Abstract—This paper studies the path planning problem for mobile robots to move smoothly and safely along a shorter curvature-constrained path in completely known dynamic environments. The cost of travel is defined by an obstacle-avoidance cost, which is designed as a weighted penetration depth to vertices of obstacles, and a length cost. The path is composed of a pre-specified number of cubic spiral segments. The intermediate configurations are generated by evolutionary multi-objective optimization aiming for smooth and shorter collision-free paths. Comparison of path planning performance of two different intrinsic cost definitions based on island-based parallel genetic algorithm (IPGA) with migration are conducted in terms of success rate in separate runs and path length whenever a collision-free path can be found. Results are presented for simulated environments containing three distinct types of obstacles: polygons, walls and their combinations.

I. INTRODUCTION

Path planning of a mobile robot has been under studied in recent years since the robot is becoming so important in this high technology trend. In the past few decades, there are a lot of papers talking about no matter global or local path planning with many complicated numerical methods [2][3]. These methods sometimes can not be implemented into an environment that consists of many obstacles. So this leads the application of evolutionary computing into the problem.

There are a lot of papers implement the genetic algorithm to plan the path for a mobile robot [4]-[7], actually they treat a problem that robot can turn any direction in the origin. In our previous work [8], we implement the parallel genetic algorithm based on the island model (IPGA) to generate multiple collision-free paths composed by a pre-specified number of cubic spirals. In reality, the planner of a mobile robot must not only be able to generate a feasible path but also a path with minimum length as possible, this leads to a multi-objective optimization problem (MOP). Most real-world optimization problems always involve multiple objectives, unfortunately, these objectives sometimes conflict with each other. In [9], we extended our previous work to plan a safe and shorter path via multi-objective evolutionary algorithms.

Path planning in the dynamic environments is indeed a challenging problem. Many other researches solved the dynamical problem with the numerical methods [10] and evolutionary methods [11, 12]. Actually, these researches solve the path planning problem with complicated mathematical computations and the planned paths are non-smooth since they compose piece-wise line segment. In this paper, we will plan a smooth path for mobile robot with cubic spiral in not only static but also dynamic environments.

In the following paragraph, Section II would briefly introduce the cubic spiral method. The proposed GA-based evolutionary searching algorithm (IPGA) will be mentioned in Section III. Then we proposed some simulations in Section IV. Finally, we make a conclusion in Section V.

II. PRELIMINARIES

Let \( q = (x, y, \theta) \) represents a configuration where \((x, y)\) and \(\theta\) denotes the position and orientation, respectively, of the mobile robot. The path followed by a unit-speed mobile robot starting from the initial configuration \((x_0, y_0, \theta_0)\) is governed by integrating the nonholonomic kinematic constraints,

\[
\begin{align*}
\theta(s) &= \theta_0 + \int_0^s \kappa(t) dt \\
x(s) &= x_0 + \int_0^s \cos(\theta(t)) dt \\
y(s) &= y_0 + \int_0^s \sin(\theta(t)) dt
\end{align*}
\]

where \(x, y\) and \(\theta\) represent the function of position in \(x\)-axis and \(y\)-axis and orientation of robot through a path, \(s\) is the path length, and is set as \(0\) at the initial point of robot \((x_0, y_0)\), as Fig. 1 depicted. The \(\kappa(t)\) can be defined in the following paragraph (eqn. 3).

A. Two objectives for path optimization

The cost of travel is path length and the cost of obstacle avoidance, which can be represented by the intrinsic cost by assuming that the robot is a point to avoid time-consuming collision detection between rigid objects [3]. When used in environments of varying geometric shapes and sizes, we see some defects in the original definition of intrinsic cost, i.e. only calculates how many intersections. As Fig. 1 shown, similar paths (represented by solid and dashed lines) cross a thin or a large obstacle (compared to the length of a path segment) will have the same intrinsic cost. Nonetheless, the
solid path is visually more proximate to collision-free than the dashed path. To invoke the measure of proximity to collision-free into a path to be evaluated, we design a modified intrinsic cost in the following. Refer to Fig. 1. Suppose there are \( n \) sampling nodes \( q_{1}, \ldots, q_{n} \) of a path segment intersecting a designated obstacle. For convenience, every path segment of different length is sampled as the same number of nodes \( N_{s} \). The normalized intrinsic cost for \( i \)-th path segment is defined as:

\[
 r_{i} = n_{i} / N_{s}, i = 1, \ldots, N_{seg}
\]

where \( n_{i} \) represents the number of nodes of \( i \)-th path segment that are within a designated obstacle, \( N_{seg} \) is a prespecified number of path segments. For each segment, we measure the penetration depth as the minimum distance from the middle node \( q_{m} \) to all vertices of the intersecting obstacle, \( d_{min}^{m} \), as the proximity to collision-free. Note that if the number of intersected nodes is odd, \( q_{m} \) would be unique. On the other hand, if the number is even, there are two median nodes, i.e. \( q_{m} \) and \( q_{m+1} \), to derive two minimum distances, \( d_{min}^{m} \) and \( d_{min}^{m+1} \). Their arithmetic average is the proximity to collision-free. The general form of newly designed intrinsic cost function can thus be defined as:

\[
f_{new} = \sum_{i=1}^{N_{seg}} r_{i} \cdot D_{i}
\]

where \( r_{i} \) and \( D_{i} \) of \( i \)-th path segment are defined as:

\[
r_{i} = n_{i} / N_{s} \quad (\text{normalized intrinsic cost})
\]

\[
 D_{i} = \left\{ \begin{array}{ll}
 (d_{min}^{m}), & \text{if } n_{i} \text{ is odd} \\
 \left( \left( d_{min}^{m} \right)^{2} + \left( d_{min}^{m+1} \right)^{2} \right) / 2, & \text{if } n_{i} \text{ is even}
\end{array} \right.
\]

where subscript \( i \) denotes \( i \)-th path segment.

For dynamic cases, we can compute time sequence of all sampling points of the path according to the predefined velocity of robot. After we get the time sequence of robot, we can modify the positions of all obstacles according to their own velocities when the robot moves around, it implies that we are aware of the detail information of the dynamic environment during every specific moment. The main difference between static and dynamic cases is that former arbitrates the whole path in one time, and the later arbitrates every single points of the path in different time indices when compute the intrinsic cost.

B. Review of Cubic Spiral Method

For smooth path generation, the path is made up of cubic spiral segments, which is curvature continuous.

1. Cubic Spiral: By definition, cubic spiral is a set of trajectories that the direction function \( \theta \) is a cubic polynomial of curve length \( l \). Its angle, which describes how much the curve turns from the initial orientation to final orientation, is denoted by

\[
\alpha = \theta(l) - \theta(0)
\]

Based on equation (1) and the boundary conditions at \( s=0 \) and \( s=l \), we have (Lemma 2, [1]),

\[
\kappa(s) = \frac{6\alpha}{l^{\frac{3}{2}}} \cdot (l - s)
\]

If the length of a cubic spiral is 1, its size is given by (Lemma3, [1])

\[
D(\alpha) = \frac{2}{3} \int_{0}^{1/2} \cos \left( \alpha \left( \frac{3}{2} - 2r^{2} \right) \right) dr
\]

Due to similarity of all cubic spirals, the value \( D(\alpha) \) can be computed and then derive the curve’s length \( l \) by the following equation (Proposition 8, [1]),

\[
l = \frac{d}{D(\alpha)}
\]

where \( d \) is the distance of two configurations.

2. Concept of Symmetric Configurations: For an arbitrary configuration \( q_{1}, q_{2} \) denotes its position \((x, y)\), and \( \theta \) its direction. For a configuration pair \((q_{1}, q_{2})\), the size is the distance between the two points \([q_{1}]\) and \([q_{2}]\), and the angle is the deflection angle between the two orientations \((q_{1})\) and \((q_{2})\). In [1], a symmetric mean \( q \) of any configuration pair \((q_{1}, q_{2})\) is a configuration that leads \((q_{1})\) and \((q_{2})\) are both symmetric pairs. All symmetric means of a configuration pair \((q_{1}, q_{2})\) forms a circle if \((q_{1}) \neq (q_{2})\) or a line connecting \( q_{1} \) and \( q_{2} \) if \((q_{1}) = (q_{2}) \) (Proposition 3, [1]). It is noted that the symmetric property is very important in this method because a cubic spiral can connect two symmetric configurations.

3. Original cubic spiral path planning method: The cubic spiral method can connect two given configuration \( q_{i} \) and \( q_{2} \) according to the following steps:

   I. If \( q_{i} \) and \( q_{2} \) are symmetric, connect these two configurations with a cubic spiral directly.
   II. Else, connect these two configurations with a specified symmetric mean in two different cases.
      i. Non-parallel case:
If \((q_1) \neq (q_2)\), we should define the center of the circle \(p_c\) that go through the given configurations \(q_i\) and \(q_2\) as follows:

\[
p_c = (x_c, y_c) = \left( \frac{x_1 + x_2 + c(y_2 - y_1)}{2}, \frac{y_1 + y_2 + c(x_2 - x_1)}{2} \right)
\]

where \(c = \cot((\theta_2 - \theta_1)/2)\). Thus the position of symmetric mean \([q_j]\) can be defined as:

\[
[q_j] = \left( x_j + r \cos(\beta_j + (\beta_j - \beta_1) \cdot \gamma), y_j + r \sin(\beta_j + (\beta_j - \beta_1) \cdot \gamma) \right)
\]

where \(\beta_1\) and \(\beta_2\) are represent the orientations from \(p_c\) to \(q_1\) and \(q_2\) respectively. The orientation of the symmetric mean can be defined according to the position of symmetric mean [1].

ii. Parallel case:

If \((q_1) = \emptyset\), the symmetric mean can be easily defined as:

\[
[q_1] = (x_1, y_1), \quad [q_2] = (x_2, y_2), \quad \beta = \tan^{-1}(y_2 - y_1/x_2 - x_1).
\]

For a given start configuration, a cubic spiral can be defined by the size \(d\) and deflected angle \(\alpha\), which has: its length via (5), its curvature function by (3) and its terminal configuration from equation (1). In addition, when the size of cubic spiral is negative, we can plan the backward motion of the robot according to equation (3) and (1) with \(l\) negative.

III. CUBIC SPIRAL PATH PLANNING VIA EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

A. Individual representation: candidate path

A path segment is defined by a continuous mapping \(r:[0,1] \rightarrow C\) where \(q(s) = (x(s), y(s), \Theta(s))\) denotes robot configuration with \(s\) arc length. A path is composed of a set of path segments connected via a pre-specified number of intermediate configurations. In this paper, we use cubic spiral as a path segment for a mobile robot, which is kinematically feasible. For a given start configuration, a cubic spiral can be defined by the size \(d\) and deflected angle \(\alpha\), which has: its length via (5), its curvature function by (3) and its terminal configuration from equation (1). In addition, when the size of cubic spiral is negative, we can plan the backward motion of the robot according to equation (3) and (1), i.e. \(l\) should be negative.

In this paper, a path is composed by three set of subpaths, each is composed by several cubic spiral segments: subpaths \(S, G, M\) (Fig.3). The subpath \(S\) is composed of cubic spiral segments, planned forwardly from START through a prespecified number of intermediate configurations. Similarly, the subpath \(G\) is planned backwardly from GOAL. Finally, \(S\) and \(G\) are connected by two cubic spiral segments defined via a symmetric mean [1], i.e. subpath \(M\). If we define the subpaths \(S\) and \(G\) by \(N\) cubic spiral segments, i.e.

\(N\) control points, the chromosome would consist of \(2N+1\) genes (the size and deflected angle for each cubic spiral segment in \(S\) and \(G\), and the position ratio of symmetric mean for subpath \(M\)). The composition of genes for a single chromosome \(p\) is, excluding the given START and GOAL configurations

\[
p = [d_1, \alpha_1; d_2, \alpha_2; \ldots; d_N, \alpha_N; \gamma_{SGO}], N: \text{even number}
\]

For the ease of programming, \(N\) is set as an even number so that the two subpaths \(S, G\) have equal number of segments \((N/2)\).

B. Island-based PGA (IPGA) with migration [9]

PGA based on island model is a parallelization scheme of genetic algorithms that can reduce the execution time. Since each island is run to follow a different solution trajectory, the island model may help to promote genetic diversity. The island model may have synchronous/asynchronous migration of individuals. This scheme divides the population into several communicating subpopulations each evolving via SGA in an island with a common pool serving as a migration center (Fig.3), created by cross-fertilization among the individuals of different islands. For every \(M\) generations (\(M\) is called migration frequency/interval), migration takes place. A fraction of subpopulations (called migration rate) of individuals of each island is selected based on their ranks to send to the common pool, and gathered. Then the common pool redistributes the individuals randomly onto the different islands. The size of the common pool equals the migration size (the number of individuals that migrate) of each island times the number of islands. Performance of IPGA is affected by four factors: number of migrants, migration interval and the selection and replacement strategy of individuals. In this work, the migration interval is defined as 1 for all simulations. The following paragraphs describe the detail of the PGA.
1). Fitness Definition: Rank-based Assignment & Pareto Ranking

Fast non-dominance sorting method proposed by [13] is used to rank the individuals in a population. Higher ranks will be given higher indices. The solutions of highest rank are termed as Pareto-frontier.

2). Selection

The roulette-wheel selection operator is employed. For faster convergence, elitism is used to retain some preferred individuals at each generation. One of the most popular criteria to select representative solutions from the Pareto-frontier is the min-max method [14]. The main idea of this method is to select a point within the two ends of Pareto-frontier that the maximum deviation of objectives is minimized. For a $m$ objectives problem, if there are $k$ number of solutions $p_1 \sim p_k$ within the Pareto-frontier, we can select a min-max picked solution according to

$$
\min \left[ \max \left( z_1, z_2, \ldots, z_m \right), \max \left( z_1, z_2, \ldots, z_m \right), \ldots, \max \left( z_1, z_2, \ldots, z_m \right) \right]
$$

where the deviations $z_i = f_i(p) - f_i^{\min} \sqrt{(f_i^{\max} - f_i^{\min})}$.

3). Genetic Operations: Crossover and Mutation

The crossover is implemented in this work as a linear interpolation between two chromosomes, also called arithmetic crossover [15]. The following equation shows the interpolation operation between two distinct chromosomes:

$$
\theta' = \theta_1 \cdot \gamma_1 + \theta_2 \cdot (1-\gamma_1), \theta_2' = \theta_2 \cdot \gamma_1 + \theta_1 \cdot (1-\gamma_1)
$$

where $\gamma_1$ is a randomly generated real number between 0 and 1. Please note that $\gamma_1$ is different in different genes.

Mutation is a mechanism introduced to explore new searching directions under evolution. Assuming $\theta_{\text{max}}$ and $\theta_{\text{min}}$ be the bounds of candidate solutions, the resultant descendant generated by the mutation operation will be

$$
\theta = \theta_{\text{min}} + \gamma_2 \cdot (\theta_{\text{max}} - \theta_{\text{min}})
$$

where $\theta$ stands for a chromosome in a population, $\gamma_2$ is a random number between 0 and 1.

4). Non-smoothness handling

The cubic spiral would be a spiral curve for certain values of deflected angle and size. Please refer to Fig.2 for following explanation. As this situation occurs in the evolution of paths, a non-smooth path is generated when we connect the subpaths $S$ and $G$ by a subpath $M$, since there is no guarantee that the deflected angle between last nodes at subpaths $S$ and $G$ is smaller than the specific value, as shown by dot circle of Fig. 4. It implies that a generated path is likely to be non-smooth during evolution. Thus the path problem we tackle with is a constraint-handling optimization problem. However, this type of constraint is very difficult to be reflected in the design space defined by $p$ in (6), thus we only can make a boolean discrimination for every individuals (i.e. only indicating smooth or non-smooth). This results in two categories of population at every generation, i.e. smooth and non-smooth solutions. For the proposed PGA, we generate the offspring based on two rankings: The first is a ranking for entire populations, the second is a ranking only for smooth solutions, to make the evolution to preserve the diversity of searching. In our implementation, these paths from these two rankings have the same probability to be selected, therefore smooth paths have more chances to be selected for further genetic operations.

C. Subgoals manipulation operator [8, 9]

For the mobile robot path planning problem, it is important to consider how to elaborate an infeasible path into a more acceptable path. The subgoals manipulation operator operates on the segments of infeasible paths that cross the obstacles in order to accelerate the evolution to find out the collision-free paths. The operator can locally mutate the nodes derived by those cubic spiral segments within the predefined bounds, so that this local refinement of path shape occurs in the sub-regions.

In this work, the number of manipulations for each run is defined as 10% of populations/subpopulations, and the paths with lower extrinsic cost will be selected firstly.

IV. Experimental Results

For empirical comparison of path planning performance of different intrinsic cost definition to make clear which one performs better, four performance indices are considered: The success rate represents the number of specific runs that find out at least one feasible path for 20 independent runs.
The best and worst path means the path with minimum and maximum length within those feasible paths in every single run of 20 runs and average length and standard deviation from all success runs. The initialization of candidate paths respects the following predefined range: size $d$: \([\text{diagonal distance of map} / (N^2), \text{diagonal distance of map} / (N/2)]\); deflected orientation $\alpha$: \([-\pi, \pi]\); position ratio of symmetric mean $\gamma$: \([0.2, 0.8]\), and are the same in each simulation. As Fig. 5 shown, there are three different dynamic environments used for simulation. These environments compose of four moving or rotating polygonal/wall-like obstacles in a rectangular and planar map. Please note that the velocity of the mobile robot and all of the moving obstacles have the same velocities.

| Table 1 Parameter definition and result of comparison between new and old intrinsic cost for three testing environments |
| Infor. | 3 islands | PGA |
| Population Size | 40 (per island) | |
| Evolutionary Generations | 50 | |
| Migration Interval | 1 | |
| Number of manipulations of inaccessible paths (/generation) | 10% of populations | |
| Crossover/Mutation rate | 0.85/0.1 | 0.1/0.9 | 0.4/0.3 |

| Dynamic Environment-1 with four moving polygonal obstacles |
| Success rate (20 Runs) | New Cost | Old Cost |
| Best path | 684.85 | 667.49 |
| Worst path | 769.25 | 907.43 |
| Average length $\pm$ SD of best paths | 734.83±30.53 | 760.76±81.24 |

Table 1 reveals the parameter definition of IPGA and the comparison result based on different intrinsic cost definition. For Environment-1, these two cost definition have the same success rates but different averages of length. It’s obviously that new intrinsic cost definition helps IPGA to plan a safe and shorter path in this dynamic environment. For the second environment, it consists of four moving wall-like obstacles and actually these two different cost definitions have similar performances in this simulation. In the third environment, there has a great difference in success rate since the new intrinsic cost definition dramatically helps IPGA to find out the optimum path. The difficulty of this environment for the evolutionary planner might be two rotating wall-like obstacles located in the neighbors of START and GOAL.

Fig. 6 demonstrates another dynamic environment with more moving and rotating obstacles and a safe, smooth, and shorter path is planned by the proposed IPGA. In this case, the Obstacle-7 and Obstacle-8 are static.

V. CONCLUSION

In this paper, a Pareto-based evolutionary multi-objective optimization scheme, IPGA with migration intervals 1, is employed to solve a bi-objective optimization problem for generating a smooth path to move a mobile robot safely from a start configuration to a goal configuration in completely known dynamic environments containing moving/rotating polygonal and wall-like obstacles. The cubic spiral segments are used to compose a curvature-continuous directed planar curve. A modified intrinsic cost function incorporating the penetration depth into obstacle avoidance is effective for identifying which
paths are closer to collision-free, thus raising the success rate in searching a feasible path via multi-objective evolution. Our comparative study based on simulations show that the modified intrinsic cost definition helps IPGA to plan a path more robustly.

ACKNOWLEDGMENT

This study was supported by National Science Council of R.O.C. under contract NSC 94-2212-E-001-001.

REFERENCES