolation Algorithms*. Natick, USA: A K Peters, Ltd., 1995.

[17] L. Piegl and W. Tiller, *The NURBS Book*, 2nd ed. New York, USA: Springer-Verlag, 1997.

[18] C. R. Reeves and J. E. Rowe, *Genetic Algorithms - Principles and Perspectives: A Guide to GA Theory*. Norwell, USA: Kluwer, 2003.

[19] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 8th ed. Boston, USA: McGraw-Hill Higher Education, 2005.

[20] H. S. Kasana and K. D. Jumar, *Introductory Operations Research*. Berlin, Germany: Springer-Verlag, 2004.

[21] Ö. Ergun and J. B. Orlin, "A dynamic programming methodologies in very large scale neighborhood search applied to the traveling salesman problem," *Discrete Optimization*, vol. 3, no. 1, pp. 78–85, March 2006.

[22] R. L. Carraway, T. L. Morin, and H. Moskowitz, "Generalized dynamic programming for stochastic combinatorial optimization," *Operations Research*, vol. 37, no. 5, pp. 819–829, September 1989.