SHORT PAPER

Path Exploration in Unknown Environments Using Fokker-Planck Equation on Graph

Haoyan Zhai[1] · Magnus Egerstedt[2] · Haomin Zhou[1]

Abstract

This paper introduces a graph-based, potential-guided method for path planning problems in unknown environments, where obstacles are unknown until the robots are in close proximity to the obstacle locations. Inspired by the Fokker-Planck equation and the intermittent diffusion process, the proposed method generates a tree connecting the initial and target configurations, and then finds a path on it using the available environmental information. The tree and path are updated iteratively when newly encountered obstacle information becomes available. The resulting method is a deterministic procedure proven to be complete, i.e., it is guaranteed to find a feasible path, when one exists, in a finite number of iterations. The method is scalable to high-dimensional problems. In addition, our method does not search the entire domain for the path, instead, the algorithm only explores a sub-region that can be described by the evolution of the Fokker-Planck equation on graph with a changing of diffusion coefficient intermittently. We demonstrate the performance of our algorithm via several numerical examples with different environments and dimensions, including high-dimensional cases.

1 Introduction

This paper considers path exploration for a robot, or a group of robots moving simultaneously, in an unknown environment. In other words, a set of robots in given initial configurations are tasked with finding a feasible path to the target configuration, while avoiding collisions with obstacles. We consider scenarios where the number of robots is fixed and where obstacles are detected when they are within detection range to one of the robots in the group. We assume that the system employs a broadcast strategy in the sense that the obstacle information, once available,

is shared among the group immediately. Compared to the path planning problem in known environments, there are several significant challenges when the problem is posed in unknown environments. First of all, due to lacking environmental information, path re-exploration while moving becomes inevitable when an established path is blocked by newly detected obstacles. This is an intrinsic nature of the problem that must be faced by any algorithm dealing with unknown environments. Secondly, there are livelock concerns, i.e., the robots may end up moving back and forth, or in loops, and never reach the target even when there exist feasible paths. Thirdly, there may be narrow pathways between obstacles, which poses hurdles to identify them in the search process. More importantly, for problems with unknown environments, optimality becomes less meaningful or can only be addressed in the currently known environment. Hence, one may have to accept feasible solutions after all. In addition, for multi-robot systems, the configuration space may be high-dimensional, especially when there are a large number of robots. For example, a 2-dimensional d-robot-planning problem yields a $2d$ configuration space, since the position of each robot contributes two variables and they must be considered simultaneously. In this case, grid-like discretizations often lead to intractable computations. Working with graphs is a viable option to reduce the

✉ Haoyan Zhai
haoyanzhai@gmail.com

Magnus Egerstedt
magnus@uci.edu

Haomin Zhou
hmzhou@math.gatech.edu

1 School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332, USA

2 Samueli School of Engineering, University of California, Irvine, Irvine, CA 92697, USA

computation burden. However, the cost can still be high if the graph has to span everywhere in the high-dimensional space.

There exists an extensive literature on path planning and exploration. For example, the well-known Probabilistic Road Map (PRM) method generates a random graph that does not intersect with obstacles and then finds a path over the graph to connect the initial and target configurations [2, 27, 44, 49]. PRM guarantees a connection between the initial and target configurations when the graph is dense enough in the configuration space. Many additional PRM have been reported in the past decades, see [4–6, 20] for details. The Rapidly-exploring Random Tree (RRT) is another influential method that utilizes randomness and tree structures for path exploration [31]. The algorithm has been adopted to path finding in unknown environments [50, 56] and has some recent improvements such as RRT* as reported in [26, 42]. The Artificial Potential Field (APF) assigns the robot and obstacles positive charges while the target configuration a negative one [28]. This drives the robot towards the target while avoiding collisions with the obstacles. However, a known limitation of APF is the creation of unnecessary local minima due to the presence of obstacles, which may fail the algorithm. In recent years, there are reported improvements of APF, see [1, 9, 39, 46] and the references therein.

The family of Bug Algorithms, starting from the original Bug0, Bug1 and Bug2 [35, 36] to the later developments, such as TangentBug [24], DistBug [23] and many other variants, adopt two basic modes as their design principle: motion-to-goal mode and boundary-following mode. They are powerful tools, with theoretical convergence guarantees, especially suitable for path planning and exploration in unknown environments in 2-dimensional working space. Some recent survey and performance comparison studies can be found in [38, 41].

In addition, widely known graph based methods, like A^* [17], D^* [47], Focused D^* [48] and D^* lite [29] can be used for path planning and exploration in both known and unknown environments [45]. When applied to the problems in unknown environments, they often require to cover the entire region by discrete lattice grids, on which the algorithms are performed to find paths. In literature, there are other types of algorithms such as genetic algorithm [32, 54], evolutionary programming [12], fuzzy logic [19, 55], neural network [37], network simplex method [18], method of evolving junctions [34], fast marching tree [21], and a few hybrid approaches that combine different methods [16, 51], and also many more swarm strategies for multi-agent systems in recent years [13, 14, 53].

In this paper, we present a potential guided, tree-based path exploring method inspired by the evolution of Fokker-Planck Equation (FPE), which has been studied intensively in optimal transport theory [52]. Optimal transport theory is a branch of mathematics studying how to transport one probability distribution to another with the optimal cost. There are different ways to formulate the theory such as using linear programming [25], or partial differential equations (PDEs) [7, 8] etc. A remarkable result stemmed from optimal transport theory is that the FPE can be viewed as the gradient flow of a free energy, consisting of potential energy and entropy, in the probability space equipped with the so-called Wasserstein metric (optimal transport distance) [22, 43]. Using the optimal transport theory, one can show that FPE can escape the traps of any local minima in a potential field and reach the Gibbs distribution which concentrates on the global minimizer. Incorporating this property and advantages of several existing algorithms, we design a novel method for path exploration in unknown environments. Our goal is providing an alternative algorithm that can work efficiently, especially for problems with high dimensional configuration spaces, such as multi-agent systems. When designing our algorithm, we introduce a potential field, defined by the distance to the target configurations in this paper. The unique global minimum of the potential field is at the target configurations. Unlike APF, the obstacles do not contribute to the potential field, instead, they define the infeasible regions. We generate a graph that has a tree structure originated from the initial configuration, growing in a deterministic manner guided by the flow direction of FPE towards the target configuration. Our algorithm has the following features:

1. The algorithm is a tree based deterministic procedure with guaranteed convergence, meaning that the algorithm terminates in a finite number of steps, and returns a feasible path if there exists one. Thus, the algorithm is complete. We highlight that the convergence of our algorithm is deterministic, in contrast to the probabilistic, asymptotic convergence results shared by many methods using randomness. If the algorithm does not find a feasible path, it indicates that there is no feasible path within the current resolution.

2. The number of new nodes added to the tree grows linearly with respect to the dimension of the configuration space in the algorithm. Together with the dimension reduction techniques proposed to rapidly escape local traps, the algorithm can efficiently handle high dimensional problems.

3. The algorithm only explores a limited region defined by the solution of the FPE, even when the obstacles are not known a priori. However, the algorithm does not require to solve the FPE on graph numerically.

4. The exploration region, which the algorithm doesn't need to know in advance, is constructed by an intermittent

diffusion process [11] that alternates the diffusion coefficient in FPE between 0 and a positive number.

We want to note that optimal transport theory has been considered in several recent studies for path planning and exploration. For example, swarming robots are modeled by a distribution, and their optimal transport map is calculated by linear programming in [3]. A different swarming model, based the Kantorovick-Rubinstein relaxation formulation from optimal transport, is proposed to solve the robot deployment by using PDEs [30]. Our method is different in that we directly use the evolution of FPE on graph to guide the path construction.

In the next section, we present the details of the algorithm with the finite step stopping property. In Section 3, we show some numerical examples to illustrate the performance in both low and high dimensional configuration spaces. Section 4 gives strategies for dimension reduction near local minima to further lower the computational cost. In Section 5 the relationship between the algorithm and optimal transport theory is discussed. The convergence proof is given in Section 6. We end the paper with a brief conclusion in the last section.

2 Algorithm

Let the configuration space Ω be a bounded connected domain in $\mathbb{R}^n$. We assume that the robots can alter configurations freely, with a linear dynamics $\dot{x} = u$ where $x \in \Omega$ is the configuration variable and $u \in \mathbb{R}^n$ is the control variable, in Ω as long as the change does not violate the required constraints. There are two types of constraints we consider in this paper. One is the constraints known in advance, for example, two robots can't be too close or too far away from each other in the multi-agent system. We denote those constraints by $\phi = (\phi_1, \phi_2 \cdots, \phi_{k_1})$, where ϕ_i is real valued function, and a configuration $x \in \Omega$ does not satisfy the constraints when $\phi_i(x) < 0$ for some $i \in \{1, \cdots, k_1\}$. The other type of constraints is given by unknown environments, such as unknown obstacles. We represent them by $\psi = (\psi_1, \psi_2 \cdots, \psi_{k_2})$, and $\psi_i(x) < 0$ for some $i \in \{1, \cdots, k_2\}$ means the constraints being violated. We assume that $\psi(x)$ can only be detected if robots are close enough to the obstacles. This implies that the knowledge of $\psi(x)$ must be updated dynamically while the robots are in motion. For the simplicity in discussion, we also assume that both ϕ and ψ are continuous.

To illustrate the setups, we give a single robot example in Fig. 1. The configuration space is a square, all the gray bars are the obstacles that the robot cannot collide with. The light gray indicates undetected obstacles. Like in the second picture in Fig. 1, if the robot moves horizontally

but not too far away from its initial configuration, there is no detected obstacle. As the robot moves, more obstacles are recognized. Our goal is finding a path from the initial configuration x_s (red diamond in Fig. 1) to the target configuration x_t (red circle). More precisely, we want to find a feasible path

$$\gamma(t) : [0, T] \rightarrow \Omega,$$

satisfying $\phi(\gamma(t)) \geq 0$ and $\psi(\gamma(t)) \geq 0$ for all $t \in [0, T]$, such as the red path in Fig. 1, while $\psi(x)$ is updated with newly detected obstacles as $\gamma(t)$ changes.

To describe the dynamical change of the unknown constraints while moving along a path, we mark a configuration x as part of the detected obstacles if $(\psi_i(x) < 0)$ and x is within distance R to the current configuration of the robot. The distance function is induced as $dist(x, y) = \|x - y\|$ given $x, y \in \Omega$, where $\|\cdot\|$ can be any norm defined on Ω. To be precise, we define

$$\widetilde{\psi}(x, t, \gamma) = \begin{cases} \psi(x) & \text{if } \|x - \gamma(\tau)\| \leq R \text{ for some } \tau \leq t \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

as the detected part of the environment along the path.

For the convenience of discussion, we assume there exists at least a feasible path connecting the initial and target configurations, and this feasible path is contained in a tubular obstacle-free region $\mathcal{T}$ with radius L as shown by the shadow part in the first picture in Fig. 1. This assumption is a technique requirement that is used for the proof of the convergence and can be rewritten as the following equation

$$\sup_{\gamma \in \Gamma} \inf_{t \in [0,T]} \sup_{r \geq 0} \{r : B(\gamma(t), r) \cap \mathcal{O} = \emptyset\} = L > 0, \quad (2)$$

where $\mathcal{O} = \{x \in \Omega : \psi_i(x) < 0 \text{ or } \phi_j(x) < 0 \text{ for some } i, j\}$ is an open set. $B(x, r) = \{y \in \Omega : \|x - y\| < r\}$ is also open. We denote $S(x, r) = \{y \in \Omega : \|x - y\| = r\}$ as the boundary of $B(x, r)$, and $\partial\mathcal{O}$ as the boundaries of $\mathcal{O}$ separating the constrained regions from the feasible regions.

Let us define the set of all possible paths from x_s to x_t in the full time interval $[0, T]$ as
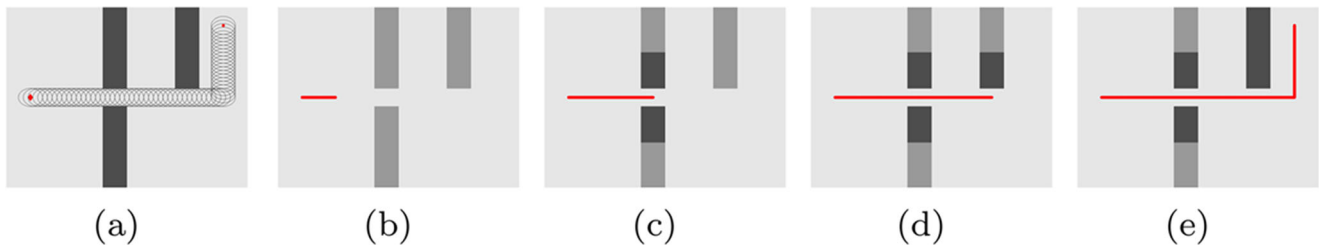
$$\Gamma = \{\gamma : \gamma(0) = x_s, \gamma(t) = x_t, \forall t \geq T_0,$$
$$\text{for some } T_0 \leq T, \gamma \cap \mathcal{O} = \emptyset\}. \quad (3)$$

and denote the currently known constraint set as

$$\mathcal{O}_c = \left\{x \in \Omega : \widetilde{\psi}_i(x, T_0, \gamma) < 0 \text{ or } \phi_j(x) < 0 \text{ for some } i, j\right\}.$$

Then our dynamical path exploration algorithm is given in Algorithm 1:

In the remaining part of this section, we discuss, assisted with examples, the three major steps in detail.

Fig. 1 The obstacles are the light and dark gray regions (light as undetected and dark as detected). In (a), the red diamond and circle are the start and target configurations. The shaded tubular region $\mathcal{T}$ is obstacle-free. (b)–(e) are snap shots of a robot moving along a path

---

Algorithm 1: Path exploration in unknown environment.

Data: initial configuration x_s, target configuration x_t, initially known constraints $\mathcal{O}_0$

1 Current configuration $x_c = x_s$
2 Current known constraints $\mathcal{O}_c = \mathcal{O}_0$
3 **while** $x_c \neq x_t$ **do**
4 | **Graph Generating:** Generate a connected graph G containing x_c, x_t with all edges and vertices not in $\mathcal{O}_c$
5 | **Path Finding:** Find a (shortest) path γ on G from x_c to x_t
6 | **Environment Updating:** Moving along γ while updating $\mathcal{O}_c$, if γ is blocked by $\mathcal{O}_c$, stop at x near the block point, otherwise let $x_c = x_t$
7 **end**

2.1 Graph Generating

The first step is to generate a graph $G = (V, E)$, where V is the vertex set and E is the edge set, connecting the current configuration x_c and the target x_t with currently known environment. The vertices are configurations in Ω while the edge (u, v) linking $u, v \in V$ is the straight line segment between u and v. Meanwhile, we would like to create the graph satisfying two properties: 1) the graph does not violate any known constraints; 2) the graph cannot contain too many vertices due to the computation complexity concern in the high dimensional cases. To achieve these goals, we introduce a convex potential function $p(x)$, admitting a unique global minimizer x_t, to help choosing the vertices. We select an n-dimensional orthonormal basis N (here n is the dimension of Ω) to determine the directions used to add new vertices to V. For simplicity, we take $p(x) = \|x - x_t\|$, the distance to the target, as the potential, and the standard coordinate axes $N = \{e_j\}_{j=1}^{n}$ as the orthonormal basis in this paper.

At the first generating step, we have $V = \{x_c\}$ and $E = \emptyset$. In each step afterward, a vertex $v \in V$ with the lowest potential is chosen. We pick $2n$ new points $\{v_i\}_{i=1}^{2n}$ along the orthonormal basis N originated at v, with distance l to v, and use them as the candidates to expand V (first figure in Fig. 2a). Before adding those points into the vertex set, we first delete all candidates that violate the currently known constraints ($\phi_k(v_i) < 0$ or $\widetilde{\psi}_j(v_i, \gamma, T_0) < 0$ for some $k \in \{1, \cdots, k_1\}$ and $j \in \{1, \cdots, k_2\}$, γ is the previous trajectory of the robots). For example, the robot shown in Fig. 2a stops at the red diamond position and generates four points around it. Among them, the point in the obstacle is removed. Next, we delete vertices whose edges violate the constraints as shown in Fig. 2b. In this case, there exists a point $x \in (v_i, v)$ such that x is not in the feasible region. In addition, to avoid repeating vertices, we remove those already included in V from the candidate list, as shown in Fig. 2c. After these deleting steps, we add all remaining candidates, and their associated edges, to V and E respectively. This process is repeated until the target x_t is within a small neighborhood of a vertex in V. For example, the final graph after several iterations is plotted in Fig. 2d. The graph generating procedure can be written in Algorithm 2. We call x the ancestor of y if x is picked to generate nodes, and y is added to the vertex set as newly generated node.

Remark 1 The choice of the generating radius l can be arbitrary, although L and l must satisfy an inequality to have the convergence guarantee theoretically (as is shown in Section 2.4). Larger l leads to fewer vertices in V while smaller l giving a finer search in Ω. For simplicity, we treat those obstacles with distance less than l to be a single obstacle by ignoring the gaps among them in our theoretical analysis. In practice, as shown in our experiments, the graph generation can still create nodes passing through the gap between obstacles with distance less than L or even l. Similarly, the orthonormal basis N can be replaced by other vector sets as long as the number of the vectors depends on the dimension n linearly and the conclusion of Lemma 3 in Section 6 is achievable by the vector sets. In this case, the computational complexity and completeness of the algorithms remain unchanged.
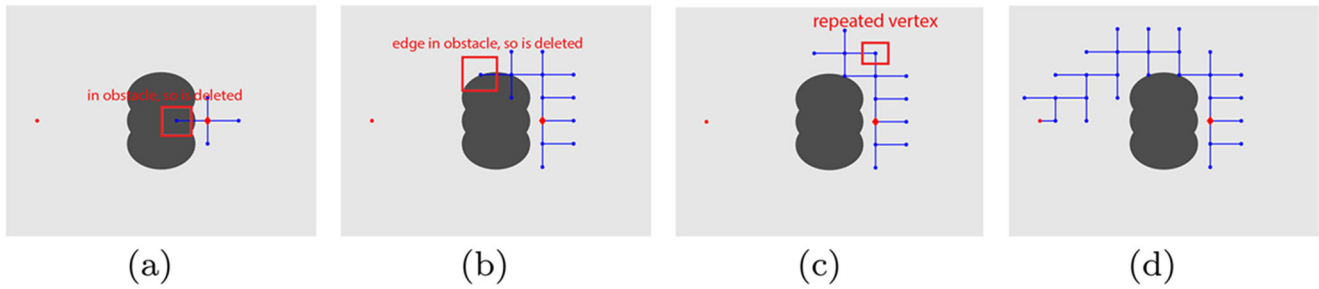
Fig. 2 graph generating steps

2.2 Path Finding

After generating the graph $G = (V, E, p)$, the next step is to find a feasible path moving from the current configuration to the target using only vertices and edges on the graph. Our goal is to minimize the total travel distance. The graph generated by Algorithm 2 has the following property:

Proposition 1 *There exists a unique path from the current configuration x_c to the target x_f over the generated graph G. If the path is denoted by $\{x_i\}_{i=1}^q \subset V$ with $x_c = x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_q = x_f$, in which x_i is the ancestor of x_{i+1}.*

This proposition is guaranteed by the tree structure of the graph. To make the proof complete, we show the detailed proof in the Section 6.3.

By our graph generation algorithm, if there is an edge between two nodes, one of them must be the unique ancestor of the other. This suggests a simple strategy to identify the path: from the target configuration, we simply back trace the ancestor of each node in the path until reaching starting configuration x_c.

Other algorithms can be applied to find the path as well. For example, we can define the distance of the edge e_{ij} linking vertices v_i, v_j as $k_{ij} = len((i, j)) = \|v_i - v_j\|$. Then the well-known Dijkstra method, or its improvements, can be used to obtain the path with computational complexity $O(|E| + |V| \log |V|)$ where $|V|$ is the number of vertices and $|E|$ is the number of edges [15].

Another way is to assign each edge distance 1 which is equivalently to introduce the modified adjacency matrix $K = (k_{ij})$ on the graph G, where

$$k_{ij} = \begin{cases} 1 & \text{if } (i, j) \in E \\ \infty & \text{otherwise.} \end{cases}$$

Then the Breadth First Search (BFS) can be used to find the path with the complexity $O(|E| + |V|)$ [40, 57], which is faster than Dijkstra. Other graph-based path planning and exploration algorithms, such as A^*, D^* or D^* lite, can be used too.

It is worth mentioning that if we assume the path has Λ nodes, the suggested back-tracing approach is of complexity $O(\Lambda)$. While the generated graph has at least $O(n\Lambda)$ nodes. Obviously the complexities of BFS and Dijkstra methods are higher than our back-tracing strategy.

2.3 Environment Updating

While robots move along a path γ_i in the configuration space, the knowledge of constraints is updated at the same time by Eq. 1. If a point on the path intersects the boundary of a newly detected constrained region, we stop the motion at a point before arriving the intersection.

To be more precise, let us denote the environment update at each time step as

$$\mathcal{O}_c^t = \mathcal{O}_c \cup \{x \in \Omega : \widetilde{\psi}_j(x, t, \gamma_i) \\ < 0 \text{ for some } j \in \{1, \cdots, k_2\}\}.$$

If the path is found activating constraints while moving at time T_s, i.e. $\gamma_i \cap \mathcal{O}_c^{T_s} \neq \emptyset$, we define

$$T_b = \inf\{t : \gamma_i(t) \in \mathcal{O}_c^{T_s}\}$$

as the first intersection time. Then $\gamma(T_b)$ must be on the boundary of $\mathcal{O}_c^{T_s}$, i.e. $\gamma(T_b) \in \partial \mathcal{O}_c^{T_s}$. When this happens, we can always pick a stopping time $T_i \leq T_b$ such that the distance from $\gamma(T_i)$ to the nearest obstacle is smaller than the detection radius R. Then we update $\mathcal{O}_c = \mathcal{O}_c^{T_i}$, assign the initial configuration as $x_c = \gamma(T_i)$ and go back to the graph generating step. Each time a new path γ_i is produced when the current path is blocked. We collect all paths produced in Algorithm 1 as $\{\gamma_i\}_{i=1}^m$, and their stopping time set as $\{T_i\}_{i=1}^m$. From our choices of stopping time, we can require that there exists a positive constant q satisfying $q < R$, and for all $\epsilon > q$, $B(\gamma_i(T_i), \epsilon)$ has non-empty intersection with $\mathcal{O}_c$ for every $i = 1, \cdots, m$. Such selected stopping time set satisfies the following property

$$\sup_i \inf_\epsilon \left\{ \epsilon : B(\gamma_i(T_i), \epsilon) \cap \mathcal{O}_c^{T_i} \neq \emptyset \right\} = q < R, \tag{4}$$

in which the detectable region at configuration x, using Eq. 1, is defined as a closed set by $\bar{B}(x, R) = \{y \in \Omega : d(x, y) \leq R\}$. We emphasize that q can be selected uniformly. For example, we can simply let robots stop at a position that has a distance of $R/2$ to the obstacles each time

---

Algorithm 2: Graph generation.

Data: The starting configuration x_c, target configuration x_t, the potential function p, currently known environment $\mathcal{O}_c$, graph generating radius l and a set of orthonormal basis N

Result: G=(V,E,p)

1 $V = \{x_c\}$, $Q = V$, $E = \emptyset$
2 **while** $x_t \notin V$ **do**
3     $point\_add = False$
4   **while** $not \; point\_add$ **do**
5         $v = \arg\min_{x \in V} p(x)$
6   **if** $p(v) < +\infty$ **then**
7             $K = \{q : q = v \pm l \times y, y \in N, (v,q) \cap \mathcal{O}_c = \emptyset, q \notin \mathcal{O}_c\}$
8             $K = K \backslash V$
9             $V = V \cup K$
10        $E = E \cup \{(v,q) : q \in K\}$
11   **if** $K \neq \emptyset$ **then**
12            $point\_add = True$
13            $p(v) = +\infty$
14   **end**
15   **for** $q \in K$ **do**
16   **if** $\|q - x_t\| \le L \; and \; (q, x_t) \cap \mathcal{O}_c = \emptyset$ **then**
17               $V = V \cup \{x_t\}$
18               $E = E \cup \{(q, x_t)\}$
19   **end**
20   **end**
21   **else**
22   **return** $G = \emptyset$
23   **end**
24   **end**
25 **end**
26 **return** $G = (V, E, p)$

---

when the path is blocked. In this setup, $q = R/2 < R$. In general, we can select different stop positions. The finite-step convergence property, presented in the next Section, is guaranteed as long as Eq. 4 is satisfied.

2.4 Convergence and Complexity

The proposed algorithms terminate in finite steps with guaranteed convergence, which is stated in the following main theorems.

Theorem 1 *Assuming that* Eq. 2 *is true and* $l < (2L)/\sqrt{n}$, *where* n *is the dimension of* Ω, *the graph generation algorithm (Algorithm 2) stops in finite steps. That is, the loop in the algorithm terminates in finite iterations, the*

generated graph $G = (V, E, K, p)$ is connected and has a finite number of vertices $|V| < \infty$. Furthermore, $x_s, x_t \in V$ if $\Gamma \neq \emptyset$.

Theorem 2 *Let* $\{\gamma_i\}_{i=1}^m$ *be the paths produced by Algorithm 1 with* $\{T_i\}_{i=1}^m$ *being the stopping time set. If the assumptions in Theorem 1 and* Eq. 4 *hold, then* $m < \infty$.

Theorem 1 shows that, given the currently known environment, the graph generating procedure stops in finite steps. Theorem 2 tells that our algorithm breaks the loop in Algorithm 1 in finite steps. The two theorems together ensure that Algorithm 1 is convergent in finite steps and guarantees a feasible path with the condition (2). Therefore, the algorithm is complete. We leave the proofs of both theorems in Section 6.

Furthermore, if the configuration space Ω is of dimension n, there are at most $2n$ new points generated at each step in Algorithm 2, hence the growth rate for the size of the graph V is $O(n)$ at each iteration. The complexity of the Updating Environment step relies on the techniques used to detect the environment, so we do not consider it here. Overall, the proposed algorithms are scalable to high dimensional problems, because the growth of the graph is controlled linearly with respect to the dimension n and it stops in finite steps. Moreover, the algorithm merely searches part of the region. This searching region of the algorithm is theoretically bounded by the evolution trajectory of the Fokker-Planck equation, while the algorithm itself does not involve in any computation of the PDE. This property will be discussed further in Section 5. Before that, we present several numerical examples to illustrate the completeness of the algorithm in the next section.

3 Numerical Examples

We set the working space to be $[0, 1] \times [0, 1]$ in all examples and denote the graph generating radius as l. In this section, we show various low and high dimensional experiments with different environments. In all examples, the start configurations are always marked as red diamonds while the targets are the red circles.

3.1 Low Dimensional Cases

The first example is one robot moving in an unknown environment (Fig. 3). The configurations are the physical locations of the robot, so this is a two-dimensional problem. We take $l = 0.03$ in our graph generating algorithm. Initially, the robot is at the bottom right corner. It only has the knowledge of a few nearby obstacles at the beginning, while other obstacles are not known. Hence, the graph

Fig. 3 The graph produced in the one robot case with generating radius $l = 0.03$ and light (dark) gray the undetected (detected) obstacles. The graph expands greedily towards the target. If obstacles are on the greedy direction, it searches around the obstacles and generates new nodes with potential as low as possible. (a)–(d) are graphs that cross undetected obstacles so the robot stops while moving on those graphs. With enough environment knowledge, (e) is a graph containing a true feasible path from the current initial configuration and the target
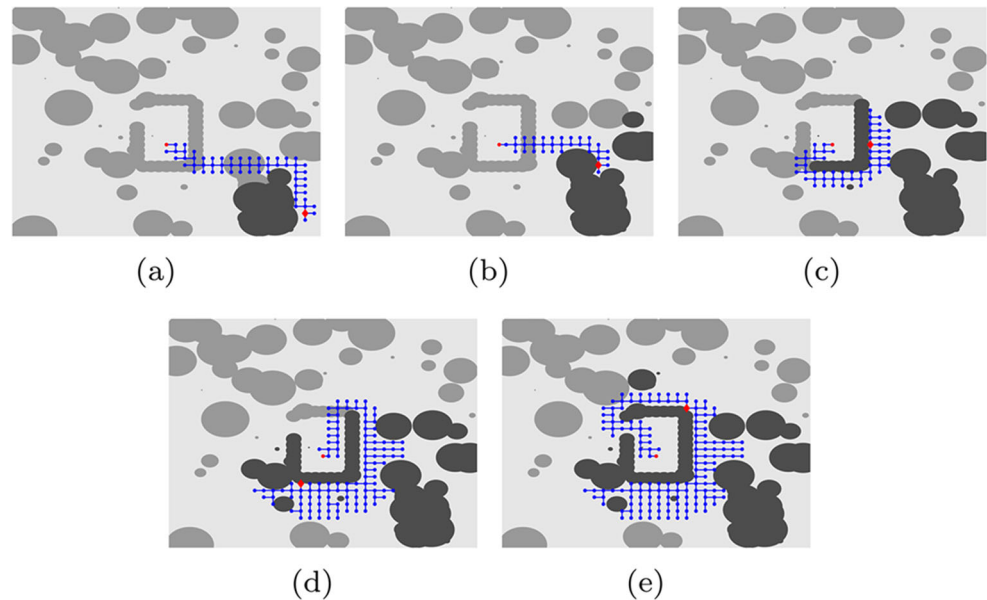

expands towards the target greedily until reaching the destination as shown in Fig. 3a. The first path is shown in Fig. 4a. However, while moving, the robot detects that the path is blocked. It stops before reaching the obstacle boundary and starts a new round of graph generating, path finding and environment updating steps. During the process, the robot generates several graphs (Fig. 3b,c, and d) and updates the environment while moving along the corresponding paths as shown in Fig. 4b,c,d, all of which fail to reach the destination. In the end, it generates a graph (Fig. 3e) and finds a path (Fig. 4e) to the target. The complete path from initial to target is plotted in Fig. 4f.

3.2 High Dimensional Cases

In the next few examples, we calculate the paths for several multiple-agent systems. In addition to the constraints imposed by the obstacles, we also enforce that the robots cannot be too close or too far away from each other. The constraints are introduced to avoid collision—they cannot be too close, or model communication restrictions if this is needed. For example, robots in the group have short communication range and they cannot be too far away in order to stay connected. In our examples, we set that any two robots must keep their distance between 0.03 and 0.13 while in

Fig. 4 The paths calculated based on the results in Fig. 3. (a)–(d) are middle steps that the robot stops because of the newly detected obstacles while moving and (e) is the path on which the robot get to the target. (f) gives the complete path of the robot
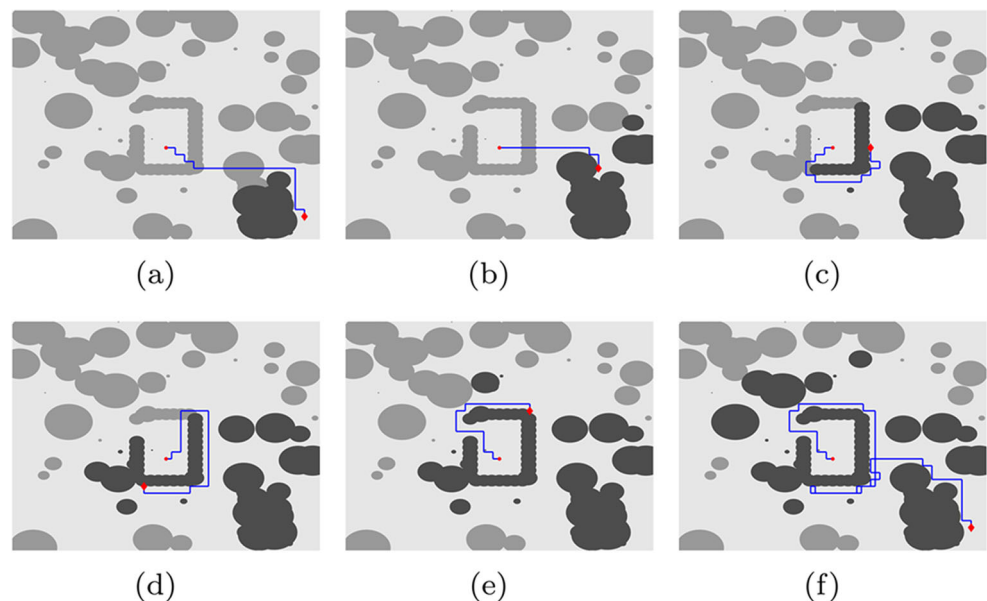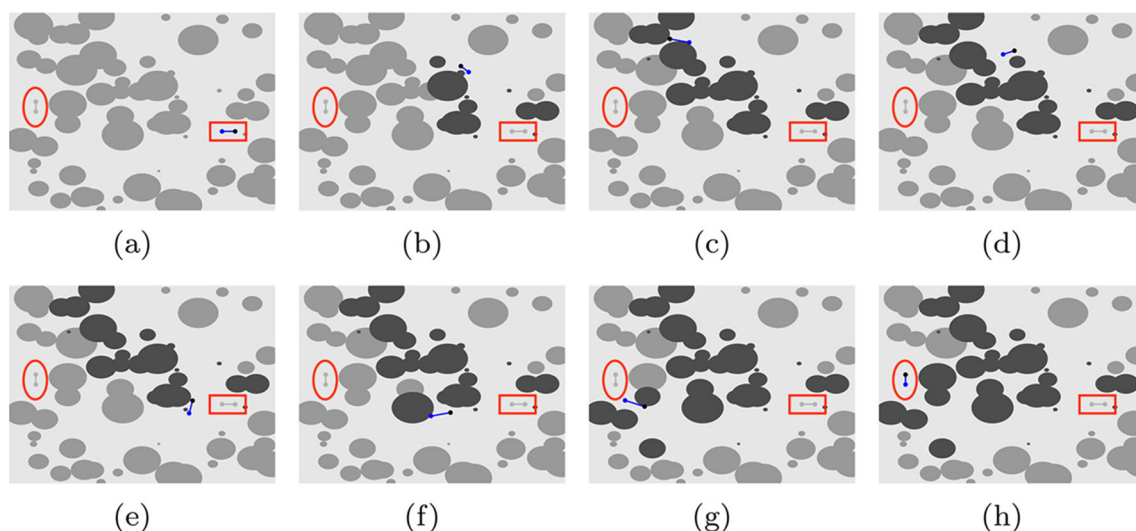
Fig. 5 Moving path for two robots with $l = 0.03$ with linking of each two robots not blocked by obstacles. The initial configuration is indicated by the red box. The target configuration is in the red ellipse. The robots move into a local trap first, and then move out of it before reaching the target configuration

motion. Besides, the link between each pair of robots cannot be blocked by obstacles. All examples are accompanied by youtube videos, with the web links given in the footnotes. In Fig. 5, a 2-robot (4 dimensional) system is used. From the pictures, we observe that the robots move up until trapped, because they always choose the fastest potential-decaying direction in the known environment. Then they retreat back and eventually find the correct way.[1] The next example is a 3-robot system (6 dimensional problem). The environment allows a direct path from the initial to the target. The algorithm immediately finds this direct path and avoids taking other sideways.[2] Finally, a 5-robot system is shown to demonstrate that the algorithm is capable of solving a 10-dimensional problem with complicated environment with video given,[3] in which the robots need to twist so that they can successfully pass through the gaps between obstacles. Another example of a 10-robot system (with 20 dimensional configuration space) can also be found online.[4] We would like to note that it takes about 1 minute to finish the entire computation (including the generation of the movie) for this 10-robot system by using Matlab on a regular laptop (a Macbook Pro with 2.9 GHz Intel Core i5 CPU) with no particular effort being made to optimize the implementation of the algorithm.

In addition to the displayed paths, we illustrate the performance of the algorithms by using several other measurements. Table 1 shows the collective information about the number of vertices in graphs generated during the procedure. "Figure" column indicates the corresponding figures of the examples, "num of robots" represents the number of robots, and "l" is the step size used in generating graphs. To show the efficiency of our algorithms, we use the average number of nodes in the graphs, represented in "avg", and the maximum number of vertices amongst all graphs which is listed in the column "max". We can see that as the dimension of the problem (indicated in the "dim" column) increases, the size of the graphs increases, but not as fast as the exponential growth with respect to the dimensionality.

Furthermore, we observe that the algorithm generates the particular graph with the maximum number of vertices when the robots are trapped in local minimizer (shown in "trapped" column). In the 6 dimensional example, the robots do not encounter any local minimizer, which results in much fewer vertices. In fact, the sizes of graphs are smaller than those in the four-dimensional case. We also observe that the number of graphs generated by the algorithm ("num of G" column) highly relies on the environments and the choices of the orthonormal bases. Thus it is not used as a criterion to judge the efficiency of the algorithm. Overall, our algorithm is relatively efficient especially when dealing with high dimensional problems. The most costly part is to escape the local traps, and we propose a couple of strategies to improve the performance in the next section.

4 Escaping Local Traps Rapidly

From the experiments conducted, we notice that the number of generated vertices increases when the robots are trapped

---

[1] Video at https://youtu.be/6wKe7wnlG58

[2] Video at https://youtu.be/q84VhKfYUyo

[3] Video at https://youtu.be/H5lfzAYbfRA

[4] Video at https://youtu.be/gVinTsto7pE

Table 1 Information about numbers of vertices for the examples

Figure	Num of robots	l	dim	avg	max	Trapped	Num of G
Figures 3 and 4	1	0.03	2	81.2	149	Yes	5
Figure 5	2	0.03	4	606.7	3433	Yes	9
No Figure	3	0.02	6	632.4	1183	No	5
No Figure	5	0.03	10	2178.4	6938	Yes	7

in local minimizers, and the number of nodes at each local trap is proportional to the volume of the trap. This is not a surprise because the nearly exhausted search is used to escape local traps. In order to reduce cost, we present two different strategies. Before doing so, we need to identify local minimizers and define their trap regions. We say that a node point x is a local minimizer if no lower-potential points around x can be generated by Algorithm 2. Since a local trap can only be created by constraints because of the convexity of the potential function, we define the trap region as a set enclosed by the boundary of local constraints and the level curve (hyper-surface in high dimensional problems) of potential function. To this ends, without loss of generalization, we assume the distance between two disconnected obstacles is greater than the graph generation radius l. If not, we can easily construct a new patch of obstacle to connect them. Under this assumption, the trap is only related to one connected obstacle, and we denote it as $\mathcal{O}(x)$, where x is a local minimizer. Now we define the trap region as

$$\mathcal{L}(x) = \{c \in \Omega : c\bar{x}_t \cap \mathcal{O}(x) \neq \emptyset, p(c) \geq p(x)\}, \qquad (5)$$

where $c\bar{x}_t$ is the straight line between c and x_t.

When a local trap is identified, our goal is to find points, starting from where the gradient flow trajectory to the target configuration is no longer blocked by the currently known environment as quickly as possible, and then continue to generate vertices outside of the trap region. Here we introduce two different dimension reduction methods to achieve this goal.

Keep the robot near obstacles: We know that some of the constraints in ϕ, ψ must be nearly activated around the local minimizer $x \in \partial\mathcal{O}$. For the ease of presentation, we denote those nearly activated constraints as $g_i(x) \leq \epsilon$ for some integer i where ϵ is a small positive number and g_i is some ϕ_j or ψ_k. For example, it can be chosen as

$$\epsilon = \min\left(\min_{k=1,2,\cdots,k_1} \phi_k(x), \min_{k=1,2,\cdots,k_2} \psi_k(x)\right).$$

We modify the algorithm so that it only generates points satisfying the inequalities, that is, only add points v such that $g_i(v) \leq \epsilon$ to V until there is a vertex $x \in \mathcal{E}$ with

$$\mathcal{E} = \{x \in B(y, \sqrt{2}l) : \exists z \in \{c \in (x \pm \mathcal{N})\backslash V : p(c) \\ < p(x)\}\backslash\mathcal{O}\},$$

where $\mathcal{N}$ is the substituted set of orthonormal basis for N in the subspace, V is the current vertex set, $y = \arg\max_{g_i(z)\leq\epsilon} p(z)$ and g_i is the constraint mentioned above. After this point, we go back to Algorithm 2. With the same assumptions as stated in Theorems 1 and 2, we can show that this method find the path in finite steps, and the proof of the convergence follows the same arguments as provided in Section 6. Since $g_i(x) = \epsilon$ is continuous locally, the modified search is conducted in a low-dimensional subspace if ϵ is chosen appropriately.

Fix the shape formed by robots: A different way to get out of the local traps is to introduce a set of new constraints $\{h_1(x) = 0, \cdots, h_k(x) = 0\}$ on the robots so that they restrict the graph generation in a low dimensional subspace $\hat{\Omega}$. For example, one may fix the pairwise distance between robots, so $h_i(x) = 0$ indicates that the distance between a certain pair of robots is a given value. In 2-D or 3-D workspace, those restrictions often lead to a fixed shape formed by the robots. Each h_i reduces the search dimension by one because the new vertices added to V must satisfy $h_i(x) = 0$. Similar to the previous strategy, we stop this procedure when a vertex $x \in \mathcal{E}$ is generated, which indicates the robots moved out of the known local trap. On the other hand, it is possible that after adding new constraints, there is no feasible way to move out. In this case, no new vertex can be generated in V, then we remove one of the added constraints, and continue with the graph generating algorithm in a subspace which is one dimension higher than the previous subspace. The procedure is repeated if necessary. For this method, if we further assume that there is a feasible tube in the low dimensional subspace defined by all constraints, including the added ones $h_i = 0$, we can use the same proof to show its convergence in a finite number of steps. In this paper, we implement this dimension reduction strategy in our high dimensional examples.

In the two and five robots cases demonstrated in Section 3, we fix the distance between each pair of robots when a local minimizer is encountered. To compare the results, we carry out several new experiments, in which all set-ups including initial and target configurations, the obstacles and all parameters are the same. The final path and how the robots move can be found in videos.[5] The information on the generated graphs is displayed in Table 2. When comparing the number in Table 2 with the numbers shown for 2 and 5 robots cases in Table 1, we can see that the number of vertices, both average and maximum numbers, decreases significantly. In the 5 robots case, the largest graph is no longer produced at local traps. Instead, the first generated graph contains more nodes because of the long distance from initial to target configurations. In our examples, we observe that nodes needed around the local minimizers are reduced from $O(\alpha^n)$ to $O(\alpha^2)$, where α is the edge length assuming the local trap is a square and n is the dimension of Ω.

We also observe a common feature in all examples: the environment is not entirely explored, and the generated graphs are greedily expanding towards the target configuration. This special feature is not by accident. In fact, it is determined by the Fokker-Planck equation in optimal transport theory. We give a thorough discussion on their connections in the next section.

5 Relation to FPE on Graph

The design of the graph generating algorithm, Algorithm 2, is inspired by the evolution of FPE, which determines a region $\mathcal{R}_f$ where the search is conducted. Although the algorithm does not require the knowledge of the region a priori, the region provides critical information for our understanding of the algorithmic behavior. In this section, we describe in detail on how the region evolves following the solution of FPE,

$$\begin{cases} \frac{\partial \rho}{\partial t}(x, t) = \nabla \cdot (\rho(x, t)\nabla p(x)) + \beta \Delta \rho(x, t) \\ \rho(x, 0) = \rho_0(x) \\ \nabla \rho(x, t) \cdot \mathbf{n} = 0, \qquad \forall x \in \partial \Omega \end{cases}, \qquad (6)$$

where ρ_0 is a given distribution, $\partial \Omega$ is the boundary of the configuration space Ω, $\mathbf{n}$ is the normal vector of $\partial \Omega$, and $p(x)$ is the potential function. Based on Eq. 6, the region $\mathcal{R}_f$ is constructed by an intermittent diffusion process, meaning we take β to be 0, so that the density is transported greedily along the negative gradient direction, while we adjust $\beta > 0$ to trigger a diffusion process when trapped in a local minimizer. For simplicity, we call $\beta = 0$ the gradient

Table 2 The numbers of vertices for selected examples using the escaping local traps algorithm

Num of robots	l	dim	avg	max	Num of G
2	0.03	4	212.4	295	8
5	0.03	10	1307	2492	7

As we can see, the algorithm generates much fewer vertices compared to the 2 and 5 robots cases in Table 1

part and $\beta > 0$ the diffusion part. However, since our graph generating algorithm only choose new points along the given orthonormal directions N, we must replace $\nabla p(x)$ in Eq. 6 by its projection onto N:

$$\mathbf{u}(x) = \sum_{y \in \mathcal{H}(x)} P_y \nabla p(x),$$

where P_y is the projection operator to y, and $\mathcal{H}(x)$ is defined as,

$$\mathcal{H}(x) = \arg \min_{y \in \mathcal{A}} \frac{\langle \nabla p(x), y \rangle}{\|\nabla p(x)\|}$$

with $\mathcal{A} = \{y : y \in N \text{ or } -y \in N \text{ and } x + \lambda y \notin \mathcal{O} \forall \lambda \in (0, \xi) \text{ for some } \xi > 0\}$. The resulting equation is

$$\begin{cases} \frac{\partial \rho}{\partial t}(x, t) = \nabla \cdot (\rho(x, t)\mathbf{u}(x)) + \beta \Delta \rho(x, t) \\ \rho(x, 0) = \rho_0(x) \\ \nabla \rho(x, t) \cdot \mathbf{n} = 0, \qquad \forall x \in \partial \Omega \end{cases}. \qquad (7)$$

We note that both Eqs. 6 and 7 can be rewritten as

$$\frac{\partial \rho}{\partial t}(x, t) = \nabla \cdot \left\{ \rho(x, t) \left[\mathbf{v}(x) + \nabla(\beta \log \rho(x, t)) \right] \right\},$$

where $\mathbf{v} = \nabla p$ for Eq. 6 and $\mathbf{v} = \mathbf{u}$ for Eq. 7. This expression can be approximated by the following upwind discretization of Eq. 6 on a lattice grid $GL \subset \Omega \backslash \mathcal{O}$ (here we assume that x_i is one of the grid points), with mesh size Δx and orientation aligned with the orthonormal basis N used in Algorithm 2 [10],

$$\frac{\partial \rho_j}{\partial t} = \left(\sum_{k \in Nb(j)} (F_k - F_j)_+ \rho_k d_{jk} - \sum_{k \in Nb(j)} (F_j - F_k)_+ \rho_j d_{jk} \right) \frac{1}{(\Delta x)^2}, \qquad (8)$$

where $(\cdot)_+ = \max(\cdot, 0)$, $\rho_j = \rho(x_j, t)$, $Nb(j)$ is the set of all adjacent nodes (neighbors) of node x_j on the grid GL, $F_j = \frac{\partial}{\partial \rho_j} \mathcal{F}(\rho, \beta)$, in which $\mathcal{F}(\rho, \beta)$ is the free energy

$$\mathcal{F}(\rho, \beta) = \sum_{j=1}^{N} \left(p(x_j)\rho_j + \beta \rho_j \log \rho_j \right), \qquad (9)$$

and $d_{jk} = d_{kj} = 1$ for Eq. 6. A similar discretization can be derived for Eq. 7. The value of d_{jk} in the discretization of $\mathbf{u}$,

which, if assume $\phi(x_j) > \phi(x_k)$ without loss of generality, can be defined as

$$d_{jk} = \begin{cases} 1 & \langle x_j - x_k, \nabla p(x_j) \rangle = \min_{\{i \in Nb(j):p(x_i)<p(x_j)\}} \langle x_j - x_i, \nabla p(x_j) \rangle \\ 0 & \text{otherwise} \end{cases}. \tag{10}$$

If the projection is not involved as is in the diffusion part, we simply let $d_{jk} = 1$ for all j, k.

In the rest of this section, we show how to build $\mathcal{R}_f$ using Eq. 8 with $\Delta x = l$. The strategy is that we alternate the procedures between the gradient ($\beta = 0$) and diffusion ($\beta \neq 0$) to grow the region. When a new part of region is formed each time, we simply union it with the existing one. We want to mention that at any point we change the procedures (β from 0 to a nonzero value, or vice versa), we reinitialize the density before evolving (7). We terminate the procedure, if the target configuration is included in $\mathcal{R}_f$.

5.1 Gradient Part of $\mathcal{R}_f$

For the gradient case, the points on the grid expand along the projection of the negative gradient of the potential function onto N. We evolve Eq. 8 with $\beta = 0$ and the initial condition

$$\rho(x, 0) = \delta_{x_i} = \begin{cases} 1 & x = x_i \\ 0 & x \neq x_i \end{cases},$$

where x_i is the starting point of the current gradient procedure. Until reaching the steady state, the solution $\rho(x, t)$ on the grid GL can be calculated. The steady state solution satisfies for the following property,

Proposition 2 $\rho(x, \infty) = \delta_{V_{loc}}$, where $V_{loc} \subset GL$ is a subset of local minimizers of $p(x)$ on the given grid.

Then we select points such that

$$\mathcal{R}_1(x_i) = \left\{ x \in GL : \frac{\partial}{\partial t} \rho(x, t) > 0 \text{ for some } t > 0 \right\}, \tag{11}$$

Once $\mathcal{R}_1(x_i)$ is determined, we merge it to the set $\mathcal{R}$ constructed in the previous steps (we use empty at the first step), i.e. $\mathcal{R} = \mathcal{R} \bigcup \mathcal{R}_1$. If $x_i \neq x_f$, we continue to amend the set $\mathcal{R}$ with the diffusion procedure described in Section 5.2.

5.2 Diffusion Part of $\mathcal{R}_f$

We assume that the previously constructed set is $\mathcal{R}_p = \mathcal{R}$. In the diffusion part, since $\beta > 0$, $\log \rho$ is involved in the

calculation. To avoid blowing up in the computation, we initialize the density β for Eq. 8 as below:

$$\rho(x, 0) = \delta_{\mathcal{R}_p} = \begin{cases} \frac{1}{|\mathcal{R}_p|}(1 - \epsilon) & x \in \mathcal{R}_p \\ \frac{1}{|GL \backslash \mathcal{R}_p|}\epsilon & x \in GL \backslash \mathcal{R}_p \end{cases},$$

where ϵ can be an arbitrarily small positive real number. With this initialization, we calculate $\rho(x, t)$ in Eq. 8 until reaching the stationary solution $\rho(x, \infty)$. Now following $\rho(x, \infty)$, we choose points on the grid as:

$$\mathcal{R}_2^s = \left\{ x = \arg \max_{y \in GL \backslash \bigcup_{j=0}^{s-1} \mathcal{R}_2^j} \rho(y, \infty) : \right.$$
$$\left. \mathcal{R}_2^{s-1} \bigcap Nb(x) \neq \emptyset \right\},$$
$$\mathcal{R}_2^0 = \mathcal{R}_p, \text{ and } \mathcal{R}_2 = \bigcup_{s=0}^{W} \mathcal{R}_2^s,$$

where W is defined by

$$W = \arg \min_s \left\{ s : z \in \mathcal{R}_2^s, \exists y \in Nb(z) \backslash \left(\bigcup_{\tau=1}^{s} \mathcal{R}_2^\tau \right) \right.$$
$$\left. \text{with } p(y) < p(z) \right\}.$$

We union $\mathcal{R}_2$ into $\mathcal{R}$ by defining $\mathcal{R} = \mathcal{R} \bigcup \mathcal{R}_2$. In the newly selected $\mathcal{R}_2$, we pick

$$x_i = \arg \min_y \{ p(y) : y \in Nb(x) \backslash \mathcal{R} \text{ for some } x \in \mathcal{R} \}. \tag{12}$$

This x_i is the new starting point for the next gradient procedure, and we return to the gradient part as described in Section 5.1.

By alternating the procedures to obtain $\mathcal{R}_1$ and $\mathcal{R}_2$ until x_f is included, we define the final region

$$\mathcal{R}_f = \bigcup_{x \in \mathcal{R}} Box(x, l)$$

where $Box(x, l)$ is the closed box centered at x with edge length $2l$. With the constructed $\mathcal{R}_f$, we have the following theorem.

Theorem 3 *Assuming that the robots only stops on node points with assumptions in Theorem 2 and $R > L$, the complete path γ generated by the algorithm satisfies $\gamma \subset \mathcal{R}_f$.*

The proofs of Theorem 3 and Proposition 2 are given in Section 6.

We show two examples in Fig. 6 with initial configurations indicated by red diamonds and target red circles. The gray region is calculated by the gradient and diffusion procedures described in this section. The computation is done on a grid with mesh size $\Delta x = l$. As we can see clearly in Fig. 6a, starting from the right middle part of Ω, the graph G first expands along the x-axis, which is the projected negative gradient direction on the lattice grid, until it hits an obstacle. Then the procedure is switched to the diffusion case, and produces a region in front of the obstacle following the Gibbs' distribution until finding a way out. After that, the procedure changes back to the gradient case and moves to the target greedily. This time, the projected negative gradient direction coincides with the actual one. Again, the diffusion case kicks in when a local minimizer is encountered. The procedure repeats until the target is reached. Figure 6b shows another example with more complicated set ups.

We would like to mention that we use the forward Euler method to discretize Eq. 8 in time,

$$\rho_j^{n+1} = \rho_j^n + \left(\sum_{k \in Nb(j)} (F_k - F_j)_+ \rho_k^n d_{jk} \right.$$
$$\left. - \sum_{k \in Nb(j)} (F_j - F_k)_+ \rho_j^n d_{jk} \right) \frac{\Delta t}{(\Delta x)^2}. \quad (13)$$

To make the scheme convergent, we need the following proposition, which can be regarded as the Courant-Friedrichs-Lewy (CFL) conditions for numerical PDE schemes.

Proposition 3 *Given the lattice grid with grid size Δx, Eq. 13 is stable if*

$$\Delta t < \min \left\{ \frac{1}{\max_{j \in GL} \sum_{k \in Nb(j)} (F_j - F_k)_+ d_{jk}}, \right.$$
$$\left. \min_{j \in GL} \frac{1 - \rho_j}{\sum_{k \in Nb(j)} (F_k - F_j)_+ \rho_k d_{jk}} \right\} (\Delta x)^2, \quad (14)$$

Proof To make the scheme stable, we need

$$\begin{cases} \rho_j + \sum_{k \in Nb(j)} (F_k - F_j)_+ \rho_k d_{jk} \frac{\Delta t}{(\Delta x)^2} < 1 \\ \rho_j - \sum_{k \in Nb(j)} (F_j - F_k)_+ \rho_j d_{jk} \frac{\Delta t}{(\Delta x)^2} > 0 \end{cases}$$
$$\implies \begin{cases} \Delta t < \frac{1}{\sum_{k \in Nb(j)} (F_j - F_k)_+ d_{jk}} (\Delta x)^2 \\ \Delta t < \frac{1 - \rho_j}{\sum_{k \in Nb(j)} (F_k - F_j)_+ \rho_k d_{jk}} (\Delta x)^2 \end{cases}$$

for all $j \in GL$. This leads to Eq. 14. In addition, following the proof of Theorem 3 in [33], we can obtain $\rho_i(t) \geq \epsilon$ for a fixed grid. Therefore $(F_j - F_k)_+$ and $(F_j - F_k)_+ \rho_k$ are bounded from above for all edges $\{(j, k)\}$ in GL. This implies that the right hand side of Eq. 14 is bounded from below by a positive real number, so Δt can stay strictly positive. □

Remark 2 For each given l, we can get a region $\mathcal{R}_f(l)$. If we let l go to 0, which means that the lattice GL approaches to the continuous space, we can define $\mathcal{R}_f(0) = \limsup_{l \to 0} \mathcal{R}_f(l)$. The graph G produced by our algorithm must satisfy $G \subset \mathcal{R}_f(l)$. In fact, $\mathcal{R}_f(0)$ is the smallest bounded region in which the search is conducted. We want to re-iterate that our algorithm neither involves solving the FPE on graph, nor needs to know the region in advance. The region that we constructed here is only used to understand the behavior of the algorithm, and for the proof of the convergence.

Fig. 6 In both (a) and (b), there exists gradient and diffusion part. The gray shadow is $\mathcal{R}_f$ and the blue part is the graph G

(a)   (b)
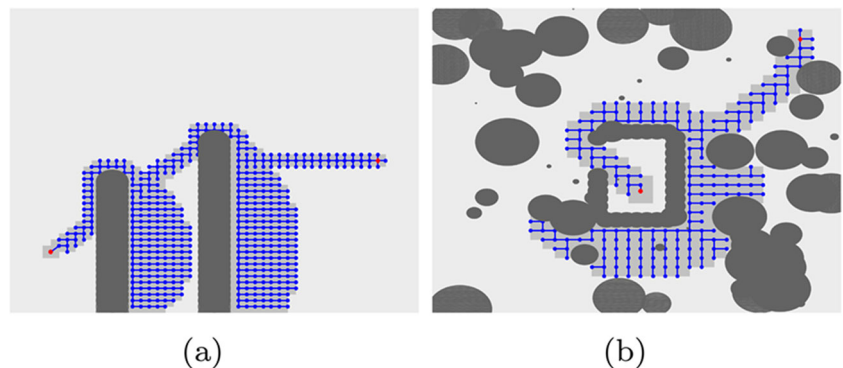
6 Convergence Analysis

6.1 Convergence of the Algorithm

In this section, we give detailed proofs for the Theorems 1 and 2. To do this, we first prove several lemmas. We begin with showing that the set of all feasible path Γ is compact in a finite time interval $[0, T]$ (Lemma 1) and there exists a feasible path in Γ within a tubular region $\mathcal{T}$ that is clear of obstacles (Lemma 2).

Lemma 1 *If there exists at least a feasible path, Γ is non-empty and compact with respect to the L^∞ norm given by $d_\Gamma(\gamma_1, \gamma_2) = \max_{t \in [0,T]} \|\gamma_1(t) - \gamma_2(t)\|$, where γ_1 and γ_2 are two paths in Γ.*

Proof Let us denote the feasible path as $\gamma : [0, T_0] \to \Omega$. If we define $\gamma(t) = x_t$ for all $t \in (T_0, T]$, we have $\gamma \in \Gamma$. To prove Γ is compact, we assume there is a sequence of paths $\{\gamma_i\}_{i=0}^\infty \subset \Gamma$ such that $\lim_{i \to \infty} d_\Gamma(\gamma_i, \gamma) = 0$. Since Ω is compact and $\mathcal{O}$ is open, it implies that $(\Omega \backslash \mathcal{O})$ is compact. Therefore for any $t \in [0, T]$, $\gamma_i(t) \to \gamma(t)$ as $i \to \infty$, we have $\gamma(t) \in (\Omega \backslash \mathcal{O})$. This includes

$$\gamma(0) = \lim_{i \to \infty} \gamma_i(0) = x_s, \quad \gamma(T) = \lim_{i \to \infty} \gamma_i(T)$$
$$= x_t, \quad \gamma(t) \subset (\Omega \backslash \mathcal{O}),$$

which gives $\gamma \in \Gamma$, and Γ is a close set. In addition, since Ω is compact, Γ is bounded, we conclude that Γ is compact. $\square$

Lemma 2 *Assume that* Eq. 2 *is true, there must exist a feasible path $\gamma \in \Gamma$ satisfying $\left(\cup_{t \in [0,T]} B(\gamma(t), L)\right) \cap \mathcal{O} = \emptyset$, where $\mathcal{O}$ is the constrained set.*

Proof Because of Eq. 2, there exists a sequence of paths $\{\gamma_n\} \subset \Gamma$ satisfying

$$\lim_{n \to \infty} \inf_{t \in [0,T]} \sup_{r \in [0, \infty)} \{r : B(\gamma_n(t), r) \cap \mathcal{O} = \emptyset\} = L.$$

From Lemma 1, we know that Γ is compact, therefore $\lim_{n \to \infty} \gamma_n = \gamma_0 \in \Gamma$. For an arbitrary $t \in [0, T]$, denote

$$L(t) = \sup_{r \in [0, \infty)} \{r : B(\gamma_0(t), r) \cap \mathcal{O} = \emptyset\}$$

As the whole space Ω is compact, the limit $L(t) < \infty$. Since $\inf_{t \in [0,T]} L(t) = L$, one has $L \leq L(t)$ for arbitrary time $t \in [0, T]$. That is, fix the curve γ_0 defined as above, for all $t \in [0, T]$, $B(\gamma_0(t), L) \cap \mathcal{O} = \emptyset$ and thus $\left(\cup_{t \in [0,1]} B(\gamma_0(t), L)\right) \cap \mathcal{O} = \emptyset$, which proves the lemma. $\square$

In the next few lemmas, we prove several results that ensure the generating graph algorithm (Algorithm 2) creating new points in the feasible region when the radius l is

small enough compared to L, and the process does not stop until reaching a neighborhood of the target configuration.

Lemma 3 *Given a point x on an n-dimensional Euclidean space and $L > 0$. Let $y \in S(x, L)$, and N be a set containing n orthonormal vectors, then $\exists z \in K = \{y \pm lN : y \pm le, e \in N\}$ such that $z \in B(x, L)$ if $0 < l < (2L)/\sqrt{n}$.*

Proof Without loss of generality, we assume $x = (0, \cdots, 0)$. Let us denote $y = (y_1, \cdots, y_n)$, we can rewrite $K = \{z_k = (y_1, \cdots, y_k \pm l, \cdots, y_n)\}_{k=1}^n$. Since $y \in S(x, L)$, then $\exists k \in \{1, \cdots, n\}$, so that $|y_k| \geq L/\sqrt{n}$. Consider the point $z_k = (y_1, \cdots, y_k - sign(y_k)l, \cdots, y_n)$, then

$$\|z_k - x\|^2 = \sum_{i \neq k} y_i^2 + (y_k - sign(y_k)l)^2$$
$$= L^2 + l^2 - 2 sign(y_k)y_k l$$
$$= L^2 + l^2 - |y_k|l \leq L^2 + l^2 - (2Ll)/\sqrt{n}.$$

One has $L^2 + l^2 - (2Ll)/\sqrt{n} < L^2$ if $0 < l < (2L)/\sqrt{n}$. So $z_k \in B(x, L)$. $\square$

Lemma 4 *Given a continuous path γ and a closed set $\mathcal{V} \subset \Omega$ with $\gamma(0) = x, \gamma(T) = y$, and $x \in \mathcal{V}$ and $y \notin \mathcal{V}$, then there exists $z \in \gamma$ such that $z \in \partial\mathcal{V}$ and $\gamma \cap \partial\mathcal{V}$ is closed.*

Proof We use the signed distance function $f(u, \mathcal{V})$ with $f > 0$ if $u \in \mathcal{V}$ and $f < 0$ when $u \notin \mathcal{V}$. It is clear that f is continuous with respect to u. Hence,

$$g = f \circ \gamma : [0, T] \to \Omega$$

is also continuous with $g(0) \geq 0$ and $g(T) \leq 0$ since $\gamma(T) = y \notin \mathcal{V}$ and $\gamma(0) = x \in \mathcal{V}$. As a result, there is at least one point $t_1 \in [0, T]$ so that $g(t_1) = 0$, hence $\gamma(t_1) \in \partial\mathcal{V}$. In fact, all points satisfying $f(u) = 0$ are on $\partial\mathcal{V}$.

Assume $\gamma \cap \partial\mathcal{V}$ is open, $\mathcal{B} = (f \circ \gamma)^{-1}(\gamma \cap \partial\mathcal{V})$ must be open, because $(f \circ \gamma)$ is continuous. This implies that $\mathcal{B}$ is the disjoint union of some open intervals. Take one of the open interval, say (a, b), we have $a \notin \mathcal{B}$ and $f \circ \gamma((a, b)) = 0$. Due to the continuity of $f \circ \gamma$, we have $f \circ \gamma(a) = 0$, which means $a \in \mathcal{B}$, and this is a contradiction. Therefore, $\gamma \cap \partial\mathcal{V}$ must be closed. $\square$

Lemma 5 *Assume that* Eq. 2 *holds. Graph $G = (V, E, K, p)$ is generated by Algorithm 2 with $l \leq \frac{2L}{\sqrt{n}}$. If $x_t \notin V$, then the graph generating step of Algorithm 2 does not stop, and there exists at least one point in the feasible region that can be added to G by the algorithm.*

Proof Let us assume that the graph generating algorithm terminates after finite steps, returning a connected graph $G = (V, E)$ containing x_s. We denote $C = \cup_{v \in V} \bar{B}(v, L)$, which is a closed set with $x_s \in C$. First we want to prove $x_t \in C$ by contradiction. Let us assume $x_t \notin C$. Take

the path γ in Lemma 2, it is true that $\gamma(0) = x_s \in C$ while $\gamma(T_0) = x_t \notin C$. Since γ is continuous, there exists at least one point that γ intersects with ∂C by Lemma 4, and we denote it as $\{\gamma(t_i)\}$. Let $x = \sup_{t_i} \gamma(t_i)$ be the last intersection point along the path. By Algorithm 2, we can find a vertex $v_c \in V$ such that $x \in S(v_c, L)$, which is the sphere centered at v_c with radius L. Since $x \in \gamma$ and by Eq. 2, we know that $B(x, L) \subset (\Omega \setminus \mathcal{O})$. Further we claim that there is no $v \in V \cap B(x, L)$, otherwise, $x \in B(v, L)$ implies $x \in C^o$, which is a contradiction with $x \in \partial C$.

Since the algorithm stopped, all current vertices $v \in V$ must have been tried to generate points around them. When the vertex v_c is chosen, by Lemma 3, at least one point p can be generated in $B(x, L)$, which means either the algorithm should not terminate without having p, or p already exists before, which contradicts there is no vertex of G in $B(x, L)$. Therefore, we conclude that $x_t \in C$ if the algorithm stops.

If $x_t \in C$, since $x_t = \gamma(T_0)$, $B(x_t, L) \subset (\Omega \setminus \mathcal{O})$, then there is at least one vertex $v \in B(x_t, L)$, by the algorithm, an edge between x_t and v should be added to the graph G. Thus, if the algorithm stops, x_t and x_s are in the same connected component in the graph G. □

Now, we are ready to prove the main theorems.

Proof (Proof of Theorem 1) Since Ω is compact and the graph G is a subset of grid points with length l in Ω, G must contain a finite number of vertices. Otherwise, there is a cluster point in the vertex set, which implies that there exist two vertices in G whose distance is strictly smaller than the generating radius l regardless how small l is. This contradicts to the fact that that G is a subset of a grid with the smallest distance between any two points is l. From the construction mechanism, G is always connected. Then Lemma 5 implies that G connects x_s and x_t. □

Proof (Proof of Theorem 2) We denote the detected constraints after step i as $\mathcal{O}_c^i$. If the algorithm does not stop in finite steps, it means $m = \infty$. Because Ω is compact and $\mathcal{O}_c^i$ is open for every i, then $\Omega \setminus \mathcal{O}_c^i$ is also compact. Thus, there exists a cluster point for the stopping point set $\{\gamma_k(T_k)\}_{k=1}^{\infty}$. For an arbitrary $\epsilon > 0$, i can be chosen so that $\|\gamma_i(T_i) - \gamma_j(T_j)\| < \epsilon$ for all $j > i$. Choose $\epsilon < R - q$. Without loss of generality, one can assume that γ_i is obtained before γ_j, and at j-th step, the algorithm stops because $\mathcal{O}_c \cap \gamma_j \neq \emptyset$. Even more,

$$\bar{B}(\gamma_i(T_i), R) \cap \mathcal{O}_c \cap \gamma_j \neq \emptyset.$$

And since

$$\bar{B}(\gamma_i(T_i), R) \cap \mathcal{O}_c = \mathcal{O}_c^i,$$

it is true that

$$\mathcal{O}_c^i \cap \gamma_j \neq \emptyset$$

which is a contradiction since for each generated graph $G = (V, E, p)$ at step j,

$$V \cap \mathcal{O}_c^i = \emptyset \text{ and } E \cap \mathcal{O}_c^i = \emptyset$$

for all $i < j$, which concludes $m < \infty$. □

6.2 Proof of the Bounded Searching Region

In this section, we prove Theorem 3 and related propositions. First, we give a detailed proof for the following lemma.

Lemma 6 *Given $\beta \geq 0$, Eq. 8 is convergent with $\mathcal{F}(\rho, \beta)$, given in* Eq. 9, *being a Lyapunov function. When $\beta \neq 0$, Eq. 8 converges, with any given initial condition, to the Gibbs' distribution*

$$\rho(x) = \frac{1}{K} \exp\left(-\frac{p(x)}{\beta}\right),$$

where K is the normalization constant making ρ a density function.

Proof Taking the derivative along the solution $\rho(x, t)$ of Eq. 8, we have

$$\begin{aligned}
&\frac{d}{dt}\mathcal{F}(\rho(t)) \\
&= \sum_{j=1}^{|V|} F_j \frac{d\rho_j}{dt} = \sum_{j=1}^{|V|} \sum_{k \in Nb(j)} ((F_k - F_j)_+ \rho_k d_{jk} F_j \\
&\quad - (F_j - F_k)_+ \rho_j d_{jk} F_j) \\
&= \sum_{j=1}^{|V|} \sum_{k \in Nb(j)} (F_j - F_k)_+ \rho_k d_{jk} F_k - \sum_{j=1}^{|V|} \sum_{k \in Nb(j)} \\
&\quad (F_j - F_k)_+ \rho_j d_{jk} F_j \\
&= -\sum_{j=1}^{|V|} \sum_{k \in Nb(j)} (F_j - F_k)_+^2 \rho_k d_{jk} \leq 0
\end{aligned}$$

Because $\mathcal{F}(\rho, \beta)$ is bounded from below, Eq. 8 is convergent. One can check directly that the Gibbs' distribution is the stationary solution of Eq. 8, and it is also the minimizer of $\mathcal{F}(\rho, \beta)$. □

Proof (Proof of Proposition 2) First of all, by Lemma 6, Eq. 8 converges when $\beta = 0$. We assume that the support of one stationary solution $\rho(x)$ contains a point other than local minimizers, say $x_0 \in GL$. Then there exists $z \in Nb(x_0)$ with $p(z) < p(x_0)$. By the definition of d_{ij}, there must be an edge linking x_0 and z such that $d_{x_0 z} = 1$ and $p(z) < p(x_0)$. We construct

$$\hat{\rho}(x) = \begin{cases} \rho(x) & x \neq x_0, x \neq z \\ 0 & x = x_0 \\ \rho(z) + \rho(x_0) & x = z \end{cases}$$

It is easy to check $\mathcal{F}(\hat{\rho}) < \mathcal{F}(\rho)$, which leads to a contradiction that ρ is the minimizer of $\mathcal{F}(\rho)$. $\quad\square$

Given an arbitrary environment $\mathcal{O}$, let us denote $G_f = (V_f, E_f)$ the graph generated by Algorithm 2 with $\mathcal{O}_c = \mathcal{O}$ all the time. We show that any node in V_f must be in the region $\mathcal{R}_f$ constructed in Section 5.

Lemma 7 *If $x \in V_f$, then $x \in \mathcal{R}_f$.*

Proof Since the grid size of GL is l, we have $G_f \subset GL$, meaning all vertices in G_f are grid points on GL. In fact every node generated by the algorithm must be a grid point on GL. By the definition of $\mathcal{R}_f$, we only need to prove that $V_o \subset \mathcal{R}$, where V_o is the set of vertices which are chosen to generate new nodes by Algorithm 2. For convenience, we call a node in V_o an interior point of G_f. We call x the ancestor of y if $y \in Nb_V(x)$, the neighborhood of x, and x is generated earlier than y by Algorithm 2. Correspondingly we call y the child of x. We call Algorithm 2 in gradient steps if the potential of the newly chosen node is lower than its ancestor, otherwise we call it in diffusion steps. We use induction to prove the lemma. Because G_f contains a finite number of nodes, the induction process stops after finite steps. At the first step, $x_s \in V_o$ and $x_s \in \mathcal{R}$. We assume that the first k generated nodes in G_f are contained in $\mathcal{R}$. We want to prove that the node $z \in \mathcal{R}$, where z is the node to generate new points at $(k + 1)$-th step. We assume w is the ancestor of z that generates w. Obviously, $w \in \mathcal{R}$ by our induction assumption. There are two cases.

Gradient case ($p(z) < p(w)$): by the construction of R in Section 5.1, to prove $z \in \mathcal{R}$, we only need to show $\rho_t(z, T) > 0$ for some T. There are two scenarios. (1) There exists T_1 such that $\rho(z, T_1) > 0$. In this case, we must be able to find $T_2 \leq T_1$ with $\rho_t(z, T_2) > 0$, because $\rho(z, 0) = 0$ initially. (2) $\rho(z, t) = 0$ for all $t \in [0, T_1]$. Because the algorithm picks the point that has the lowest p value, we have $p(w) - p(z) > p(w) - p(u)$ for all $u \in Nb(w)$, which gives $d_{wz} = 1$ in Eq. 13. Notice that $w \in \mathcal{R}$, this means there exists T_3 such that $\rho_t(w, T_3) > 0$, which implies that there is a time interval $I = [T_3 - \Delta t, T_3 + \Delta t]$, $\rho_t(w, t) > 0$ for all $t \in I$. Thus, there must exist a T_4 such that $\rho(w, T_4) > 0$. By Eq. 8, we have $\rho_t(z, T_4) > 0$ if $\rho(z, T_4) = 0$, which implies $z \in \mathcal{R}$.

Diffusion case ($p(z) \geq p(w)$): We recall $\mathcal{R}_2 = \bigcup_{s=1}^{W} \mathcal{R}_2^s$. It is easy to see that if $s > t$, we have $p(x) > p(y)$ for arbitrary $x \in \mathcal{R}_2^s$ and $y \in \mathcal{R}_2^t$ because of the property of the Gibbs' distribution. Otherwise the diffusion procedure for the set construction is stopped

according to the definition of W. We split the proof into three steps for this case. Firstly, we let

$$x = \arg\min_z \{p(z) : z \in \bigcup_{y \in \mathcal{R}_2^W} Nb(y) \setminus \mathcal{R}_2\}.$$

and $x_p \in Nb(x) \cap \mathcal{R}_2$. By the definition of W, we know that $x \in \mathcal{R}$, and x is the choice for the starting configuration for the next step. And also, we claim that x exists. Since $x_f \in GL$ has the lowest potential and GL is a connected graph, there must be a node in $\mathcal{R}$ such that one of the outside neighbor has a lower potential, otherwise the construction of $\mathcal{R}$ do not stop.

Secondly, we claim that the part of the V_o generated by the diffusion steps of graph generation is contained in $\mathcal{R}_2$, and we simply use V_o to denote the diffusion part generated by the algorithm. Otherwise, there exists $z \in V_o \setminus \mathcal{R}_2$ with its ancestor $w \in \mathcal{R}_2^s$ for some $s < W$. Without loss of generality, we assume z is the first node outside of $\mathcal{R}_2$ chosen by Algorithm 2. Because $p(z) > p(w)$, we have $z \notin \bigcup_{i=1}^{s} \mathcal{R}_2^i$. Also, z is a candidate for all $\mathcal{R}_2^i$ with $i = s + 1, \cdots, W$. Therefore, $z \notin \mathcal{R}_2$ means that $p(z) > p(u)$ for all $u \in \mathcal{R}_2$. By Algorithm 2, this can happen only when $\mathcal{R}_2^o \subset V_o$ where

$$\bar{\mathcal{R}}_2^o = \{u : u \in \mathcal{R}_2, p(u) \leq p(x_p)\},$$

and $\mathcal{R}_2^o \subset \bar{\mathcal{R}}_2^o$ is the connected component of $\bar{\mathcal{R}}_2^o$ containing x_p. Otherwise there exists $v \in Nb_V(a) \setminus V_o$ for some $a \in V_o$ such that $v \in \mathcal{R}_2$. In this situation, the algorithm will choose v instead of z because of $p(v) < p(z)$. This cause a contradiction with the fact z is the next node selected by the algorithm. On the other hand, if $\mathcal{R}_2^o \subset V_o$ holds, we have $x \in V_o$ and there exists $y \in V_o$ such that $p(y) \leq p(x) < p(z)$, leading to the fact that Algorithm 2 does not put z into V_o. Thus, $V_o \subset \mathcal{R}_2 \subset \mathcal{R}$.

At last, we claim that $x \in V_o$, which gives us that the start node of the next gradient step shared by both the construction of $\mathcal{R}$ and the graph generation in the algorithm. That is, x is chosen by Algorithm 2 to generate new points. Otherwise by the same argument for the existence of x, there is some node $y \in V$ such that Algorithm 2 generates a node q based on y with $p(q) < p(y)$ and $p(y) - p(q)$ maximizes the potential gap amongst all similar pairs. So, $y \in V_o$. However, by the definition of x, it is the lowest potential point has this property within the region $\mathcal{R}_2$. Thus $y = x$. And in the next generation step, q will be generated, and meanwhile, the region growing procedure will also choose q as the start point of the next gradient step. $\quad\square$

Proof (Proof of Theorem 3) Let $G_f = (V_f, E_f)$ be the graph generated by the algorithm with $\mathcal{O}_c = \mathcal{O}$ all the time.

We denote $G = (V, E)$ a graph generated with currently known environment $\mathcal{O}_c$ and initial configuration at x. Since $x \in G_f$ at the first step, a simple induction argument can show that every initial configuration x used to generate the graph G must satisfy $x \in G_f$ too. We claim that G and G_f must be the same in the ball $B(x, R)$, where we recall R the detectable radius, i.e. $G \cap B(x, R) = G_f \cap B(x, R)$, because G and G_f are generated by the same algorithm with the same knowledge of the environment in $B(x, R)$. We denote the connected common part of G and G_f as G_o, and further claim that the path γ is in G_o. If not, there must exist the first node generated in the graph generation step $y \in G \backslash G_o$ on the feasible trajectory on G. This implies that $y \in \mathcal{O} \backslash \mathcal{O}_c$ since the ancestor of y is in G_o, which can not contain any node in the infeasible region. However when the robots move along the path γ to the node before y, the system can detect that y is infeasible because $R > L$, and should stop before reaching y, which contradict with the fact that y is on the feasible trajectory. This concludes that the trajectory of the robots is restricted on $G_o \subset G_f$ in arbitrarily given known environment $\mathcal{O}_c \subset \mathcal{O}$. By Lemma 7, G_f is bounded by $\mathcal{R}_f$, hence the trajectory of the system is bounded by $\mathcal{R}_f$. $\qquad\square$

6.3 Proof of Proposition 2.2

Proof The existence of a path from the current configuration to the target is provided by the construction of the graph. We note that if two nodes share an edge, one of them is the ancestor of the other. Clearly, we have that x_c is the ancestor of x_2. By induction, if we assume that x_i is the ancestor of x_{i+1}, we claim that x_{i+1} is the ancestor of x_{i+2}. Otherwise, it implies that x_{i+2} must be the ancestor of x_{i+1}. This means that x_{i+i} has two ancestors, which is a contradiction with the graph generation strategy.

For the uniqueness of the path, we first notice that the algorithm stops once an edge is linked to the target x_f, from which we conclude that x_f has unique ancestor. By Algorithm 2, every node except x_c has a unique ancestor. If we assume that there are two paths γ and ξ, denoted by $\{y_i\}_{i=1}^{m_1}$ and $\{z_i\}_{i=1}^{m_2}$ respectively, we must have $y_1 = z_1 = x_c$ and $y_{m_1} = z_{m_2} = x_f$. Because of the uniqueness of the ancestor for each node, we must have $y_{m_1-1} = z_{m_2-1}$. By induction, we have $m_1 = m_2$ and $y_i = z_i$. Thus, $\gamma = \xi$ and the uniqueness is proven. $\qquad\square$

7 Conclusion

In this paper, an iterative algorithm is presented to solve the pathfinding problem in unknown environments. The algorithm is inspired by the evolution of FPE on graph in optimal transport theory. It contains graph generating, path finding and environment updating steps, among which graph generating is the key one. Guided by a potential function, the graph generating algorithm creates a tree structure, originating from the initial configuration, and terminating at the target configuration without solving the FPE numerically. Our approach has several advantages. First of all, the algorithm is deterministic and complete. It terminates in finite steps, returning a path, if there exists one. Secondly, the number of added nodes in the generated tree grows linearly with respect to the dimension of the configuration space. Together with the dimension reduction strategies for escaping the local traps, our algorithm can be used to compute high dimensional problems. Lastly, as proved by using FPE and intermittent diffusion, only part of the configuration space needs to be searched. We also emphasize that our assumption on the existence of a feasible path is only a technical requirement for the proof of the convergence. If such a path does not exist, the proposed algorithm can recognize the situation and stop the calculation in a finite number of iterations. In this case, one may conclude that there is not a feasible path which can be identified by using l as the search step size, or reduce l to further refine the path exploration computation.

The proposed algorithm is our initial investigation of using FPE for path exploration. Further improvements can be made on many fronts. For example, one can speed up the calculation in escaping the traps at local minima, by using random graph generating strategies. This is very useful in high dimensional situations. The graph generating radius l can be adaptive. Longer steps can be taken in wide open space and shorter step size is used when encountered narrow corridors. Although the path discovered by the robots is only a feasible one, the path and the environmental knowledge gained during the exploration can be used as valuable prior for the follow up groups of robots tasked to traverse the same area. In that case, the path can be used as the baseline to start iterative procedures that eventually create locally optimal path for further usage. In our current study, we don't consider some practical scenarios such as each agent is under slip or communication delay. The algorithm needs to be adjusted to account for those constraints. In addition, the proposed algorithms can be adapted to construct a general strategy for control problems with constraints, especially when some constraints are only available during the processes. Those are directions that are worth further studies, and we will report our findings in future papers.

Availability of data and material Not applicable

Code Availability All source code can be accessed on https://github.com/haoyanzhai/path_planning.git

Declarations

Conflict of Interest We declare that we have no conflict of interest.

Ethics approval and consent to participate Not applicable

Consent for Publication Not applicable

References

1. Ahlin, K.: The secant and traveling artificial potential field approaches to high dimensional robotic path planning. PhD Thesis, Georgia Institute of Technology (2018)
2. Amato, N.M., Wu, Y.: A randomized roadmap method for path and manipulation planning. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 113–120. IEEE (1996)
3. Bandyopadhyay, S., Chung, S.J., Hadaegh, F.Y.: Probabilistic swarm guidance using optimal transport. In: 2014 IEEE Conference on Control Applications (CCA), pp. 498–505 (2014)
4. Barraquand, J., Kavraki, L., Latombe, J.C., Motwani, R., Li, T.Y., Raghavan, P.: A random sampling scheme for path planning. Int. J. Robot. Res. **16**(6), 759–774 (1997)
5. Boor, V., Overmars, M.H., Van Der Stappen, A.F.: The Gaussian sampling strategy for probabilistic roadmap planners. In: 1999 IEEE International Conference on Robotics and Automation, 1999. Proceedings, vol. 2, pp. 1018–1023. IEEE (1999)
6. Branicky, M.S., LaValle, S.M., Olson, K., Yang, L.: Quasi-randomized path planning. In: IEEE International Conference on Robotics and Automation, 2001. Proceedings 2001 ICRA, vol. 2, pp. 1481–1487. IEEE (2001)
7. Brenier, Y.: Décomposition polaire et réarrangement monotone des champs de vecteurs. C. R. Acad. Sci. Paris sér. I Math. **305**, 805–808 (1987)
8. Brenier, Y.: Polar factorization and monotone rearrangement of vector-valued functions. Commun. Pure Appl. Math. **44**(4), 375–417 (1991)
9. Chen, Y.B., Luo, G.C., Mei, Y.S., Yu, J.Q., Su, X.L.: Uav path planning using artificial potential field method updated by optimal control theory. Int. J. Syst. Sci. **47**(6), 1407–1420 (2016)
10. Chow, S.N., Huang, W., Li, Y., Zhou, H.: Fokker–planck equations for a free energy functional or markov process on a graph. Arch. Ration. Mech. Anal. **203**(3), 969–1008 (2012)
11. Chow, S.N., Yang, T.S., Zhou, H.M.: Global optimizations by intermittent diffusion. In: Chaos, CNN, Memristors and Beyond: a Festschrift for Leon Chua with DVD-ROM, Composed by Eleonora Bilotta, pp. 466–479. World Scientific (2013)
12. Contreras-Cruz, M.A., Ayala-Ramirez, V., Hernandez-Belmonte, U.H.: Mobile robot path planning using artificial bee colony and evolutionary programming. Appl. Soft Comput. **30**, 319–328 (2015)
13. Cui, R., Li, Y., Yan, W.: Mutual information-based multi-auv path planning for scalar field sampling using multidimensional rrt. IEEE Trans. Syst. Man Cybern.: Syst. **46**(7), 993–1004 (2016)
14. Das, P., Behera, H.S., Panigrahi, B.K.: A hybridization of an improved particle swarm optimization and gravitational search algorithm for multi-robot path planning. Swarm Evol. Comput. **28**, 14–28 (2016)
15. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**(1), 269–271 (1959)
16. Dolgov, D., Thrun, S., Montemerlo, M., Diebel, J.: Path planning for autonomous vehicles in unknown semi-structured environments. Int. J. Robot. Res. **29**(5), 485–501 (2010)
17. Doran, J.E., Michie, D.: Experiments with the graph traverser program. Proc. R. Soc. Lond. Ser.: A Mathematical and Physical Sciences **294**(1437), 235–259 (1966)
18. Ersson, T., Hu, X.: Path planning and navigation of mobile robots in unknown environments. In: IROS, pp. 858–864 (2001)
19. Hassanzadeh, I., Sadigh, S.M.: Path planning for a mobile robot using fuzzy logic controller tuned by ga. In: 6th International Symposium on Mechatronics and Its Applications, 2009. ISMA'09, pp. 1–5. IEEE (2009)
20. Hsu, D., Latombe, J.C., Motwani, R.: Path planning in expansive configuration spaces. In: 1997 IEEE International Conference on Robotics and Automation, 1997. Proceedings, vol. 3, pp. 2719–2726 (1997)
21. Janson, L., Schmerling, E., Clark, A., Pavone, M.: Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions. Int. J. Robot. Res. **34**(7), 883–921 (2015)
22. Jordan, R., Kinderlehrer, D., Otto, F.: The variational formulation of the fokker–planck equation. SIAM J. Math. Anal. **29**(1), 1–17 (1998)
23. Kamon, I., Rivlin, E.: Sensory-based motion planning with global proofs. IEEE Trans. Robot. Autom. **13**(6), 814–822 (1997)
24. Kamon, I., Rivlin, E., Rimon, E.: A new range-sensor based globally convergent navigation algorithm for mobile robots. In: Proceedings of IEEE International Conference on Robotics and Automation, vol. 1, pp. 429–435. IEEE (1996)
25. Kantarovich, L.: Mathematical methods in the organization and planning of production. Publication House of the Leningrad State University. (Translated in Management Sc., vol. 66, pp. 366–422) (1939)
26. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. **30**(7), 846–894 (2011)
27. Kavraki, L., Latombe, J.C.: Randomized preprocessing of configuration for fast path planning. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pp. 2138–2145. IEEE (1994)
28. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. In: Autonomous Robot Vehicles, pp. 396–404. Springer (1986)
29. Koenig, S., Likhachev, M.: Fast replanning for navigation in unknown terrain. IEEE Trans. Robot. **21**(3), 354–363 (2005)
30. Krishnan, V., Martínez, S.: Distributed optimal transport for the deployment of swarms. In: 2018 IEEE Conference on Decision and Control (CDC), pp. 4583–4588 (2018)
31. LaValle, S.M.: Rapidly-exploring random trees: a new tool for path planning (1998)
32. Lei, L., Wang, H., Wu, Q.: Improved genetic algorithms based path planning of mobile robot under dynamic unknown environment. In: 2006 International Conference on Mechatronics and Automation, pp. 1728–1732. IEEE (2006)
33. Li, W.: A study of stochastic differential equations and Fokker-Planck equations with applications. Ph.D. thesis, Georgia Institute of Technology (2016)
34. Li, W., Lu, J., Zhou, H., Chow, S.N.: Method of evolving junctions: a new approach to optimal control with constraints. Automatica **78**, 72–78 (2017)

35. Lumelsky, V., Stepanov, A.: Dynamic path planning for a mobile automaton with limited information on the environment. IEEE Trans. Autom. Control **31**(11), 1058–1063 (1986)

36. Lumelsky, V.J., Stepanov, A.A.: Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. Algorithmica **2**(1-4), 403–430 (1987)

37. Luo, C., Yang, S.X.: A bioinspired neural network for real-time concurrent map building and complete coverage robot navigation in unknown environments. IEEE Trans. Neural Netw. **19**(7), 1279–1298 (2008)

38. McGuire, K., de Croon, G., Tuyls, K.: A comparative study of bug algorithms for robot navigation. arXiv:1808.05050 (2018)

39. Montiel, O., Orozco-Rosas, U., Sepúlveda, R.: Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles. Expert Syst. Appl. **42**(12), 5177–5191 (2015)

40. Moore, E.F.: The shortest path through a maze. In: Proceedings of the International Symposium Switching Theory, pp. 285–292 (1959)

41. Ng, J., Bräunl, T.: Performance comparison of bug navigation algorithms. J. Intell. Robot. Syst. **50**(1), 73–84 (2007)

42. Noreen, I., Khan, A., Habib, Z.: Optimal path planning using rrt* based approaches: a survey and future directions. Int. J. Adv. Comput. Sci. Appl **7**, 97–107 (2016)

43. Otto, F.: The geometry of dissipative evolution equations: the porous medium equation (2001)

44. Overmars, M.H.: A random approach to motion planning, vol. 92. Unknown Publisher (1992)

45. Podsedkowski, L., Nowakowski, J., Idzikowski, M., Vizvary, I.: A new solution for path planning in partially known or unknown environment for nonholonomic mobile robots. Robot. Auton. Syst. **34**(2–3), 145–152 (2001)

46. Sfeir, J., Saad, M., Saliah-Hassane, H.: An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment. In: 2011 IEEE International Symposium On Robotic and Sensors Environments (ROSE), pp. 208–213 (2011)

47. Stentz, A.: Optimal and efficient path planning for partially-known environments. In: ICRA, vol. 94, pp. 3310–3317 (1994)

48. Stentz, A.: The focussed D^* algorithm for real-time replanning. In: IJCAI, vol. 95, pp. 1652–1659 (1995)

49. Svestka, P.: Robot motion planning using probabilistic roadmaps. PhD Thesis, Universiteit Utrecht (1997)

50. Tian, Y., Yan, L., Park, G.Y., Yang, S.H., Kim, Y.S., Lee, S.R., Lee, C.Y.: Application of Rrt-based local path planning algorithm in unknown environment. In: International Symposium on Computational Intelligence in Robotics and Automation. CIRA 2007, pp. 456–460. IEEE (2007)

51. Van Den Berg, J., Ferguson, D., Kuffner, J.: Anytime path planning and replanning in dynamic environments. In: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, pp. 2366–2371. IEEE (2006)

52. Villani, C.: Optimal Transport: Old and New, vol. 338. Springer Science & Business Media (2008)

53. Wagner, G., Choset, H.: Subdimensional expansion for multirobot path planning. Artif. Intell. **219**, 1–24 (2015)

54. Walker, M., Messom, C.H.: A comparison of genetic programming and genetic algorithms for auto-tuning mobile robot motion control. In: The First IEEE International Workshop on Electronic Design, Test and Applications, 2002. Proceedings, pp. 507–509 (2002)

55. Wang, M.: Liu: fuzzy logic based robot path planning in unknown environment. In: Proceedings of 2005 International Conference on Machine Learning and Cybernetics, 2005, vol. 2, pp. 813–818. IEEE (2005)

56. Yang, K., Keat Gan, S., Sukkarieh, S.: A gaussian process-based rrt planner for the exploration of an unknown and cluttered environment with a uav. Adv. Robot. **27**(6), 431–443 (2013)

57. Zuse, K.: Der plankalkül. 63. Gesellschaft für Mathematik und Datenverarbeitung (1972)

Haoyan Zhai received the B.S. degree in mathematics from Zhejiang University in 2014, and the Ph.D. in mathematics degree from Georgia Institute of Technology in 2019. His research interests include optimal transport, and stochastic differential equations with applications in robotics and machine learning.

Magnus Egerstedt is the Dean of Engineering in the Samueli School of Engineering at the University of California, Irvine, with a faculty appointment in the Department of Electrical Engineering and Computer Science. Egerstedt received the M.S. degree in Engineering Physics and the Ph.D. degree in Applied Mathematics from the Royal Institute of Technology, Stockholm, Sweden, the B.A. degree in Philosophy from Stockholm University, and spent a large part of his career as a faculty member at the Georgia Institute of Technology. Dr. Egerstedt conducts research in the areas of control theory and robotics, with particular focus on control and coordination of complex networks, such as multi-robot systems. Egerstedt is a Fellow of IEEE and IFAC, and is a Foreign member of the Royal Swedish Academy of Engineering Science. His teaching and research awards include the Ragazzini Award, the O. Hugo Schuck Best Paper Award, and the Alumni of the Year Award from the Royal Institute of Technology.

H.M. Zhou received the B.S. in pure mathematics from Peking University, the M. Phil. in applied mathematics from the Chinese University of Hong Kong, and the Ph.D in computational mathematics from University of California, Los Angeles. He was a postdoctoral scholar at California Institute of Technology for three years. He has been a faculty member in the School of mathematics at Georgia Institute of Technology since 2003. His research interests include optimal transport, inverse problems, and stochastic differential equations with applications in robotics and machine learning.