Robotica: (2019) volume 37, pp. 1956–1970. © Cambridge University Press 2019
doi:10.1017/S0263574719000377

Optimal Path Planning Satisfying Complex Task Requirement in Uncertain Environment

Xin-Yi Yu, Zhen-Yong Fan, Lin-Lin Ou*, Feng Zhu and Yong-Kui Guo

Department of Automation, College of Information Engineering, Zhejiang University of Technology, Zhejiang Province, Hangzhou, China
E-mails: yuxy@zjut.edu.cn, 291992297@qq.com, 243409813@qq.com, 1476720217@qq.com

(Accepted March 3, 2019. First published online: April 8, 2019)

SUMMARY
Robots often need to accomplish some complex tasks such as surveillance, response and obstacle avoidance. In this paper, a dynamic search method is proposed to generate optimal robot trajectories satisfying complex task requirement in uncertain environment. The LTL-A* algorithm is presented to generate a global optimal path and the A* algorithm is provided to modify the global optimal path. The task is specified by a linear temporal logic (LTL) formula, and a weighted transition system according to the known information in uncertain environment is modeled to describe the robot motion. Subsequently, a product automaton is constructed by combining the transition system with the task requirement. Based on the product automaton, the LTL-A* algorithm is proposed to generate a global optimal path. The local path planning based on the A* algorithm is employed to deal with the environment change during the process of tracking the global optimal path for the robot. The results of the simulation and experiments show that the proposed method can not only meet the complex task requirement in uncertain environment but also improve the search efficiency.

KEYWORDS: Path planning; Complex tasks; LTL; A* algorithm; Uncertain environment.

1. Introduction

Mobile robots can work in different areas, and they have some basic properties of sensing environment, task planning and action control.[1] At present, intelligent mobile robots are widely used in the fields of factories, homes, hospitals and banks. Mobile robots often need to do some complex tasks under temporal logic constraints, such as surveillance and sequencing tasks. Therefore, the requirement for robots is getting higher and higher. In the study of mobile robot technology, the path planning is the key to complete the complex task efficiently.

The existing robot path planning methods can be divided into static path planning and dynamic path planning. Static path planning methods, including A* Algorithm,[2,3] Genetic Algorithm (GA),[4] Particle Swarm Optimization Algorithm (PSO)[5,6] and Ant Colony Optimization Algorithm (ACO),[7] can plan an optimal path according to static environment. However, present static path planning methods are offline and do not work in uncertain environment. In practical applications, the working environment of the robot will change due to some uncertain factors. Dynamic path planning methods, including artificial neural network (ANN) algorithm,[8] D* algorithm[9] and rapidly-exploring random trees algorithm (RRT),10 are online and can be applied to uncertain environment. But most of the existing dynamic path planning methods can only complete some simple tasks such as moving from

* Corresponding author. E-mail: linlinou@zjut.edu.cn

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

Point A to Point B while avoiding obstacles, and cannot meet the complex temporal task requirement in practical applications.

Recently, some new research results related to the path planning in uncertain environment have been achieved. A planning approach was proposed for the most reliable path in uncertain environments and applied to improve the reliability in robot target search tasks.[11] However, it is not suitable for the situation that the time is pressing. A Tabu path planning methodology consisting of an adaptive action selection rule and a Tabu action elimination strategy was suggested to solve the robot path planning problem in uncertain environment.[12] Stochastic level-set partial differential equations that govern the stochastic time-optimal reachability fronts and time-optimal paths were presented for vehicles navigating in uncertain, strong and dynamic flow fields.[13] A motion planning algorithm was introduced for stochastic dynamic scenarios based on a probability navigation function (PNF).14 However, few research results focus on the optimal path planning satisfying complex temporal task requirement.

The temporal logic language can be utilized to describe the complex tasks of the robot, so that some methods based on temporal logic theory were provided to solve the path planning problems of the robot under temporal logic constraints.[15–19] Some scholars have also studied the above problems in uncertain environment. For uncertain environment where a part of regions has some dynamic observations, the model checking algorithms are adopted to find a path in the transition system that is most likely to satisfy the linear temporal logic (LTL) formula, the supervising strategy for the robot motion along this path updates the execution based on the currently observed regions.[20] By bringing together tools from formal verification, graph theory and grid-based exploration, an incremental algorithm was proposed to solve the path planning problem in unknown environment.[21] The system model and the set of initial nodes for Büchi Automaton were updated through the robot-sensing environment information, and then a new optimal path is re-planned based on LTL theory.[22] The application condition of the above-mentioned method is that some environmental information is known. The above-mentioned method searches for a path in unknown environment with known environmental information. The path is just to complete the task; it is not optimal. When the known environmental information changes, the above-mentioned method is not available.

The working environment is very complicated, especially for the service robots working in the hospitals, banks or shopping malls where the transition relationship between the road nodes changes frequently due to human movement. At present, a way to satisfy the complex task requirement in uncertain environment is to re-plan an optimal path by using LTL theory according to updated environment information. It is extra time-consuming to update Büchi Automaton and product automaton every time the environment changes, especially for the frequently-changing environment. Therefore, the method has some limitations in efficiency, which leads to a narrower application field.

The A* algorithm is a heuristic search algorithm, and it is an extension of Dijkstra algorithm.[23] Comparing the completeness, time-complexity, space-complexity, and optimality of the two algorithms, the efficiency of the A* algorithm is better than the Dijkstra algorithm.[24–27]

The A* algorithm can provide better performance by using heuristics to guide its search, which reduces the computational complexity and improves the search efficiency. In this paper, a dynamic path planning method is proposed for robotic complex tasks in uncertain environment by combining the LTL theory and the A* algorithm. According to the complex task requirement, the robot task is specified by a general LTL formula. Then, the product automaton is built by combining the environment information with the LTL formula. Based on the product automaton, the LTL-A* algorithm is proposed to search for a global optimal path. When the local connectivity information of the nodes changes dynamically due to the uncertain factors of the working environment such as blockage, the local environment information is updated to construct a new transition system. Subsequently, the local search algorithm is presented to modify the global optimal path to satisfy the LTL formula of the robot in the updated environment model, such that the dynamic search method can be used in frequently-changing environment. Finally, the simulation based on V-rep demonstrates that the proposed method is feasible and effective. Furthermore, the experiment is carried out by using the SDP mini robot.

The organization of this paper is as follows. In Section 2, some preliminary knowledge about LTL is provided. A method to search optimal path under LTL constrains is proposed in Section 3 in uncertain environment. A global optimal path is generated by the LTL-A* algorithm. When the environment changes, the modified global optimal path is generated by the A* algorithm. In Section 4,

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

simulations and experiments are presented to verify the proposed method. Finally, the conclusion is given in Section 5.

2. Preliminary Knowledge

2.1. Weighted transition system

A kind of environment model called the weighted transition system is introduced. The weighted transition system is a tuple $T = (Q, q_0, \delta_T, \Pi, \zeta, w_T)$, where Q is a finite set of states, $q_0 \in Q$ in Q is an initial state, $\delta_T \to 2^Q$ is a transition relation,Π is a set of atomic propositions, $\zeta : Q \to 2^\Pi$ is a labeling function and $w_T : \mathbb{R} \to \mathbb{R}_{>0}$ is a weight function.

In this paper, the weighted transition system is finite. For any atomic proposition π, $\pi \in \zeta(q)$ is true if and only if π is satisfied in State q. The transition relation $\delta_T \to 2^Q$ is defined: if the system is in State q_1 at time t_1, it is in State q_2 at time $t_1 + w_T((q_1, q_2))$ (i.e. $q_2 \in \delta(q_1)$). For the weighted transition system T, the path r_T is denoted as $q_0q_1q_2...$, where $q_0q_1q_2...$ is an infinite sequence of states, $q_{i+1} \in \delta(q_i)$ for all $i \geq 0$. The path r_T includes a word $o = o_0o_1o_2...$, consisting of sets of atomic propositions valid at each state, where $o_i \in \zeta(q_i)$.

2.2. LTL specification and Büchi Automaton

The complex task can be specified by a general LTL formula.[28] The LTL formula over the atomic propositions is defined as follows:

$$\phi ::= \mathrm{T} | \pi | \phi_1 \vee \phi_2 | G\neg\phi_3 | X\phi_4 | \phi_5 \mathrm{U} \phi_6 \tag{1}$$

where T is predicate true in each state of a system, $\pi \in \prod$ is an atomic proposition, $\neg$ (negation) and $\vee$ (disjunction) are standard Boolean connectives,X (next) and U (until) are temporal operators, and G (always) is the derivations of operators. Additionally, other useful derivations are F (eventually) and $\to$ implication.[29]

In order to integrate environmental information with task information, *LTL2BA* toolbox 30 is used to transform LTL formula into a graph (Büchi Automaton). Büchi Automaton is a tuple $B = (S_B, S_{B0}, \sum_B, \delta_B, F_B)$, where S_B is a finite set of states, $S_{B0} \in S_B$ is a set of initial states, $\sum_B$ is an input alphabet, $\delta_B \subseteq S_B \times \sum_B \times S_B$ is a transition relation and $F_B \subseteq S_B$ is a set of accepting final states.

2.3. Büchi acceptance

A word ω is accepted by Büchi Automaton B if and only if there exists a run r_B over ω so that $\inf(r_B) \bigcap F \neq \phi$, where $\inf(r_B)$ denotes the set of states appearing infinitely often in run r_B

3. Optimal Path Planning Under LTL Constraints in Uncertain Environment

Assume that the working environment of the robot is shown in Fig. 1. In Fig. 1, the gray lines represent the roads, upper right corner is the starting point of the robot, A, B and C represent three factories, and M, N and P represent road nodes. Consider the complex task that the robot needs to fulfill is to infinitely surveillance A, B and C points of the factory in order to collect data. All the things and persons can move freely in the environment, and these uncertain factors will cause that the node connectivity changes frequently. In the paper, our goal is to dynamically generate an optimal path satisfying the above-mentioned task requirement, which adapts to the frequent changes of the environment.

The transition between the points in the environment spends some cost. The distance is chosen as the cost. The cost function is defined as

$$C(r_\mathrm{T}) = \lim_{i \to \infty} \sup(T_\pi(i+1) - T_\pi(i)) \tag{2}$$

where $T_\pi(i)$ is the *ith* satisfying time instance of proposition π. r_T is defined the satisfying run. Associated with r_T there is a sequence of time instances $T := t_0t_1t_2...$, where $t_0 = 0$ and t_i denotes the time at which State q_i is reached

$$(t_{i+1} = t_i + w((q_i, q_{i+1})))$$

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

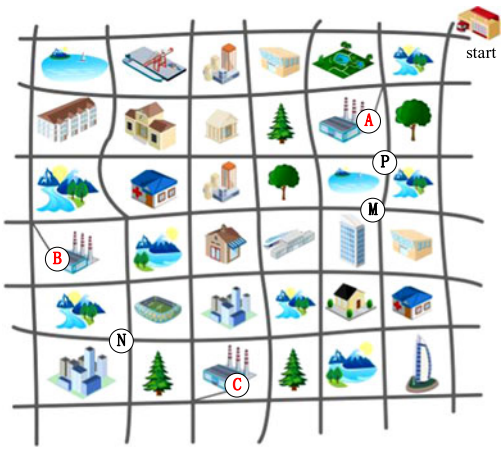


Fig. 1. The environment of robots working.

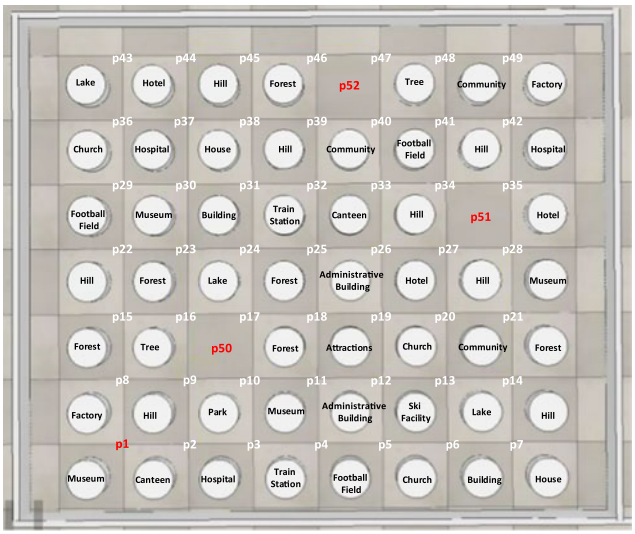


Fig. 2. The environment model.

where w is a weight function and $w((q_i, q_{i+1}))$ is defined as the distance between q_i and q_{i+1}. In this paper, we assume that the distance value between each two adjacent points is 1. For example, in Fig. 2, the distance between p_1 and p_{17} is 4. For the robot, the time and energy are proportional to the distance.[31]

3.1. Dynamic search method

According to the environment shown in Fig. 1, some special points like intersection nodes and task nodes are selected to construct a weighted transition system, as shown in Fig. 2. In this paper, the distance is equal between two points. Node p_1 is the start point of the robot, Nodes p_{50}, p_{51} and p_{52} are task points, and the other nodes are intersection ones. Additionally, an adjacency matrix $T.adj$ is used to represent the transition weights among different nodes.

The task in Fig. 1 can be mapped into Fig. 2. It is described that the robot starts from Node p_1, and patrols Nodes p_{50}, p_{51} and p_{52} in Fig. 2, which can be described by the following LTL formula:

$$\phi = GFp_{50} \wedge GFp_{51} \wedge GFp_{52} \tag{3}$$

We translate the LTL formula in form (3) into a Büchi Automaton B by LTL2BA toolbox, as shown in Fig. 3. The Büchi Automaton contains four states: S_0, S_1, S_2 and S_3, where $S_0 = init$ is initial state and S_2 is accepting(final) state. According to the weighted transition system T and a Büchi Automaton B, the product automaton is defined as $P = T \otimes B$. It is a tuple $P = (S_P, S_{P0}, \delta_P, w_P, F_P)$,

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

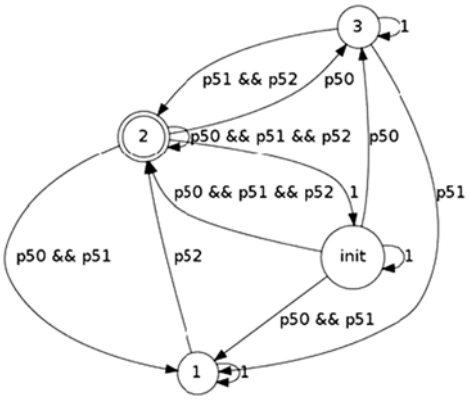


Fig. 3. A Büchi Automaton corresponding to the LTL formula in ref. (3).

where $S_P = Q \times S_B$ is the set of finite states, $S_{P0} = \{q_0\} \times S_{B0}$ is the set of initial states, $\delta_P : S_P \to 2^{S_P}$ is a transition relation which is defined as $(q_j, s_l) \in \delta_P((q_i, s_k))$ if and only if $q_j \in \delta_T(q_i)$, $s_l \in \delta_B(s_k, \zeta(q_i))$, $\omega_P : S_P \times S_P \to R^+$ is the weight function: $w_p((q_i, s_k), (q_j, s_l)) = w_T(q_i, q_j), \forall (q_j, s_l) \in \delta_P((q_i, s_k))$, $F_P = Q \times F_B$ is the set of final accepting states.

The environment information and task information can be known by the product automaton which is a weight graph. The path is obtained in the weight graph satisfies the working environment and LTL formula constrains. The product automaton can be seen as a Büchi Automaton because the input alphabet is trivial. Thus, when $\inf(r_P) \bigcap F_P \neq 0$, a run r_P in product automaton P is accepting.

Each run $r_P = p_0 p_1 p_2 \ldots$ can be replace $T_P = t_0 t_1 t_2 \ldots$ in the transition system, where $t_0 = 0$, and t_i denotes the time at which the *ith* vertex in the run is reached $t_{i+1} = t_i + w_P(p_i, p_{i+1})$. A sequence $T_{P,\pi}$ can be extracted from this time sequence. $T_{P,\pi}$ contain time instance t_i, where $p_i \in S_{P,\pi}$. For example, $T_{P,\pi}$ represent the optimizing proposition π in T.

The product automaton can also be viewed as a weighted graph. The states of the product automaton are vertices of the graph. The transitions of the product automaton are the edges. Thus, runs of the product automaton can be called paths. An infinite path consists of finite paths. Each accepting run of the product automaton is equivalent to a run of the transition system. They are both satisfying the LTL formula.

The traditional methods based on LTL theory use Dijkstra algorithm to find an optimal path. The Dijkstra algorithm is a BFS (breadth-first-search) algorithm. Thus, when the working environment of the robot has more nodes, the Dijkstra algorithm searches slowly. The A* algorithm is more effective than the Dijkstra algorithm in finding the shortest path.[24] The A* algorithm is a heuristic algorithm. It utilizes heuristics to guide its search direction so that the search space is reduced and the efficiency of path searching is greatly improved. Therefore, the optimal accepting path of the product automaton can be found quickly based on the A* algorithm.

The evaluation function of the A* algorithm is the following structure:

$$f(n) = g(n) + h(n) \tag{4}$$

where n is the last node on the graph, $g(n)$ is the cost of the path from the start Node S_{P0} to node n and $h(n)$ is a heuristic function that estimates the cost of the cheapest path from Node n to the final states F_P. The time complexity of the A* algorithm depends on the heuristic function which never over-estimates the actual cost from the current node to the nearest goal node. In this paper, the following Manhattan distance is selected as the heuristic function:

$$h(n) = \sqrt{abs(n.x - goal.x)^2 + abs(n.y - goal.y)^2} \tag{5}$$

where $n.x$ and $n.y$ represent the abscissa values and the ordinate values of Node n, respectively, $goal.x$ and $goal.y$ represent the abscissa values and the ordinate values of goal node, respectively, and $abs(\bullet)$ calculates the absolute value.

The heuristic function $h(n)$ can be used to control A*'s behavior. $d(n)$ is the actual distance from Node n to the final node.

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

(I) At one extreme, if $h(n)$ is 0, then only $g(n)$ play a role, and A* turns into Dijkstra's Algorithm, which is guaranteed to find a shortest path.
(II) If $h(n)$ is always lower than (or equal to) $d(n)$, then A* is guaranteed to find a shortest path. The lower $h(n)$ is, the more node A* expands, making it slower.
(III) If $h(n)$ is exactly equal to $d(n)$, then A* will only follow the best path and never expand anything else, making it very fast. Although you cannot make this happen in all cases, you can make it exact in some special cases. It's nice to know that given perfect information, A* will behave perfectly.
(IV) If $h(n)$ is sometimes greater than $d(n)$, then A* is not guaranteed to find a shortest path, but it can run faster.
(V) At the other extreme, if $h(n)$ is very high relative to $g(n)$, then only $h(n)$ plays a role, and A* turns into Greedy Best-First-Search.

Lemma 1. *$d(n)$ is the actual distance from Node n to the final node. When $h(n) <= d(n)$, the A* algorithm can get optimal solution.*[32]

Because the robot can only travel on the grid line, so $d(n)$ is $abs(n.x - goal.x) + abs(n.y - goal.y)$. Thus, $d(n)$ is higher than $h(n)$. The method of this paper can guarantee to get an optimal solution.

The A* algorithm needs *open* set that contains the discovered nodes and *closed* set that contains the covered nodes. Meanwhile, if a node can be arrived from some nodes, *parent* node is constructed to eventually contain the previous step. At each iteration, the A* algorithm selects the node whose $f(n)$ is minimal as the last node. If the last node is the goal node, the A* algorithm returns the optimal path. Otherwise, the node is removed from *open* set, and is added to *closed* set. Set S'_{P0} to be the start node, and F'_P to be the goal node, the search process of the A* algorithm is shown in Algorithm 1, where $S'_{P0} \in S_{P0}$, and $F'_P \in F_P$.

Algorithm 1: A* path search $A_Search(T, S'_{P_0}, F'_P)$

Input: product automaton P, the start node S'_{P_0}, the goal node F'_P
Output: the optimal path r'_P

```
open = S'_P0, closed = ∅, FATHER = ∅;
while open is not empty do
    B = {n|MIN(f(n)), n ∈ open};
    if B = F'_P then
        return optimal path r'_P;
        end while
    end
    remove B from open;
    add B to closed;
    for each neighbor B' of B point do
        if B' ∈ closed then
            ignore B';
        end
        cost(B') = g(B) + cost(B, B');
        if B' ∉ open and B' ∉ closed then
            g(B') = cost(B'), B' ∈ open,
              parent(B') = B ;
        end
        f(B') = g(B') + h(B');
    end
end
if open = ∅ then
    return failure
end
```

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

After a product automaton is constructed by combining the transition system with the Büchi Automaton, a global optimal path can be obtained in the product automaton. The path r_P is with the prefix–suffix structure, the prefix path r_{prefix} is the path from an initial state to an accepting state, and the suffix path r_{suffix} is the path from an accepting state back to itself. Algorithm 2 is proposed to find a global optimal path r_P satisfying the task requirement and work environment. Prefix path and suffix path can be generated based on Algorithm 1. It is worth noting that the prefix is executed only once by the robot while the suffix path is repeated infinitely.

The path r_P not only contains the path information in the environment but also the state information of the Büchi Automaton. Therefore, it is necessary to map r_P to the weighted transition system so as to get the path r_T in the environment.

Lemma 2 (Product Run Projection).[22] *For any accepted run $r_P = (p_0, s_0), (p_1, s_1), (p_2, s_2)\ldots$ of the product automaton P, the sequence $r_T = p_0 p_1 p_2\ldots$ is a run of T, the trace of which should satisfy LTL formula ϕ automatically. Furthermore, the cost given in (2) corresponding to r_P and r_T are equal.*

Algorithm 2: Global optimal path search $LTL_AStar(P)$

Input: product automaton P
Output: the optimal path r_P
for *each initial state* $S^i_{P0} \in S_{P0}$ *and each accepting state* $F^j_P \in F_P$ **do**
  $r^{ij}_{prefix} = A_Search(P, S^i_{P0}, F^j_P)$;
  $r^j_{suffix} = A_Search(P, F^j_P, F^j_P)$;
  $r^{ij}_P = (r^{ij}_{prefix}, r^j_{suffix})$;
end
find the r^{ij}_P that minimize the sum of weight between states;
return $r_P = (r^{ij}_{prefix}, (r^j_{suffix})^\omega)$

According to Lemma 2, the accepted run r_P by Algorithm 2 can be projected to an optimal path r_T in T satisfying ϕ in (4). When the connectivity of the path node changes, we update the corresponding local environment information to obtain the new weighted transition system T' according to the changing node $r_T(i)$. In the new weighted transition system, the start point of the path is set to be $r_T(i)$, and the target point is set to be $r_T(i+2)$. Thus, the local optimal path satisfying the task requirement can be obtained based on the A * algorithm. The local optimal path r is presented (6) as follows:

$$r = A_Search\left(T', r_T(i), r_T(i+2)\right) \tag{6}$$

According to the obtained local optimal path, the original global optimal path is modified into:

$$r_T = r_T(0) r_T(1) \ldots r_T(i-1) r r_T(i+3) \ldots r_T(n) \tag{7}$$

The robot continues to finish the task along the modified path r_T in (7) and always detects whether the environment changes again. If the environment changes, the local search algorithm is used to correct r_T. By fixing the path constantly, the robot can cope with the environment changes and finish the task successfully.

The local environment for the robot is shown Fig. 4 Assume that the task specification is the movement from Node p_1 to Node p_{17} in Fig. 4. An optimal global path satisfying the task specification is generated by Algorithm 2, which is given by

$$r_T = p_1\, p_8\, p_{15}\, p_{16}\, p_{17}$$

when arriving at Node p_{15}, the robot finds that Node p_{16} is blocked, and cannot reach Node p_{17} through Node p_{16}. At this time, Node p_{16} is labeled as an obstacle in the weighted transition system

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

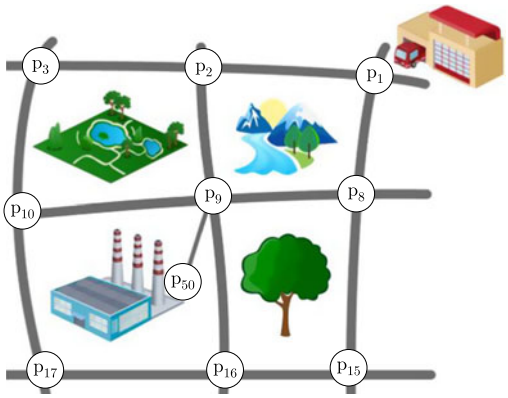


Fig. 4. Local environment.

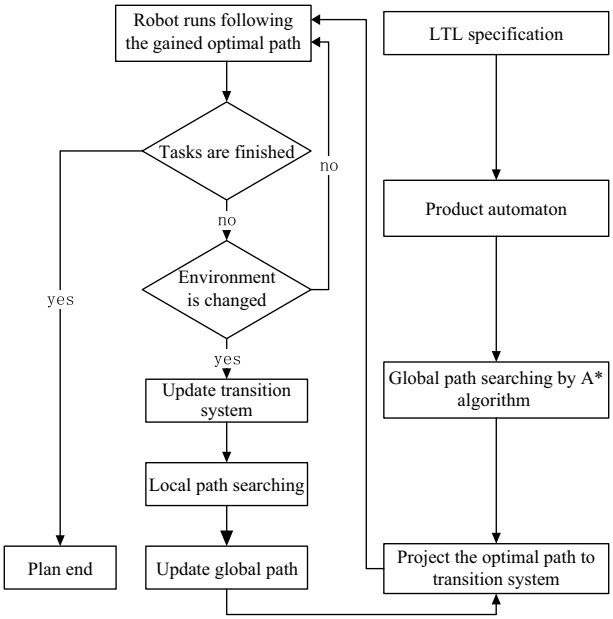


Fig. 5. Dynamic path planning framework.

$T(T \to T^{'})$, and Algorithm 1 is directly employed to gain the local optimal path:

$$r_T = p_{15}\, p_8\, p_9\, p_{10}\, p_{17}$$

Then, the robot continues to perform the tasks along the updated path. Once the local environment changes again, the local path search method continues to fix the path r_T. Eventually, the robot can accomplish its task efficiently. At present, most of the LTL-based path planning methods in uncertain environment need to reconstruct the product automaton for the changing environment information, and the global optimal path is searched repeatedly based on the product automaton, which reduces the search efficiency and cannot adapt well to the uncertain environment with frequent changes. The proposed method only updates the weighted transition system according to the local environment and performs the local optimal path search for the changing environment. Since it does not need to update the Büchi Automaton and the product automaton, the method fewer nodes than traditional LTL-Dijkstra method. Thus, the search efficiency is improved for the proposed method.

3.2. The optimal path planning algorithm in uncertain environment

In practical applications, the robot often needs to complete some complex temporal tasks and the connectivity of the road node often changes in the working environment. The optimal path planning method satisfying the complex task requirement is proposed in uncertain environment, and the dynamic path planning framework is shown in Fig. 5. Firstly, the working environment is transformed

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

into a weighted transition system. The task requirement of the robot is specified by the LTL formula. A product automaton is built by combining the environment information with the LTL formula. Then, the global optimal path is generated based on the A * algorithm in the product automaton, and the global optimal path in the transition system is obtained according to Lemma 2. Lastly, when the robot performs the task along the global optimal path, the local environment information is updated to obtain the new transition system. Based on the new transition system, the local search algorithm is employed to obtain the local optimal path. The detailed optimal process is shown in Algorithm 3. According to the cost given in (2), it can generate the optimal path.

Algorithm 3: Dynamic search in uncertain environments $DynPathSearch()$

Input: a weighted transition system T, LTL ϕ.
Output: an optimal path r_T
1 Given a LTL specification ϕ, convert ϕ to a Büchi Automaton B by $LTL2BA$ toolbox;
2 Computer the product automaton $P = T \otimes B$;
3 Computer the global optimal path r_P in P based on Algorithm 2;
4 Project r_P to r_T in T, according to Lemma 1;
5 The robot performs tasks according to r_T and detects whether the environment changes;
6 **if** *the environment changes* **then**
7 Determine the current node $r_T(i)$ of the robot, update the weighted transition system T to get T';
8 Search the local optimal path r: $r = A_Search(T', r_T(i), r_T(i+2))$;
9 **return** the optimal path r_T to the changing environment: $r_T = r_T(0)r_T(1)...r_T(i-1)\ r\ r_T(i+3)...r_T(n)$;
10 **end**
11 Repeat Steps 5, 6, 7, 8, 9, and 10 until the robot completes the task.

4. Simulation and Experimental

In this section, two examples are presented to verify the efficiency and applicability of the proposed algorithm in the uncertain environment. V-rep software is used for simulation and SDP mini robot is used for experiment.

Before the experiment begins, the working environment for the robot is built. The houses, mountain, obstacles and others are represented by wooden boxes. SDP mini robot gets environmental information by sensor. The task is given in the console. The optimal path is obtained according to the proposed method. The console sends the optimal path to the robot. The robot moves according to the optimal path obtained. When the environment changes, the robot transmits the information obtained to the console. The console performs a local search to obtain the modified optimal path. The console sends the modified optimal path to the robot. The robot moves according to the modified optimal path obtained.

The experiments are fully reported in the video which can be obtained from https://youtu.be/NLwDaXApv0A. Parts 1–4 show that the robot can complete the tasks by the proposed method.

Example 1. Consider the working environment in Fig. 1. The task of the robot is that the robot starts from the starting point, and infinitely patrols Points A, B and C. When some road nodes cannot be passed due to some uncertain factors, the robot can adjust the route to continue the task. Corresponding to the environment model in Fig. 2, the task is that the robot starts from Node p_1, and patrols Nodes p_{50}, p_{51} and p_{52}. The robot cannot be affected by environment change during the task execution.

The above-mentioned task can be interpreted as the LTL formula in (1). The global optimal path is generated from Steps 1 to 4 in Algorithm 3, and the optimal path is presented in Fig. 6. According to the cost given in (2), the cost $C(r_T)$ is 26. Figure 7 shows the experiment of the SDP mini robot and its practical running trajectory.

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

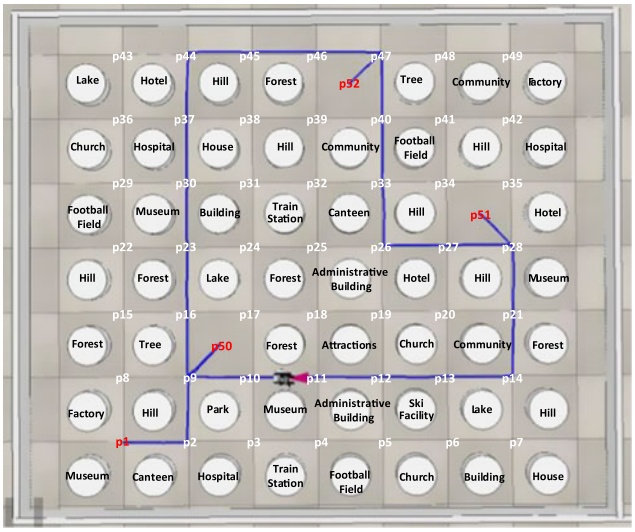


Fig. 6. Global optimal path for Example 1 (simulation).

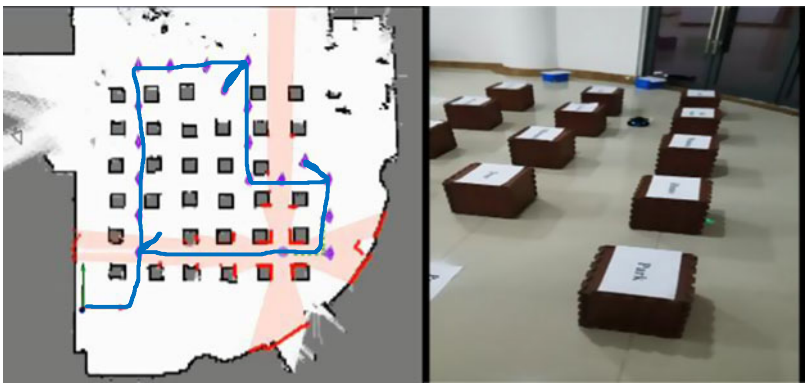


Fig. 7. Global optimal path for Example 1 (experiment).

When arriving at Node p_{13}, the robot finds that Nodes p_{14} and p_{21} are blocked due to human movement, and cannot reach Node p_{28} through Nodes p_{14} and p_{21}. According to Steps 5–10 in Algorithm 3, the local optimal path r from Node p_{13} to Node p_{28} is generated, that is, $r = p_{13}\, p_{20}\, p_{27}\, p_{28}$. The cost $C(r_T)$ is 26. One local changing environment and the modified running trajectory of the robot in such the environment are presented in Fig. 8. Comparing Figs. 7 with 9, the robot can complete the tasks when environment changes by the proposed method.

In order to verify that the path generated by the presented method can satisfy the complex tasks in uncertain environments, a more complex temporal task is provided in Example 2.

Example 2. Based on the working environment in Fig. 1, the weighted transition system is shown in Fig. 2. The following task is considered. The robot starts from Node $4p_{51}$, gathers data repeatedly at Nodes p_{50} and p_{51} with avoiding Node p_{16}, and reaches Node p_{23} to upload data after arriving at Node p_{50}. Similarly, the robot must reach Node p_{41} to upload data after arriving at Node p_{51}.

According to the task requirement, the corresponding LTL formula is as follows:

$$\begin{aligned}\phi = {} & GFp_{50} \wedge G(Fp_{50} \rightarrow X(\neg Fp_{50}\, U\, Fp_{23})) \wedge \\ & GFp_{51} \wedge G(Fp_{51} \rightarrow X(\neg Fp_{51}\, U\, Fp_{41})) \wedge G\neg p_{16}\end{aligned} \tag{8}$$

According to Algorithm 3, the global optimal path r_T in T is shown in Fig. 10. The experiment environment and the practical trajectory of the robot are shown in Fig. 11. The cost $C(r_T)$ is 26. When the robot moves from Nodes p_{23} to p_{28}, Node p_{24} cannot be passed temporarily. Algorithm 1 is used to search for the local optimal path from Nodes p_{23} to p_{25}. When the robot comes back to

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

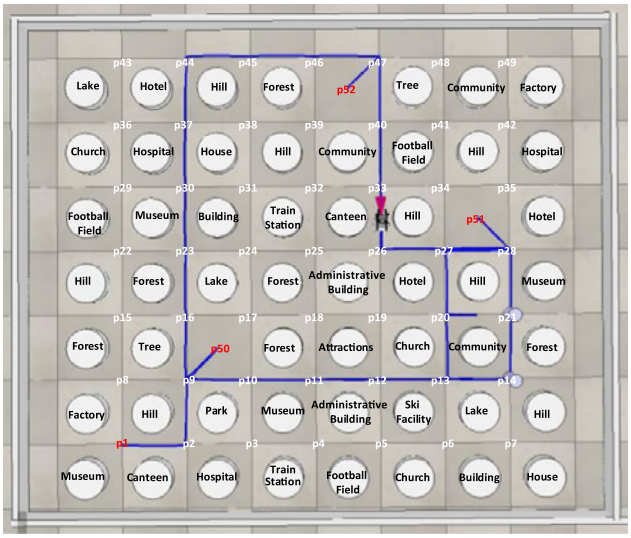


Fig. 8. Final optimal path in the case of environment change for Example 1 (simulation).



Fig. 9. Final optimal path in the case of environment change for Example 1 (experiment).

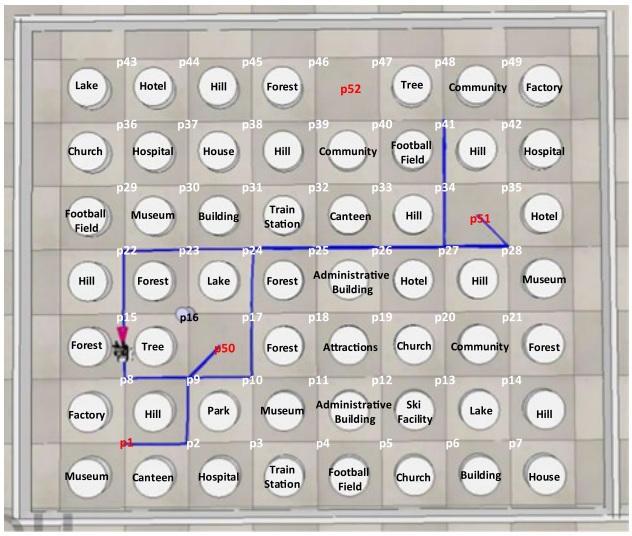


Fig. 10. Global optimal path for Example 2 (simulation).

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

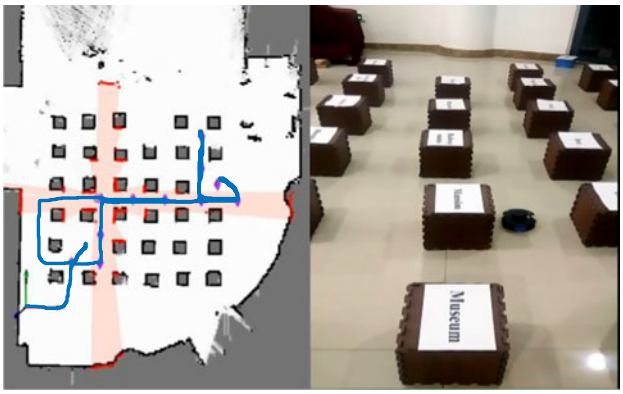


Fig. 11. Global optimal path for Example 2 (experiment).

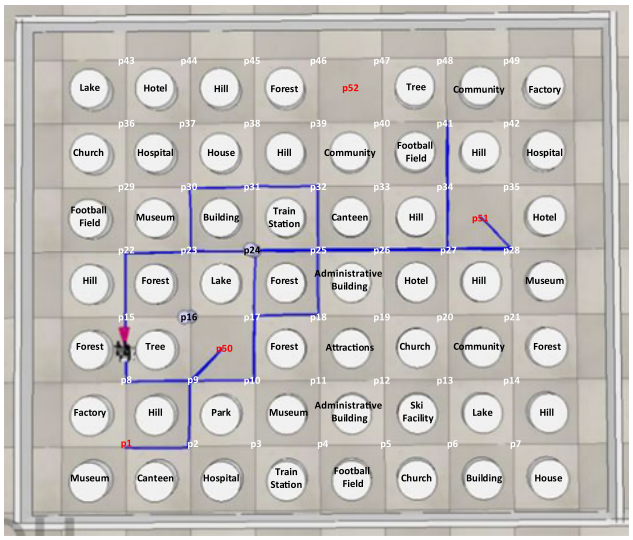


Fig. 12. Final optimal path in the case of environment change (simulation).

Node p_9 from Node p_{24}, Node p_{24} is still unable to be passed. Algorithm 3.1 is employed again to search for the local optimal path from Nodes p_{25} to p_{17}. The final optimal path is shown in Fig. 12. The changing environment and the corresponding experiment result are presented in Fig. 11. The cost $C(r_T)$ is 28.

Comparing Figs. 11 with 13, it is known that the robot can also complete the complex temporal tasks described by (8) when the environment changes. Furthermore, the practical trajectories of the robot are optimal in both Figs. 11 and 13 for complex task in uncertain environment. Thus, the proposed method is effective.

From the above experimental results, it can be found that the proposed method can dynamically generate the optimal path satisfying the complex task requirement in uncertain environment. In addition, in order to improve the efficiency of the algorithm, the global optimal path is generated based on the A* algorithm instead of Dijkstra algorithm in the product automaton.

The difference between the proposed method and the method of ref. [31] is given as follows.

(I) The proposed method can be applied to uncertain environments. Although the method given in ref. [31] can search an optimal path which satisfies the given specification, the resultant path may not work anymore when the environment changes. For the existing method, it cannot be applied to uncertain environment directly. In order to deal with the environment changes, the weighted transition system related to the whole environment model has to be reconstructed and

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

Table I. The comparison of the run times for different algorithms.

	Run time (ms)	
Task node number	**Proposed method**	**The method of ref. [31]**
2	15.6	24.5
3	19.8	32.3
4	24.7	37.4
5	27.3	60.1
6	30.7	86.8
7	33.1	92.5
8	39.4	112.8
9	40.9	137.1
10	54.5	190.4

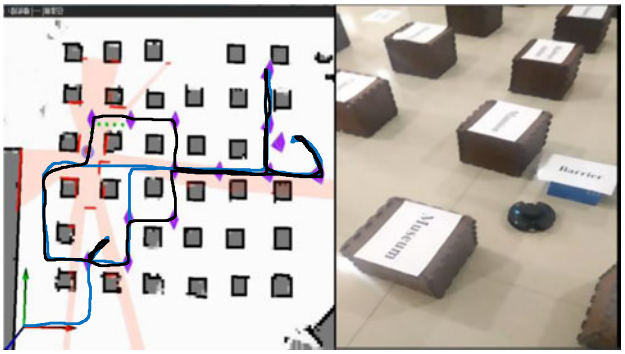


Fig. 13. The final optimal path in the case of environment change (experiment).

the new path needs to be found based on the global weighted transition system. If the environment changes frequently, the computation complexity will be very large and cannot update the path in time according to the environment change.

(II) The A* algorithm is used to find the shortest path instead of the Dijkstra algorithm as in ref. [31]. When the number of the nodes for searching is very large, the method of ref. [31] is very time-consuming. The comparison results are shown in Table I. From Table I, it is seen that the method of this paper is more effective than the method of ref. [31].

In order to further prove that the proposed method is more effective than the method of ref. [31], several sets of experiments in MATLAB are used for comparison. Given the weighted transition system in Fig. 2, the proposed method is compared with the LTL-Dijkstra traditional one in the search time of optimal path. The patrolling tasks with different number of task nodes are provided to computer the run time of the proposed method and the method of ref. [31]. The comparison result is shown in Table I. Both the methods can generate the path satisfying the task requirement for the same task, but the run time of the proposed method increases more slowly with the increase of task complexity than the run time of the method of ref. [31]. The comparison result is carried out in MATLAB (R2012a) on a desktop computer with Inter(R) Core (TM) i7-3770K CPU @3.50 GHz and 4 GB of RAM.

5. Conclusion

In this paper, we presented a method to efficiently search the optimal path satisfying complex surveillance task requirement in uncertain environment. The complex task is specified by the LTL formula, and a global optimal path is generated based on the A* algorithm instead of Dijkstra algorithm in the product automaton which is built by combining the environment information with the task requirement. When the environment changes, the proposed method avoids the repetitive model establishment of the whole working environment and global path planning. Only a local optimal path satisfying the changed environment needs to be found so as to correct the global path. Hence, the

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

proposed method improves the search efficiency and provides the basis for the robot application in changing environment. The results of the simulation and experiment demonstrate the validity of the proposed method. Future research could be extended to multi-robot systems with more complex environment.

Acknowledgements
This work was partly supported by National Key R&D Program of China (2018YFB1308000) and the National Science Foundation of China [grant number 61473183].

References

1. B. G. Xu, Y. X. Yi and M. J. Zhou, "Current status and prospects of intelligent mobile robot technology," *Rob. Tech. Appl.* **2**(2), 29–34 (2007).
2. I. Chabini and S. Lan, "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks," *Trans. Intell. Transp. Syst.* **3**(1), 60–74 (2002).
3. Y. Xu and R. Liu, "Path planning for mobile articulated robots based on the improved A* algorithm," *Int. J. Adv. Rob. Syst.* **14**(4), 1–10 (2017).
4. J. Tu and S. X. Yang, "Genetic algorithm based path planning for a mobile robot," *International Conference on Robotics and Automation*, Taipei, Taiwan (2003), pp. 1221–1226.
5. Y. Shi, "Particle swarm optimization: Developments, applications and resources," *Int. Conf. Evol. Comput*. **1**, 81–86 (2001).
6. J. Kennedy, *Particle Swarm Optimization. Encyclopedia of Machine Learning* (Springer, Perth, Australia, 2011), pp. 760–766.
7. X. Fan, X. Luo, S. Yi, S. Yang and H. Zhang, "Optimal path planning for mobile robots based on intensified ant colony optimization algorithm," *International Conference on Robotics, Intelligent Systems and Signal*, Changsha, China (2003) pp. 131–136.
8. W. L. Xu, S. K. Tso and Y. H. Fung, "Sensor-based reactive navigation of a mobile robot through local target switching," *International Conference on Advanced Robotics*, Monterey, CA, USA (1997), pp. 361–366.
9. A. Stentz, "The focussed D* algorithm for real-time replanning," *International Joint Conference on Artificial Intelligence*, Montreal, Quebec, Canada (1995) pp. 1652–1659.
10. A. H. Qureshi and Y. Ayaz, "Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments," *Rob. Autonom. Syst.* **68**, 1–11 (2015).
11. B. T. Zhang, Y. Liu, Q. Lu and J. Wang, "A path planning strategy for searching the most reliable path in uncertain environments," *Int. J. Adv. Rob. Syst.* **13**(5), 1–9 (2016).
12. C. Y. Wei and F. S. Ni, "Tabu temporal difference learning for robot path planning in uncertain environments," *Lect. Notes Comput. Sci.* 123–134 (Springer, Cham, 2018).
13. N. S. Deepak, Q. T. Wei and F. J. Pierre, "Stochastic time-optimal path-planning in uncertain, strong, and dynamic flows," *Comput. Methods Appl. Mech. Eng.* **333**, 218–237 (2018).
14. H. Shlomi, S. Shraga and S. Nir, "Applying probability navigation function in dynamic uncertain environments," *Rob. Autonom. Syst.* **87**, 237–246 (2017).
15. F. Bacchus and F. Kabanza, "Planning for temporally extended goals," *Ann. Math. Artif. Intell.* **22**, 5–27 (1998).
16. Y. Xinyi, C. Hao, G. Yongkui, C. Cheng, O. Linlin and Y. Li, "Path planing of warehouse robots based on linear temporal logic," *Chin Hig. Technol. Lett.* **26**(1), 16–23 (2016).
17. Y. T. Xiao, O. Lin-Lin and Y. Li, "Optimal patrolling path planning via linear temporal logic," *Acta Autom. Sin.* **40**(10), 2126–2133 (2014).
18. M. Guo, K. H. Johansson and D. V. Dimarogonas, "Motion and action planning under LTL specifications using navigation functions and action description language," *International Conference on Intelligent Robots and Systems*, Tokyo, Japan (2013) pp. 240–245.
19. K. J. Leahy, D. Aksaray and C. Belta, "Informative path planning under temporal logic constraints with performance guarantees," *American Control Conference*, Seattle, WA, USA (2017) pp. 1859–1865.
20. M. Kloetzer and C. Mahulea, "LTL planning in dynamic environments," *Int. Fed. Accountants* **45**, 294–300 (2012).
21. A. M. Ayala, S. B. Andersson and C. Belta, "Temporal logic motion planning in unknown environments," *International Conference on Intelligent Robots and Systems*, Tokyo, Japan (2013) pp. 5279–5284.
22. N. Xu, J. Li, Y. Niu and L. Shen, "An LTL-based motion and action dynamic planning method for autonomous robot," *Int. Fed. Accountants. Ifac PapersOnLine.* **49**, 91–96 (2016).
23. H. Wang, Y. Yu and Q. Yuan, "Application of Dijkstra algorithm in robot path-planning," *Second International Conference on Mechanic Automation and Control Engineering*, Inner Mongolia, China (2011) pp. 1067–1069.
24. C. Sheng-qun and D. Lin-fei, "Applied analysis of Dijkstra algorithm and A-star algorithm in the intelligent guiding," *J. Chongqing Univ. Sci. Technol. (Natural Sciences Edition)* (2010), pp. 1673–1980.
25. G. Nannicini, D. Delling, L. Liberti and D. Schultes, "Bidirectional A ∗ Search for time-dependent fast paths," *J. Am. Chem. Soc.* **136**(24), 8626–8641 (2014).
26. A. R. Soltani, H. Tawfik, J. Y. Goulermas and T. Fernando, "Path planning in construction sites: Performance evaluation of the Dijkstra, A ∗, and GA search algorithms," *Adv. Eng. Inf.* **16**, 291–303 (2002).

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press

27. J. Yao, C. Lin, X. Xie and A. J. Wang, "Path planning for virtual human motion using improved A* star algorithm," *Seventh International Conference on Information Technology*, Las Vegas, NV, USA (2010), pp. 53–58.
28. S. L. Smith, J. Tůmová, C. Belta and D. Rus, "Optimal path planning under temporal logic constraints," *International Conference on Intelligent Robots and Systems*, Taipei, Taiwan (2010) pp. 3288–3293.
29. E. A. Emerson, "Temporal and modal logic," *Handb. Theor. Comput. Sci.* 995–1072 (Elsevier, Amsterdam, Netherlands, 1990).
30. P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," *International Conference on Computer Aided Verification*, Paris, France (2001) pp. 53–65.
31. S. L. Smith, J. Tůmová, C. Belta and D. Rus, "Optimal path planning for surveillance with temporal-logic constraints," *Int. J. Rob. Res.* **30**(14), 1695–1708 (2011).
32. P. E. Hart, N. J. Nilsson and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst. Sci. Cybern.* **4**, 100–107 (1968).

https://doi.org/10.1017/S0263574719000377 Published online by Cambridge University Press