# A Graph-based Ant-like Approach to Optimal Path Planning

Tingjun Lei[1], Chaomin Luo[1]*, John E. Ball[1], and Shahram Rahimi[2]
Department of Electrical and Computer Engineering[1]
Department of Computer Science and Engineering[2]
Mississippi State University
Mississippi State, USA
*Email: chaomin.luo@ece.msstate.edu

**Abstract – Motion planning of an autonomous mobile robot is involved in generating safe, optimal, short, and/or reasonable trajectories in its workspace and finally reaching its final target while avoiding collision with obstacles and escaping traps. This paper presents a new hybrid model to optimize trajectory of the global path of a mobile robot using a graph-based search algorithm associated with an ant colony optimization (ACO) method. Once a graph representing the robot workspace populated with obstacles is modelled by MAKLINK graph theory, Dijkstra algorithm is utilized to seek the *sub-optimal* collision-free robot trajectory. On the basis of the initial global sub-optimal trajectory generated by Dijkstra algorithm, the motion trajectory of the mobile robot is optimized in Cartesian space through the ACO method. Most importantly, a B-spline curve based smoothing scheme is, in a greater degree, applied to generate safer and smother trajectories with reasonable distance from obstacles. Results of simulation and comparison studies in various sorts of environments are addressed in order to demonstrate the superiority of the proposed hybrid graph-based model.**

**Index Terms –** ***robot path planning, bacteria foraging optimization, swarm intelligence, Bézier curve, trajectory optimization***

## I.INTRODUCTION

NOWADAYS, real-time motion planning of an autonomous mobile robot has constantly drawn attention in the robotics field for decades. Real-time motion planning of an autonomous robot aims to generate safe, optimal, short, and/or reasonable trajectories in its workspace and finally reach its destination while it avoids collision with obstacles and escaping traps. However, it is difficult for an autonomous robot to plan a short, optimal, collision-free trajectory in a short period due to the complexity of the robot's working environment and differences in operating conditions, as well as interference and sensor errors. Therefore, it is still an open challenge in robotics to plan safe and short trajectories efficiently and effectively for autonomous mobile robots in navigation.

A variety of robot motion planning methods has been proposed. For instance, genetic algorithm (GA) [1], particle swarm optimization (PSO) [2], ant colony optimization (ACO) [3], intelligent water drops (IWD) algorithm [4], fuzzy logic [5], neural network [6]. Bakdi *et al.* [1] applied genetic algorithm to generate a collision-free optimal path then developed an adaptive fuzzy-logic controller to keep track of an autonomous robot on the planned trajectory. Nie *et al.* [2] proposed two improved particle swarm optimization (PSO), PSO with nonlinear inertia weight and simulated annealing PSO for mobile robot path planning. PSO with nonlinear inertia weight coefficients optimized the global search ability and local search accuracy while simulated annealing PSO overcome some shortcomings of the basic PSO, such as easily trapped into the local optimum. In [3], an improved ant colony optimization called max-min ant system algorithm is applied to resolve the robot path planning problem. It provides a simple and effective way to execute the navigation of the path in an unknown environment. Salmanpour *et al.* [4] proposed a generalized intelligent water drops (IWD) algorithm to solve mobile robot navigation. It contains two different layers, which first layer finds best global path and second layer execute local search to reduce the response time. Meyer-Delius and Burgard [5] proposed a sample-based mapping method using a fuzzy *k*-means algorithm to find a map that maximizes the likelihood of the original data. Luo and Yang [6] developed a bio-inspired neural network method of intelligent robot coverage navigation model in a non-stationary environment, which is extended to unknown workspace by concurrent mapping and navigation.

Graph-based model is also one of the most used methods for robot path planning. Suh *et al.* [7] proposed an efficient graph-based informative path planer by adopting rapidly-exploring random graphs and cross entropy (CE), to search a near-optimal informative path based on cost budget constraint. Luo *et al.* [8] developed a two-level approach, which is an enhanced Voronoi Diagram associated with Vector Field Histogram algorithm based on the LIDAR sensor information for real-time robot path planning.

However, many motion planning models proposed previously have not taken the safety constraint into account

[10]. In this paper, a new hybrid model is proposed to optimize trajectory of the global path of a mobile robot using a graph-based search algorithm associated with an ant colony optimization (ACO) method. Initially, a graph representing the robot workspace populated with obstacles is modelled by MAKLINK graph theory. Dijkstra algorithm is utilized to seek the sub-optimal collision-free robot trajectory. In light of the initial global sub-optimal trajectory generated by the Dijkstra algorithm, the motion trajectory of the mobile robot is optimized in Cartesian space through an ACO method. A piecewise cubic B-spline curve based smoothing scheme is, in a greater degree, considered to plan safer and smother trajectories with reasonable distance from obstacles. The effectiveness of the proposed hybrid model has been validated by both simulations and comparison studies.

## II. Environment Model of the Mobile robot

Environmental modeling refers to the construction of a spatial environment model that helps solve the robot's path planning problem. Free space is mentioning the workspace in which the autonomous robot can move freely in the environment. Therefore, in order to facilitate the establishment of a free space model, the following assumptions need to be made:

1) The spatial environment of robot is two-dimensional.
2) The spatial environment and obstacles are static, and both defined as polygon shapes.
3) In order to ensure a safe distance between the planned path and all obstacles, the obstacle boundary expands outward as a virtual obstacle, whose size is based on the minimum cross-section of the robot as well as the minimum detection distance of the sensor.
4) The robot is considered as a particle in this section, ignoring its size.

Polygonal terrains such as obstacles in a robot workspace may be enclosed by a sequence of free MAKLINK lines. The mobile robot workspace is modeled as a graph as most graph-based motion planning mythologies. The graph of a mobile robot workspace may be established by finding the free MAKLINK line and lines of each node on every expanded virtual obstacle based on MAKLINK graph theory:

1) Each free MAKLINK line is a straight line that connected by two vertices on two different obstacles or a straight line starts from the apex of the obstacle and is perpendicular to the environment boundary
2) Each free MAKLINK line must not pass through any of the grown obstacles.

Based on the above definitions and assumptions, the MAKLINK graph theory builds the environment model in the following order to make use of its feature:

1) Find out all free MAKLINK lines as dash lines in Fig.1 according to the definition of free MAKLINK line.
2) Get the start point $S$, target point $T$ and the middle points of all $n$ free MARKLINK lines, which can be denoted by $v_1, v_2, \cdots, v_n$, respectively.
3) Connect middle points on adjacent free MAKLINK lines; Connect the starting point $S$ and middle points on adjacent free MAKLINK lines; Connect the target $T$ and middle points on adjacent free MAKLINK lines.

Each free link represents a collision-free trajectory for mobile robots as red dot lines in Fig.1.

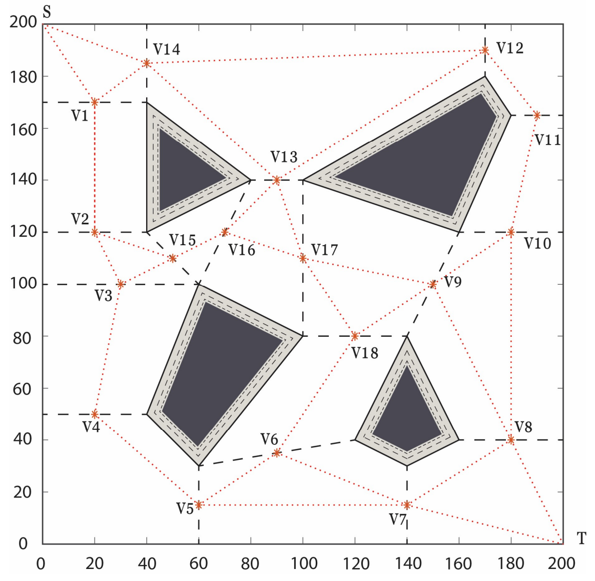


Fig.1 Illustration of workspace based on the MAKLINK graph theory

## III. Proposed Algorithm for Robot Path Planninging.

In this section, a Dijkstra's algorithm associated with an improved ACO method is described to implement robot navigation and path planning. An adjacency matrix with weights is defined in this section used to compute the shortest path, and a smooth sche

### A. Initial Path Planning

Dijkstra's algorithm is one of most used algorithms to search the shortest path on a network graph. In this section, this algorithm is used to find a sub-optimal path between the starting point $S$ and goal $T$. When using the Dijkstra algorithm, the cost function of a path needs to be calculated. The cost function can be represented as the sum of the length of all the line segments of the path. Generally, each line segment on a path will be given a weight, so the cost function can be defined as the sum of the weights of all the line segments of the path. The length of a line segment is used as its weight here.

Before using the Dijkstra algorithm, an adjacency matrix with weights must be defined, which can be used to calculate the shortest path. Each element of the matrix represents the length of the straight-line segment between two adjacent path points $v_i$ and $v_j$. Therefore, each element of the adjacency matrix can be defined as

$$adj[j] = \begin{cases} length(v_i, v_j) & if\ v_i, v_j\ are\ adjacent \\ \infty & other \end{cases} \quad (1)$$

The initial sub-optimal path, planned by Dijkstra's algorithm in the MAKLINK graph, can be represented by path points $P_0, P_1, P_2, \dots, P_d$, and $P_{d+1}$, where $P_0$ and $P_{d+1}$ represent the starting point $S$ and the goal $T$ respectively. Since these path points are the middle points of the corresponding free MAKLINK lines, they have poor performance in open areas. Therefore, the positions of the path points need to be adjusted and optimized on its free MAKLINK lines to achieve the shortest path.

Let the free MAKLINK lines crossed by the initial path are $L_i (i = 1,2, \cdots, n)$. Assuming that $P_i^1$ and $P_i^2$ are the two endpoints of line $L_i$, then the location of $P_i$ on its free MAKLINK line $P_i^1 P_i^2$ can be defined as

$$P_i(\gamma_i) = P_i^1 + (P_i^2 - P_i^1) \times \gamma_i, i = 1,2, \cdots, n \quad (2)$$

where $\gamma_i$ is the scale factor, $\gamma_i \in [0,1]$; $n$ is the number of free MAKLINK lines.

Obviously, a new robot path can be generated by a set of optimal scale factor $(\gamma_1, \gamma_2, \cdots, \gamma_n)$, thus the optimal and shortest path will be obtained, the objective function of the optimization problem can be expressed as

$$L = \sum_{i=0}^{d} length\{P_i(\gamma_i), P_{i+1}(\gamma_{i+1})\} \quad (3)$$

where length $\{P_i(\gamma_i), P_{i+1}(\gamma_{i+1})\}$ represents the straight-line distance between two adjacent path points $P_i$ and $P_{i+1}$.

### B. Improved by ACO

Ants in ACO are intelligent agents in robot path planning. They use pheromone trails, which is priori heuristic information, to navigate from one waypoint to another. The ant is initially placed in a path point which is located on the free MAKLINK lines. Ant pheromone strength $\tau_{ij}(t)$, a piece of numerical information, defined with each arc $(i, j)$ is updated in the ACO algorithm, in which $t$ is the iteration counter. Assume that at initial time $t = 0$ all the path points have the same pheromone strength $\tau_0$, that is, $\tau_{ij}(0) = \tau_0$ $(i = 1, 2, \dots, d;\ j = 0, 1, 2, \dots, g)$. $g$ represents the number of equal parts of the free MAKLINK line.

In moving process, for an agent, or a mobile robot, $k$. The transition rule of an ant moving from the previous path point online $h_{i-1}$ to the path point $j$ of the next line $h_i$ is denoted by:

$$j = \begin{cases} argmax_{u \in \delta}\{[\tau_{iu}(t)][\eta_{iu}]^{\beta}\} & if\ q \le q_0 \\ U & if\ q > q_0 \end{cases} \quad (4)$$

where $\delta$ is the set of points $\{0, 1, 2, \dots, g\}$ on $j$-th MAKLINK line available to be selected; $\tau_{iu}(t)$ is the pheromone strength of node $n_{iu}$; $\beta$ determine the relative influence of the heuristic information; $t$ is the current number of iterations; $q$ is a random variable uniformly distributed over [0,1]; $q_0$ is a parameter that tunes the exploration and the exploitation of ACO ($0 \le q_0 \le 1$); $U$ is a node which is selected next, which is calculated using the probability that the robot moves from point $i$ to point $j$ as shown in Eq.(6). $\eta_{ij}$ is the heuristic value, which is defined as follows:

$$\eta_{ij} = \frac{1.1 - |y_{ij} - \tilde{y}_{ij}|}{1.1} \quad (5)$$

where $y_{ij}$ is the $y$ coordinate of node $n_{ij}$; the values of $\tilde{y}_{ij}$ in each of the following iterations are set to the values of parameters $h_1, h_2, \dots, h_d$ which are relevant to the path points on the optimal robot trajectory generated by the ACO in the previous iteration. In the first iteration, $\tilde{y}_{ij}$ are set to 0.5 according to the initial path nodes on the free MAKLINK lines.

At each iteration phase, a probabilistic action select rule is imposed to an agent $k$. The probability of a robot $k$, currently at waypoint $i$, which traverses to waypoint $j$ at the $t$-th iteration of the algorithm, is achieved as follows in Eq. (6).

$$p_{iU}^{k}(t) = \frac{[\tau_{iU}(t)]^{\alpha} \cdot [\eta_{iU}]^{\beta}}{\sum_{u \in \delta}[\tau_{iu}(t)]^{\alpha} \cdot [\eta_{iu}]^{\beta}} \quad (6)$$

where parameters $\alpha$ and $\beta$ determine the relative influence of the pheromone trail and the heuristic information. If a parameter $\rho$ is defined as the pheromone trail evaporation, $0 < \rho < 1$ to prevent the pheromone trails from accumulating unlimitedly; it allows the ACO algorithm to neglect unreasonably bad decisions previously made.

At each iteration step, $\Delta\tau_{ij}^{k}(t)$ the amount of pheromone robot $k$ places on the arcs it has visited is dynamically updated by decreasing the pheromone strength on all arcs by a constant factor before enabling each robot to supplement pheromone on the arcs. The pheromone strength $\tau_{ij}$ is dynamically updated as Eq. (7).

$$\begin{cases} \tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij} \\ \Delta\tau_{ij} = \sum_{k=1}^{n} \Delta\tau_{ij}^{k} \\ \Delta\tau_{ij}^{k} = 1/L^{B} \end{cases} \quad (7)$$

where $L^B$ is the length of the current shortest trajectory.

## IV. Piecewise Cubic B-spline Path Smoother

The smoothing scheme aims to smooth the trajectory around the turning points in the vicinity of obstacles. The B-spline curve is one of the most efficient and most used in the smooth polyline which has closed-form expressions for position coordinates. However, in some cases, the trajectory smoothed by B-spline smoother does not quite match the original trajectory. Different from the conventional B-spline, the piecewise B-spline only smooths the path around each corner respectively. Meanwhile, in order to achieve precise path following, smooth steering command needs the curvature continuity of the trajectory. Therefore, in this paper, piecewise cubic B-splines curves are utilized to smooth the trajectory.

A B-spline curve is defined by basic functions $N_{i,k}(u)$, control points $P_i$ and degree $(k-1)$, the formula is given by

$$C(u) = \sum_{i=1}^{n+1} N_{i,k}(u)\, P_i \quad (8)$$

where $P_i = [P_{ix}, P_{iy}]$ are the $(n+1)$ control points and a knot vector $u$. $N_{i,k}(u)$ are the basis functions, which are defined recursively as follows:

$$N_{i,k}(u) = \frac{(u-x_i)}{x_{i+k-1}-x_i} N_{i,k-1}(u) + \frac{(x_{i+k}-u)}{x_{i+k}-x_{i+1}} N_{i+1,k-1}(u) \quad (9)$$

$$N_{i,k}(u) = \begin{cases} 1, & u_i \le u \le u_{i+1} \\ 0, & \text{otherwise} \end{cases} ; u \in [0,1] \quad (10)$$

where with the limitation condition of the 0/0=0 for $k=1$

In the curve smoothing procedure, in order to adapt to the parameterization difference, geometric continuity is always used to assess the smoothness of the trajectory. $G^2$ continuity means the same tangent unit and curvature vector at the intersection of two continuous segments, which can avoid discontinuities in normal acceleration and make the trajectory safer for robot to follow. Therefore, cubic B-spline is adopted because it has the lowest degree to achieve $G^2$ continuity.

In order to achieve $G^2$ continuity between the inserted B-spline curve and the remaining straight-line segment, the control points $P_i$ of B-spline curve relative to the path point $W_i$ is defined as

$$\begin{aligned} P_1 &= W_i - (1+c)d_2 u_{i-1} \\ P_2 &= W_i - d_2 u_{i-1} \\ P_3 &= W_i \\ P_4 &= W_i + d_2 u_i \\ P_5 &= W_i + (1+c)d_2 u_i \end{aligned} \quad (11)$$

where $c$ is smoothing length ratio $c = d_1/d_2$, $u_{i-1}$ is the unit vector of line $W_{i-1}W_i$ and $v_i$ is the unit vector of line $W_iW_{i+1}$. The sum of $d_1$ and $d_2$ is the smoothing length. $\varphi = \alpha/2$ is half of the corner angle. If we define knot vector [0,0,0,0,0.5,1,1,1,1], the smoothing error distance $\varepsilon$ and the maximum curvature $K_{max}$ of the smooth path can be analytically obtained as

$$\varepsilon = \frac{d_2 \sin(\varphi)}{2} \quad (12)$$

$$K_{max} = \frac{4\sin(\varphi)}{3d_2\cos^2(\varphi)} \quad (13)$$

From Eq. (12) and Eq. (13) we could define the smoothing error distance $\varepsilon$ by the existing maximum curvature $K_{max}$ given by the robots:

$$\varepsilon = \frac{2\tan^2(\varphi)}{3K_{max}} \quad (14)$$

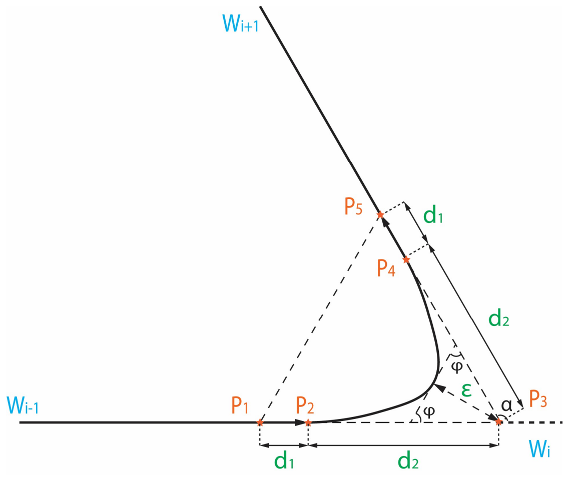


Fig. 2 Description of the $G^2$ B-spline curve

The result of basic B-spline and proposed piecewise $G^2$ B-spline smooth multiple lines with different angles, respectively is illustrated in Fig. 3. The path smoothed by the piecewise $G^2$ B-spline curve is closer to the original path than that smoothed by the basic B-spline curve. On the basis of the robot constraint, it has better performance in different degree of angles. The overall advantages of the piecewise $G^2$ B-spline can be summarized as follows.

1) The smoothed path is tangent and curvature continuity, namely that the robot has the smooth steering command, which can avoid the discontinuities of normal acceleration and the trajectory is safer for robot to follow.

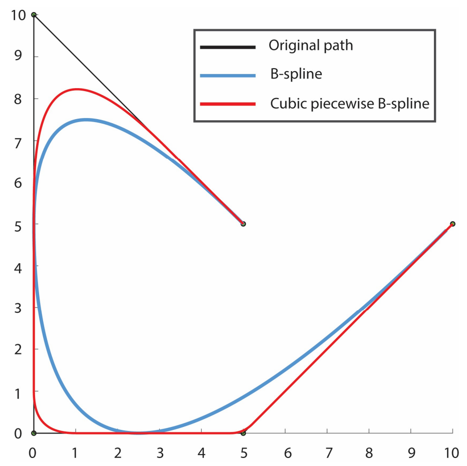


Fig.3 Illustration of the same path smoothed by the basic *B-spline* curve and piecewise $G^2$ B-spline curve separately.

2) Only the two lines at the corner will affect the smooth curve. The adjustment of any other paths does not affect the smooth curve.
3) We could easily adjust the smoothed path based on the constraint of the environment or the robot (e.g. maximum curvature).

## V. Simulation and Comparison Studies

In this section, in order to validate the model for robot navigation, simulation studies will be carried out to compare the proposed ACO model with other algorithms.

The graph from a robot workspace populated with obstacles is constructed through the MAKLINK graph theory. Dijkstra algorithm is employed to generate the suboptimal trajectory in the formed graph. The initially sub-optimized trajectory is further optimized through the ACO method. The final trajectory shown in blue line by smoothing scheme in Fig. 5 is generated by the proposed B-spline curve smoothing scheme. Finally, the MAKLINK graph method with improved ACO and the piecewise cubic B-spline path smoother are integrated to form a complete path planning scheme for robot navigation.

The proposed model is applied to a test environment with populated obstacles in comparison of the same test environment as Fig. 5 of [9] shown in Fig.4 in this context. The developed hybrid mode has been demonstrated to be effective and efficient through the simulations.

The final trajectories planned by Dijkstra, Particle Swarm Optimization (PSO) model, Immune Particle Swarm Optimization (IPSO) model and our proposed model are illustrated in Fig. 4. The proposed model compares with other models in terms of minimum path length. Path length obtained by proposed ACO method in Fig. 5 is 11.5219. The path lengths for the trajectories presented in [9] were 12.2930, 12.1670 and 11.5468 obtained by Dijkstra, PSO and IPSO methods, respectively. Our proposed method found 6.69% shorter than Dijkstra, 5.60% shorter than PSO and 0.22% shorter than IPSO, respectively. Described results are presented in Table I.

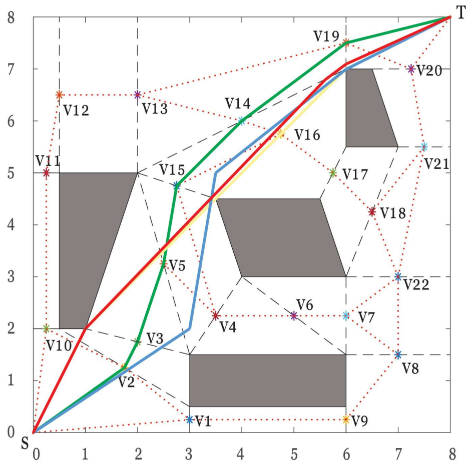


Fig.4 Illustration of robot navigation with various models. Dijkstra algorithm (green line); PSO model (blue line); IPSO model (blue line); ACO method (red line).

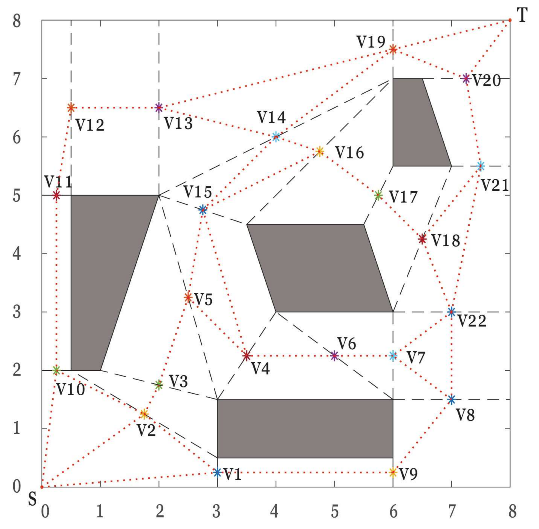
TABLE I. Comparison of Path Length

| Models | Path Length |
|---|---|
| Dijkstra | 12.2930 |
| PSO [9] | 12.1670 |
| IPSO [9] | 11.5468 |
| **Proposed ACO** | 11.5219 |

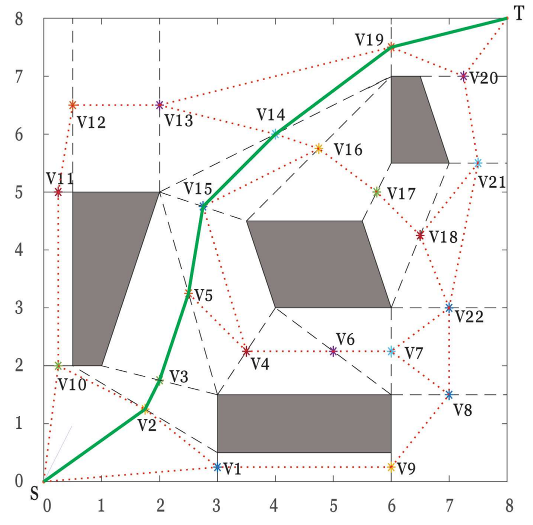
The established graph with obstacles of robot workspace through the MAKLINK graph theory is shown in Fig. 5(a). Once the workspace is built, the sub-optimal trajectory is planned by the Dijkstra algorithm shown in Fig. 5(b). The initially sub-optimized trajectory is further optimized through the ACO method, which is illustrated in Fig. 5(c). The final trajectory shown in blue line by smoothing scheme in Fig. 5(d) is generated by the proposed piece cubic B-spline curve smoothing scheme.

## VI. Conclusion

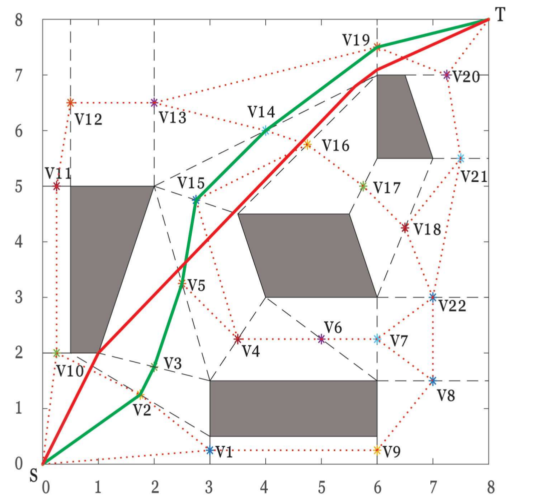
A new hybrid model to optimize trajectory of the global path of a mobile robot using a graph-based search algorithm associated with an ant colony optimization (ACO) method is proposed in this paper. Dijkstra algorithm is used to produce the sub-optimal collision-free robot trajectory, before its trajectory of the mobile robot is optimized through the ACO method. A piecewise cubic B-spline curve based smoothing scheme is utilized to plan safer and smoother trajectories with reasonable distance from obstacles. The effectiveness of the proposed model has been validated by simulations.



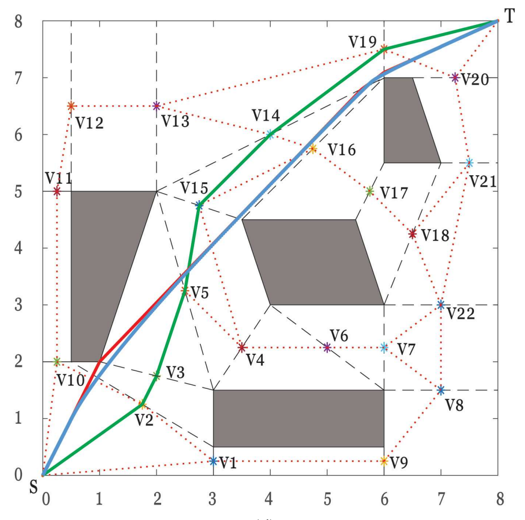
(a)



(b)



(c)



(d)

Fig. 5 Illustration of simulation based on the proposed model. (a) graph built in robot workspace by the MAKLINK graph theory; (b) sub-optimal trajectory planned by Dijkstra algorithm (green line); (c) optimal trajectory generated by ACO method (red line); (d) final trajectory shown in blue line by smoothing scheme (blue line).

## References

[1] A. Bakdi, A. Hentout, H. Boutami, A. Maoudj, O. Hachour, and B. Bouzouia, "Optimal path planning and execution for mobile robots using genetic algorithm and adaptive fuzzy-logic control," *Rob. Auton. Syst.,* vol. 89, pp. 95–109, May 2017.

[2] Z. Nie, X. Yang, S. Gao, Y. Zheng, J. Wang, and Z. Wang, "Research on autonomous moving robot path planning based on improved particle swarm optimization," in Proc. of 2016 IEEE Congress on Evolutionary Computation, 2016, pp. 2532–2536.

[3] V. D. C. Santos, F. S. Osorio, C. F. M. Toledo, F. E. B. Otero, and C. G. Johnson, "Exploratory path planning using the Max-min ant system algorithm," in *Proc. 2016 IEEE Congr. Evol. Comput.* 2016, pp. 4229–4235, 2016.

[4] S. Salmanpour, H. Omranpour, and H. Motameni, "An intelligent water drops algorithm for solving robot path planning problem," in *Proc. of 14th IEEE Int. Symp. Comput. Intell. Informatics*, pp. 333–338, 2013.

[5] D. Meyer-Delius and W. Burgard, "Maximum-likelihood sample-based maps for mobile robots," *Rob. Auton. Syst*., vol. 58, no. 2, pp. 133–139, Feb. 2010.

[6] C. Luo and S. X. Yang, "A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments," *IEEE Trans. Neural Networks*, vol. 19, no. 7, pp. 1279–1298, Jul. 2008.

[7] J. Suh, K. Cho, and S. Oh, "Efficient graph-based informative path planning using cross entropy," in Proc. of 2016 IEEE 55th Conf. Decis. Control. CDC 2016, no. Cdc, pp. 5894–5899, 2016.

[8] C. Luo, M. Krishnan, M. Paulik, B. Cui, and X. Zhang, "A novel lidar-driven two-level approach for real-time unmanned ground vehicle navigation and map building," in Proc. of Intelligent Robots and Computer Vision XXXI Algorithms and Techniques, 2014, vol. 9025, p. 902503.

[9] Y. Q. Wang and X. P. Yu, "Research for the robot path planning control strategy based on the immune particle swarm optimization algorithm," *Proc. - 2012 Int. Conf. Intell. Syst. Des. Eng. Appl.* ISDEA 2012, no. 10826098, pp. 724–727, 2012.

[10] C. Luo, S. X. Yang, X. Li, and M. Q.-H. Meng, Neural Dynamics Driven Complete Area Coverage Navigation Through Cooperation of Multiple Mobile Robots, *IEEE Trans. on Industrial Electronics*, vol. 64, no. 1, pp. 750-760, 2017.