Quick Multi-Robot Motion Planning by Combining Sampling and Search

Keisuke Okumura[1,2*], **Xavier Défago**[3]
[1]National Institute of Advanced Industrial Science and Technology (AIST)
[2]University of Cambridge
[3]Tokyo Institute of Technology
ko393@cl.cam.ac.uk, defago@c.titech.ac.jp

Abstract

We propose a novel algorithm to solve multi-robot motion planning (MRMP) rapidly, called *Simultaneous Sampling-and-Search Planning (SSSP)*. Conventional MRMP studies mostly take the form of two-phase planning that constructs roadmaps and then finds inter-robot collision-free paths on those roadmaps. In contrast, SSSP simultaneously performs roadmap construction and collision-free pathfinding. This is realized by uniting techniques of single-robot sampling-based motion planning and search techniques of multi-agent pathfinding on discretized spaces. Doing so builds the small search space, leading to quick MRMP. SSSP ensures finding a solution eventually if exists. Our empirical evaluations in various scenarios demonstrate that SSSP significantly outperforms standard approaches to MRMP, i.e., solving more problem instances much faster. We also applied SSSP to planning for 32 ground robots in a dense situation.

1 Introduction

Solving a *multi-robot motion planning (MRMP)* problem within a realistic timeframe plays a crucial role in the modern and coming automation era, including fleet operations in warehouses [Wurman *et al.*, 2008] as well as collaborative robotic manipulation [Feng *et al.*, 2020]. Nevertheless, the problem is known to be tremendously challenging even with simple settings [Spirakis and Yap, 1984; Hopcroft *et al.*, 1984; Hearn and Demaine, 2005]. Filling this gap is a key milestone in automation.

Informally, MRMP aims at finding a collection of paths for multiple robots. Those paths must be obstacle-free, and also, inter-robot collision-free. Single-robot motion planning itself is intractable in general [Reif, 1979], furthermore, the specific difficulty of MRMP comes from the second condition. As a result, most studies on MRMP decouple the problem into *(i)* how to find obstacle-free paths and *(ii)* how to manage inter-robot collisions on those paths. This decoupling takes the form of *two-phase planning* that first constructs roadmaps and then performs *multi-agent search*, finding collision-free paths on those roadmaps. Here, a *roadmap* is a graph that approximates the workspace for one robot and is carefully constructed to include obstacle-free paths from initial to goal states. We can see two-phase planning examples in [Švestka and Overmars, 1998; Solovey *et al.*, 2016; Hönig *et al.*, 2018; Solis *et al.*, 2021], to name just a few.

Contribution. In contrast to approaches based on two-phase planning, we propose a novel MRMP algorithm that *simultaneously performs roadmap construction and multi-agent search*. The crux is developing robot-wise roadmaps as necessary according to the multi-agent search progress. Doing so keeps the search space small, leading to quick MRMP solving. The proposed algorithm, called *Simultaneous Sampling-and-Search Planning (SSSP)*, guarantees to eventually find a solution for solvable instances. Although SSSP outputs *sequential* solutions such that at most one robot moves at a time, it is possible to post-process known solutions to remove redundant motion and waiting time, as presented in this paper. Our extensive evaluations on various scenarios with diverse degrees of freedom and kinematic constraints demonstrate that SSSP significantly outperforms standard approaches to MRMP, i.e., solving more problem instances much faster. We also provide a planning demo with 32 ground robots (64-DOFs in total) in a dense situation.

1.1 Related Work

The (single-robot) motion planning problem, a fundamental problem in robotics, aims at finding an obstacle-free path in cluttered environments. An established approach is *sampling-based motion planning (SBMP)* [Elbanhawi and Simic, 2014]. SBMP iteratively and randomly samples state points from the space and then constructs a roadmap; a solution is derived by pathfinding on the roadmap. Numerous SBMP algorithms have been developed so far, such as [Kavraki *et al.*, 1996; LaValle, 1998; Karaman and Frazzoli, 2011]. SBMP can solve problems even with many degrees of freedom and has achieved successful results not limited to robotics [LaValle, 2006].

In principle, SBMP is applicable to MRMP by considering one composite robot consisting of all robots [Choset *et al.*, 2005], as seen in [Sánchez and Latombe, 2002; Le and Plaku, 2018]. However, such strategies require sampling from the high-dimensional space linear to the num-

[*]Most work done when Keisuke Okumura was a Ph.D. student at Tokyo Institute of Technology.

ber of robots, being a bottleneck even for SBMP [LaValle, 2006]. Consequently, recent studies mostly take the aforementioned two-phase planning. In those studies, as the first phase, roadmaps are explicitly prepared via conventional SBMP [Švestka and Overmars, 1998; Wagner *et al.*, 2012; Solovey *et al.*, 2016; Solis *et al.*, 2021; Dayan *et al.*, 2021] or implicitly embedded as lattice grids [Han *et al.*, 2018; Hönig *et al.*, 2018; Cohen *et al.*, 2019]. Depending on the heterogeneity of robots, a roadmap is shared among robots, or, robot-wise roadmaps are constructed. The lattice grids are available when the configuration space of each robot is not high-dimensional or state transitions of robots are restricted to a limited number; otherwise, the search space dramatically grows. This study does not assume such limitations. The second phase often uses *multi-agent pathfinding (MAPF)* algorithms such as [Sharon *et al.*, 2015; Barer *et al.*, 2014; Wagner and Choset, 2015], including prioritized planning [Erdmann and Lozano-Perez, 1987; Silver, 2005], and sometimes discretized versions of SBMP [Cáp *et al.*, 2013; Solovey *et al.*, 2016]. MAPF [Stern *et al.*, 2019] is a problem of finding a set of collision-free paths on a graph and collects extensive attention since the 2010s.

SSSP is directly inspired by two algorithms, respectively for SBMP and MAPF: *(i) Expansive Space Trees (EST)* [Hsu *et al.*, 1997] is an example of SBMP, which performs planning by constructing a query tree growing with random walks. *(ii) A* with operator decomposition* [Standley, 2010] solves MAPF efficiently by decomposing successors in the search tree such that at most one agent takes an action, rather than all agents take actions simultaneously. We realize rapid MRMP by combining these techniques.

1.2 Paper Organization

Section 2 describes the problem formulation and assumptions for MRMP. Section 3 describes SSSP. Section 4 presents the empirical results. Section 5 provides discussions. Throughout the paper, we present a sub-optimal algorithm and focus on the decision problem because solving MRMP itself is challenging. Optimal SSSP is discussed at the end. The appendix, code, and video are available at https://kei18.github.io/sssp/.

2 Preliminaries

2.1 Problem Definition of MRMP

We consider a problem of motion planning for a team of n robots $A = \{1, 2, \ldots, n\}$ in the 3D closed *workspace* $\mathcal{W} \subset \mathbb{R}^3$. Each robot i is operated in its own *configuration space* $\mathcal{C}_i \subset \mathbb{R}^{d_i^{\mathcal{C}}}$, where $d_i^{\mathcal{C}} \in \mathbb{N}_{>0}$. A set of points occupied by robot i at a configuration $q \in \mathcal{C}_i$ is denoted as $\mathcal{R}_i(q) \subset \mathcal{W}$. The space $\mathcal{W}$ may contain obstacles $\mathcal{O} \subset \mathcal{W}$. A free space for robot i is then $\mathcal{C}_i^{\text{free}} := \{q \in \mathcal{C}_i \mid \mathcal{R}_i(q) \cap \mathcal{O} = \emptyset\}$. A *trajectory* for robot i is defined by a continuous mapping $\sigma_i : \mathbb{R}_{\geq 0} \mapsto \mathcal{C}_i$.

Definition 1. *An* MRMP instance *is defined by a tuple* ($\mathcal{W}$, A, $\mathcal{C}$, $\mathcal{O}$, $\mathcal{R}$, $\mathcal{Q}^{init}$, $\mathcal{Q}^{goal}$)*, where* $\mathcal{C} := (\mathcal{C}_i)^{i \in A}$, $\mathcal{R} := (\mathcal{R}_i)^{i \in A}$, $\mathcal{Q}^{init} := \left(q_i^{init} \in \mathcal{C}_i\right)^{i \in A}$*, and* $\mathcal{Q}^{goal} := (Q_i^{goal} \subseteq \mathcal{C}_i)^{i \in A}$.

Definition 2. *Given an MRMP instance, the* MRMP problem *is to find a tuple of* n *trajectories* $(\sigma_i)^{i \in A}$ *(i.e., solution) and* $t_{end} \in \mathbb{R}_{\geq 0}$*, satisfying the following conditions:*

- endpoint*:* $\sigma_i(0) = q_i^{init} \wedge \sigma_i(t_{end}) \in Q_i^{goal}$
- obstacle-free*:* $\sigma_i(\tau) \in \mathcal{C}_i^{free},\ 0 \leq \tau \leq t_{end}$
- inter-robot collision-free*:* $\mathcal{R}_i\big(\sigma_i(\tau)\big) \cap \mathcal{R}_j\big(\sigma_j(\tau)\big) = \emptyset, i \neq j \in A, 0 \leq \tau \leq t_{end}$

2.2 Constraints of Robot Motions

Definition 2 assumes that a robot can go in any direction in the configuration space unless it encounters obstacles. We call it *geometric MRMP*. Meanwhile, robots are often subject to kinematic and dynamics constraints. *Kinematic constraints* restrict the local directions of motion available to a robot from a given configuration. For instance, wheeled robots cannot translate sideways. *Dynamics constraints* are governed by the time derivatives, such as velocity and acceleration. For instance, cars cannot stop instantly. Kinematic and dynamics constraints are collectively called *kinodynamic constraints*. Motion planning under kinodynamic constraints is called *kinodynamic planning* [Donald *et al.*, 1993]. Since this study is an early-stage attempt at combining sampling and search, *we consider only kinematic constraints and ignore dynamics constraints* for simplicity.

2.3 Planning with Kinematic Constraints

Consider a *control space* $\mathcal{U}_i \in \mathbb{R}^{d_i^{\mathcal{U}}}$ for robot i, where $d_i^{\mathcal{U}} \in \mathbb{N}_{>0}$. For instance, the control space of a wheeled robot is defined by motor controls of its wheels. Then, transitions of configurations under kinematic constraints are governed by $\dot{q} = f_i(q, u)$, where $q \in \mathcal{C}_i$ and $u \in \mathcal{U}_i$. A *kinematic MRMP instance* is defined by a composition of an MRMP instance in Def. 1, control spaces $(\mathcal{U}_i)^{i \in A}$, and transition functions $(f_i)^{i \in A}$. Given a kinematic MRMP instance, the *kinematic MRMP problem* asks a sequence of control inputs for each robot i, that is, $\xi_i : \mathbb{R}_{\geq 0} \mapsto \mathcal{U}_i$. A trajectory σ_i of configurations is successively defined as $\sigma_i(0) := q_i^{\text{init}}$ and $\sigma_i(t) := \int_0^t f_i\left(\sigma_i(\tau), \xi_i(\tau)\right) d\tau + \sigma_i(0)$. ξ_i constitutes a *solution* when σ_i satisfies the conditions in Def. 2 for all robots.

2.4 Discretized Time and Local Planner

Both geometric and kinematic MRMPs are defined in continuous time. However, it is realistic for planning to discretize the time. That is, by introducing $\Delta \in \mathbb{R}_{>0}$ as a small amount of time, we aim at finding a path of configurations for each robot, such that any consecutive two points are travelable in Δ, without encountering obstacles, without inter-robot collisions, and following kinematic constraints. For instance, given a sequence of control inputs $\xi^0, \xi^1, \ldots$, a path of configurations in kinematic MRMP is successively defined as $\sigma_i^0 := q_i^{\text{init}}$ and $\sigma_i^k := \sigma_i^{k-1} + f_i\big(\sigma_i^{k-1}, \xi^{k-1}\big)\Delta$. Herein, δ can be regarded as a problem input shared between robots.

For this discretization, as often assumed in motion planning studies [Choset *et al.*, 2005; LaValle, 2006], we assume that each robot i has a *local planner* denoted as $\mathsf{connect}_i$. Given two configurations $q^{\text{from}}, q^{\text{to}} \in \mathcal{C}_i$, this function returns a unique trajectory σ of the duration Δ that satisfies: *(i)* $\sigma(0) = q^{\text{from}} \wedge \sigma(\Delta) = q^{\text{to}}$, *(ii)* $\sigma(\tau) \in \mathcal{C}_i^{\text{free}}$ for $0 \leq \tau \leq$

Δ, and *(iii)* σ follows f_i of kinematic MRMP. If no such σ is found, $\mathsf{connect}_i$ returns $\perp$. For instance, $\mathsf{connect}_i(q^{\text{from}}, q^{\text{to}})$ may output $(\Delta - \tau)q^{\text{from}} + \tau q^{\text{to}}$ for geometric MRMP, or Dubins paths [Dubins, 1957] for car-like robots. Since kinematic MRMP excludes dynamics constraints, this paper assumes that each robot can always remain in its current configuration, i.e., $\mathsf{connect}_i(q, q) \neq \perp$ for any $q \in \mathcal{C}_i^{\text{free}}$.

Observe that the local planners *hide* control inputs of kinematic MRMP; as long as they are definable, we can directly consider planning in configuration spaces. Therefore, we collectively call geometric and kinematic MRMPs the *MRMP problem* and do not distinguish the two explicitly. With the local planners, a *solution* of MRMP is a tuple of paths $(\Pi_i)^{i \in A}$, where $\Pi_i = (q_0, q_1, \dots, q_k \mid q_t \in \mathcal{C}_i)$ and k is common between robots, satisfying the following conditions:

- endpoint: $\Pi_i[0] = q_i^{\text{init}} \land \Pi_i[k] \in \mathcal{Q}_i^{\text{goal}}$
- consistent path: $\mathsf{connect}_i(\Pi_i[t], \Pi_i[t+1]) \neq \perp$
- inter-robot collision-free:

$$\begin{aligned} &\mathcal{R}_i\big(\sigma_i(\tau)\big) \cap \mathcal{R}_j\big(\sigma_j(\tau)\big) = \emptyset,\ 0 \le \tau \le \Delta, i \neq j \in A, \\ &\sigma_{\{i,j\}} = \mathsf{connect}_{\{i,j\}}(\Pi_{\{i,j\}}[t], \Pi_{\{i,j\}}[t+1]) \end{aligned} \tag{1}$$

The domain of t is $\{0, 1, \dots, k-1\}$.

2.5 Roadmap

A *roadmap* for robot i is a directed graph $G_i = (V_i, E_i)$ which approximates $\mathcal{C}_i^{\text{free}}$ with a finite set of vertices. Each vertex $q \in V_i$ corresponds to a configuration of $\mathcal{C}_i$, thus simply denoted as $q \in \mathcal{C}_i$. The roadmap G_i must satisfy $q \in \mathcal{C}_i^{\text{free}}$ for all $q \in V_i$ and $\mathsf{connect}_i(q, q') \neq \perp$ for all $(q, q') \in E_i$.

2.6 Blackbox Utility Functions

To solve MRMP, we introduce four functions. The first three are common in motion planning studies [Choset *et al.*, 2005; LaValle, 2006], whereas the last one is specific to MRMP.

Sampling. The function sample_i randomly samples a configuration $q \in \mathcal{C}_i$, where q may not be in $\mathcal{C}_i^{\text{free}}$.

Distance. The function $\mathsf{dist}_i : \mathcal{C}_i \times \mathcal{C}_i \mapsto \mathbb{R}_{\ge 0}$ defines a distance of two configurations of robot i. It is not necessary for this function to consider obstacles or inter-robot collisions, however, we assume that $p = q \Leftrightarrow \mathsf{dist}_i(p, q) = 0$, $\mathsf{dist}(p, q) = \mathsf{dist}(q, p)$, and dist_i satisfies the triangle inequality, e.g., the Euclidean distance.

Steering. The function steer_i takes two configurations q^{from}, $q^{\text{to}} \in \mathcal{C}_i^{\text{free}}$ and then returns a "closer" configuration $q \in \mathcal{C}_i^{\text{free}}$ to q^{to}. With a prespecified parameter $\epsilon > 0$, formally:

$$\mathsf{steer}_i\left(q^{\text{from}}, q^{\text{to}}\right) := \operatorname*{argmin}_{q \in U_\epsilon} \mathsf{dist}_i\left(q, q^{\text{to}}\right) \text{ where}$$
$$U_\epsilon := \left\{q \in \mathcal{C}_i^{\text{free}} \mid \mathsf{dist}_i\left(q^{\text{from}}, q\right) \le \epsilon, \mathsf{connect}_i\left(q^{\text{from}}, q\right) \neq \perp\right\}$$

In practice, steer can be approximately implemented by binary search, e.g., starting from q^{to}, repeatedly sampling $q \in \mathcal{C}_i$ while halving the distance from q^{from} until q is feasible.

Collision. For two robots i, j, given four configurations $q_{\{i,j\}}^{\text{from}}$, $q_{\{i,j\}}^{\text{to}} \in \mathcal{C}_{\{i,j\}}^{\text{free}}$ such that $\mathsf{connect}_{\{i,j\}}(q_{\{i,j\}}^{\text{from}}, q_{\{i,j\}}^{\text{to}}) \neq \perp$, the function $\mathsf{collide}_{(i,j)}(q_i^{\text{from}}, q_i^{\text{to}}, q_j^{\text{from}}, q_j^{\text{to}})$ returns `TRUE` when there is a collision if two robots simultaneously change their configurations from $q_{\{i,j\}}^{\text{from}}$ to $q_{\{i,j\}}^{\text{to}}$. Here, a *collision* is defined similarly to Eq. (1). The function otherwise returns `FALSE`. For convenience, we use $\mathsf{collide}(\mathcal{Q}^{\text{from}}, \mathcal{Q}^{\text{to}})$, where $\mathcal{Q}^{\{\text{from, to}\}} = (q_i^{\{\text{from, to}\}} \in \mathcal{C}_i^{\text{free}})^{i \in A}$. This shorthand notation returns `TRUE` if and only if there is a pair (i, j) for which $\mathsf{collide}_{(i,j)}(q_i^{\text{from}}, q_i^{\text{to}}, q_j^{\text{from}}, q_j^{\text{to}})$ returns `TRUE`. As for implementation, since collision detection is important in single-robot motion planning and has been studied for a long time, well-known open-source libraries are available, e.g., [Pan *et al.*, 2012].

Domain Independence. To sum up, we solve MRMP using only five blackbox functions: connect, sample, dist, steer, and collide. Doing so makes our approach non-restrictive to specific robotic systems, rather it is applicable to many planning domains as we will see in the experiments.

3 Algorithm Description

In a nutshell, SSSP performs a best-first search using operator decomposition [Standley, 2010] while simultaneously growing robot-wise roadmaps via random walks [Hsu *et al.*, 1997]. This section first explains the core idea, followed by the pseudocode, theoretical analysis of completeness, and postprocessing to obtain better solutions.

3.1 Core Idea

As a high-level description, SSSP constructs a search tree while expanding robot-wise roadmaps. Each search node in the tree contains a tuple of configurations $\mathcal{Q} = (q_i \in \mathcal{C}_i^{\text{free}})^{i \in A}$ and robot i that will take the next action. When this node is selected during the search process, the algorithm does *vertex expansion* and *search node expansion* in order. The former expands the roadmap G_i for robot i by random walks from $\mathcal{Q}[i]$. The latter creates the successors of the node by transiting the configuration of robot i from $\mathcal{Q}[i]$ to its neighboring configurations on G_i, and then passing the turn for i to $i + 1$. Figure 1 illustrates this procedure with initial roadmap construction explained later, while Fig. 2 shows an example of constructed roadmaps.

3.2 Details

Algorithm 1 presents the pseudocode of SSSP. Some artifacts are explained as follows.

Initial Roadmap Construction (Line 1) is done by conventional single-agent SBMP such as RRT-Connect [Kuffner and LaValle, 2000]. The objective is to secure at least one valid path from initial to goal configurations for each robot.

Best-first Search (Lines 3–22) realizes the core idea by maintaining a priority queue *Open* that stores generated nodes and a set *Explored* that stores already generated search situations. For each iteration, SSSP checks whether the popped node from *Open* satisfies the goal condition (Line 8). If so, it returns a solution by backtracking the node.

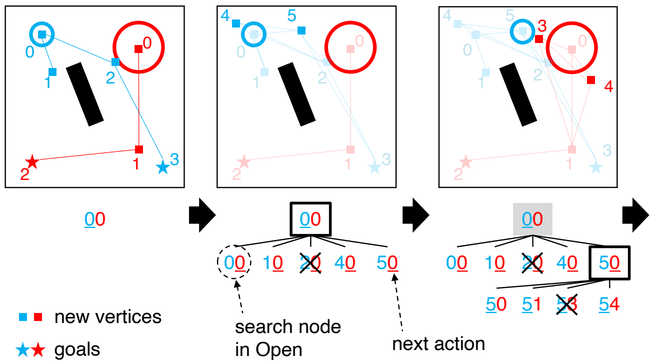

Figure 1: Illustration of SSSP. The algorithm progresses from left to right. *top:* Robot-wise roadmaps and search situations. Two robots are shown by colored circles and an obstacle by a black rectangle. *bottom:* Search trees. A node ‘xy’ corresponds to a situation where the blue and red robots are respectively at vertices ‘x’ and ‘y,’ and the blue robot will take the next action. *left:* Initial roadmaps. The search starts from the node ‘00.’ *middle:* Vertex expansion and search node expansion for the blue robot. The node ‘20’ is not generated due to inter-robot collision. *right:* The red robot’s turn. The search continues until all robots reach their goals.

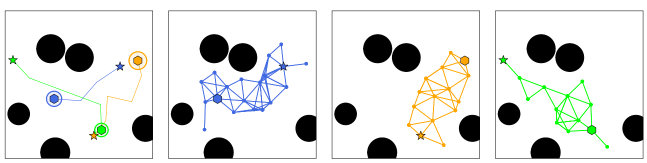

Figure 2: Example of constructed roadmaps for 2D circular robots (i.e., *Point2d* in Fig. 4). A solution is depicted at the leftmost. The others show respective roadmaps for each robot.

Vertex Expansion (Lines 11–15) is implemented by steering from the target configuration to newly sampled ones, up to a fixed number $m \in \mathbb{N}_{>0}$. To guarantee completeness, SSSP also uses vanilla random sampling with a small probability λ (0.01 in our experiments). Each new vertex must satisfy a constraint of distance threshold, which suppresses roadmaps being too dense; otherwise, the search space dramatically increases, making it difficult to find a solution. Similar techniques are seen in SBMP studies [Kala, 2013; Dobson and Bekris, 2014]. Each new vertex q^{new} is followed by edge updates, that is, connecting q^{new} to all vertices that connect returns trajectories. Note that, in practice, many sampling trials (i.e., large m) may compromise computation time.

Search Node Expansion (Lines 16–22) adds successors that *(i)* have not appeared yet in the search process and *(ii)* do not collide with other robots.

Search Iteration (Lines 2–23). The search iterates until a solution is found while decreasing the distance thresholds for vertex expansion by multiplying $0 < \gamma < 1$ by current ones.

Search Node Scoring. The heart of the best-first search is how to score each node that determines which node is popped from Open (Line 7). Given a tuple of configurations $\mathcal{Q}$, SSSP scores the node by summation over each robot i’s shortest path distance from $\mathcal{Q}[i]$ to one of the configurations in $\mathcal{Q}^{\text{goal}}_i$ on the roadmap G_i, while weighting each edge $(q^{\text{from}}, q^{\text{to}}) \in E_i$ by $\mathsf{dist}(q^{\text{from}}, q^{\text{to}})$. The path distance is calculated by ignoring inter-robot collisions, as common in heuristics for MAPF studies [Silver, 2005].

Algorithm 1 SSSP; **input**: instance I; **output**: solution

params: #sampling $m \in \mathbb{N}_{>0}$
random sampling prob. $\lambda \in (0,1]$
threshold distances $\theta_i \in \mathbb{R}_{>0}$, decay rate $\gamma \in (0,1)$

1: $G_i = (V_i, E_i) \leftarrow$ init_roadmap(I, i); for each $i \in A$
2: **while** TRUE **do**
3: initialize Open, $\mathit{Explored}$ ▷ *priority queue, set*
4: Open.push $(\langle \mathcal{Q} : \mathcal{Q}^{\text{init}},\ \mathit{next} : 1,\ \mathit{parent} : \bot \rangle)$
5: $\mathit{Explored}$.append $((\mathcal{Q}^{\text{init}}, 1))$
6: **while** $\mathit{Open} \neq \emptyset$ **do**
7: $\mathcal{N} \leftarrow \mathit{Open}$.pop()
8: **if** $\forall i \in A, \mathcal{N}.\mathcal{Q}[i] \in \mathcal{Q}^{\text{goal}}_i$ **then**
9: **return** backtrack$(\mathcal{N})$
10: $i \leftarrow \mathcal{N}.\mathit{next}$; $q^{\text{from}} \leftarrow \mathcal{N}.\mathcal{Q}^{\text{from}}[i]$
11: **for** $1, 2, \ldots, m$ **do** ▷ *vertex expansion via sampling*
12: $q^{\text{new}} \leftarrow$ sample$_i$()
13: with prob. $(1-\lambda)$: $q^{\text{new}} \leftarrow$ steer$_i(q^{\text{from}}, q^{\text{new}})$
14: **if** $\min_{q \in V_i}$ dist$_i$ $(q, q^{\text{new}}) > \theta_i$ **then**
15: $V_i \leftarrow V_i \cup \{q^{\text{new}}\}$; update E_i
16: $j \leftarrow i+1$ **if** $i \neq |A|$ **else** 1 ▷ *node expansion*
17: **for** $q^{\text{to}} \in \{q \in V_i \mid (q^{\text{from}}, q) \in E_i\}$ **do**
18: $\mathcal{Q}^{\text{new}} \leftarrow$ copy$(\mathcal{N}.\mathcal{Q})$; $\mathcal{Q}^{\text{new}}[i] \leftarrow q^{\text{to}}$
19: **if** $(\mathcal{Q}^{\text{new}}, j) \in \mathit{Explored}$ **then continue**
20: **if** collide $(\mathcal{N}.\mathcal{Q}, \mathcal{Q}^{\text{new}})$ **then continue**
21: Open.push $(\langle \mathcal{Q} : \mathcal{Q}^{\text{new}}, \mathit{next} : j, \mathit{parent} : \mathcal{N} \rangle)$
22: $\mathit{Explored}$.append $((\mathcal{Q}^{\text{new}}, j))$
23: $\theta_i \leftarrow \gamma\theta_i$; for each $i \in A$

3.3 Properties

Let $k \in \mathbb{N}_{>0}$ be the number of search iterations of Lines 2–23. Moreover, let G^k_i be the roadmap G_i at the beginning of k-th iteration (i.e., G_i at Line 3) and $G^k := (G^k_i)^{i \in A}$. A solution is called *sequential* when at most one robot transits configurations simultaneously at any time.

Lemma 3. *SSSP finds a solution in the k-th search iteration if G^k contains a sequential solution.*

Proof. For each k-th iteration, G_i is not infinitely increasing due to the distance threshold θ_i. Thus, the search space is finite: $O(|V_1| \cdot \ldots \cdot |V_n| \cdot |A|)$. Thanks to brute-force search in finite space, SSSP finds a solution if G^k contains one. □

Theorem 4. *For the geometric MRMP problem (Def. 2), the probability that SSSP finds a solution approaches one as k approaches ∞, provided the instance is solvable.*

Proof. We limit the discussion to geometric MRMP wherein each robot can move in arbitrary directions in its configuration space. The proof is based on analysis in [Švestka and Overmars, 1998], which claims if an instance is solvable, *(i)* there is a sequential solution and *(ii)* sufficiently dense

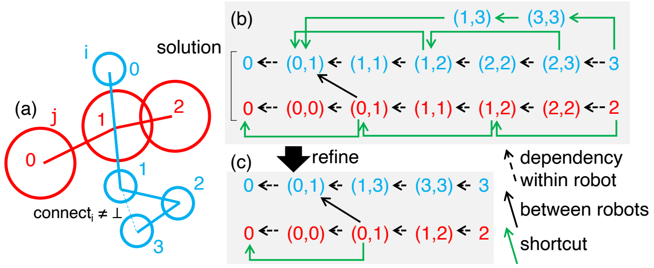

Figure 3: Postprocessing to refine a solution.

robot-wise roadmaps constructed uniformly at random sampling (e.g., PRM [Kavraki *et al.*, 1996]) contain a sequential solution. According to Lemma 3, SSSP finds a solution once roadmaps holding claim-(ii) are obtained. SSSP eventually constructs such roadmaps for the following reasons.

For each search iteration, each robot i tries to develop its roadmap with at least $m \geq 1$ new samples. With the probability $1 - \lambda > 0$, some of them are outcomes of uniformly at random sampling. Each iteration terminates in finite time (see Lemma 3), therefore, each robot does not stop attempts of uniformly at random sampling until finding solutions. Moreover, each iteration decreases the distance threshold, enabling robots to construct denser roadmaps. □

In short, Thm. 4 states that SSSP eventually finds a solution. This corresponds to *probabilistic completeness* in SBMP [Elbanhawi and Simic, 2014], defined by the probability of finding solutions bounded by the number of sampling.

3.4 Postprocessing

SSSP returns only sequential solutions and compromises solution quality such as the maximum traveling time. Thus, we briefly discuss how to realize parallel execution that enables two or more robots to move simultaneously. Specifically, we consider postprocessing to refine solutions. Smoothing solution trajectories by postprocessing is common in single-robot SBMP [Geraerts and Overmars, 2007]. However, MRMP additionally takes care of inter-robot collisions.

We describe our method with Fig. 3a. Suppose that SSSP outputs a sequential solution of $\Pi_i = (0, 1, 1, 2, 2, 3)$ (blue) and $\Pi_j = (0, 0, 1, 1, 2, 2)$ (red). The method repeats the next two steps until a given solution metric has not improved.

1. Construct a *temporal plan graph (TPG)* [Hönig *et al.*, 2016] of the solution. TPG is a directed acyclic graph that records temporal dependencies of each robot's motions. We also attach possible "shortcut" motions to TPG. Figure 3b shows an example. There is an arc between the motions $(0, 1)$ of robots $\{i, j\}$; these motions must happen in order due to collision avoidance. Several shortcut arcs exist, for example, i can skip using vertex-2 by directly going from vertex-1 to vertex-3.
2. Remove redundant motions in TPG while keeping the dependencies between robots. Figure 3c shows an example. For j, the motions $(1, 1)$ and $(2, 2)$ are removed but $(0, 0)$ survives to keep the dependency with i.

In the example, we finally obtain a refined solution $\Pi_i = (0, 1, 3, 3)$ and $\Pi_j = (0, 0, 1, 2)$.

The above refinement is applicable to any MRMP solutions not limited to those from SSSP. Indeed, the experiments applied the refinement for solutions obtained by all methods.

4 Evaluation

This section extensively evaluates SSSP on a variety of MRMP problems and demonstrates that it can solve various MRMP rapidly compared to other standard approaches. We further assess solution quality, scalability about the number of robots $|A|$, and which components are essential for SSSP, followed by a ground-robot demo in a dense situation.

4.1 Experimental Setups

Benchmarks. As illustrated in Fig. 4, we prepared various scenarios with diverse degrees of freedom and kinematic constraints, in closed workspaces $\mathcal{W} \in [0, 1]^{\{2,3\}}$. To focus on characteristics specific to MRMP, we modeled these scenarios with simple geometric patterns (e.g., spheres or lines) and reduced the effort of the connect and collide functions; these functions were performed with simple geometry calculations. For each scenario, we prepared 100 instances by randomly generating initial/goal configurations and obstacle layouts. For each instance, the number of robots $|A|$ was chosen from the interval $\{2, 3, \ldots, 10\}$. Robots' body parameters (e.g., radius and arm length) were also generated randomly and differed between robots. These parameters were adjusted so that robots are sufficiently congested, otherwise, the instances become easy to solve. Note that unsolvable instances may be included, though we excluded obviously unsolvable instances such as initial configurations with inter-robot collisions. In summary, each instance consists of a team of heterogeneous robots and each robot has a different configuration space; a shared roadmap is unavailable.

Baselines. Our goal is to develop planning algorithms that can be applied to various domains without the use of external knowledge other than five black-box functions. To this end, we carefully selected the following well-known baseline methods that are applicable to MRMP defined in Sec. 2. [1]

- **Probabilistic roadmap (PRM)** [Kavraki *et al.*, 1996] is a celebrated SBMP. For MRMP, PRM samples a composite configuration of $|A|$ robots directly from $O(|A|)$ dimensional spaces and constructs a single roadmap, and then derives a solution by pathfinding on it.
- **Rapidly-exploring Random Tree (RRT)** [LaValle, 1998] is another popular SBMP, focusing on single-query situations. Similar to PRM, RRT for MRMP samples a composite state and constructs a tree roadmap rooted in a composite one of initial configurations for all robots.
- **RRT-Connect (RRT-C)** [Kuffner and LaValle, 2000] is a popular extension of RRT, which accelerates finding a solution by bi-directional search from both initial and goal configurations.

[1] dRRT$^{(*)}$ [Solovey *et al.*, 2016; Shome *et al.*, 2020] was not included due to requiring an additional oracle.

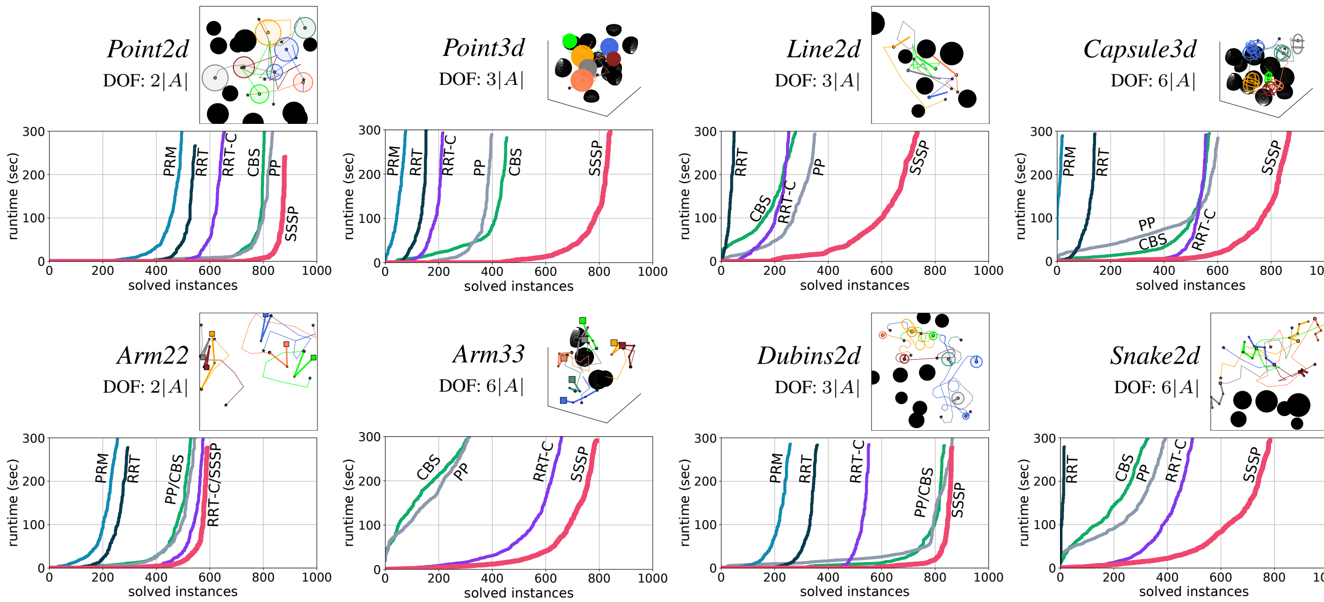

Figure 4: Summary of results. Each scenario is visualized with robots' bodies (colored circles, spheres, or lines), obstacles (black-filled circles or spheres), and a solution example (thin lines). Degrees of freedom (DOF) are denoted below scenario names. In *Arm*{*22,33*}, each robot has a fixed root, represented as colored boxes. These two and *Snake2d* prohibit self-colliding. In *Dubins2d*, robots must follow Dubins paths [Dubins, 1957]. The runtime scores of PP, CBS, and SSSP include the initial roadmap construction.

- **Prioritized Planning (PP)** [Erdmann and Lozano-Perez, 1987; Silver, 2005; Van Den Berg and Overmars, 2005] is a standard approach to MAPF such that robots sequentially plan paths while avoiding collisions with already planned paths. We applied PP to roadmaps constructed by robot-wise PRMs, as taken in [Le and Plaku, 2018]. PP was repeated with random priorities until the problem is solved.
- **Conflict-Based Search (CBS)** [Sharon *et al.*, 2015] is another popular MAPF algorithm. CBS is applicable to MRMP when roadmaps are given [Solis *et al.*, 2021]. We run CBS on robot-wise roadmaps constructed by PRM. Moreover, we manipulated the heuristic of CBS to avoid collisions as much as possible during the search. Doing so loses the optimality of CBS but speeds up finding solutions significantly [Barer *et al.*, 2014].

All hyperparameters of each method including SSSP were adjusted prior to the experiments (see the appendix). SSSP used RRT-Connect [Kuffner and LaValle, 2000] to obtain initial robot-wise roadmaps (Line 1). Note that this is irrelevant to RRT-C in the baselines. PP/CBS were tested with PRM rather than RRT-Connect because otherwise constructed roadmaps do not include detours, which is essential for solving MAPF in the second phase of two-phase planning. Since all methods rely on non-determinism, we tested each method with 10 different random seeds for each instance (1,000 trials in total).

Metrics. The objective is to find solutions as quickly as possible. Therefore, we rate **how many instances are solved within given time limits** (maximum: 5 min).

Evaluation Environment. The simulator and all methods were coded in Julia. The experiments were run on a desktop PC with Intel Core i9-7960X 2.8 GHz CPU and 64 GB RAM. A maximum of 32 different instances were run in parallel using multi-threading. All methods used exactly the same implementations of connect, collide, sample, and dist.

4.2 Results of Various MRMP Problems

Figure 4 summarizes the results. In short, SSSP outperforms the other baselines in all tested scenarios, i.e., solving more instances much faster. We acknowledge that runtime performance heavily relies on implementations; however, these results indicate that SSSP is very promising. The results of SSSP in *Arm22* and *Dubins2d* are relatively non-remarkable but we guess this is due to many unsolvable instances, which could be easily generated in these scenarios. We later discuss why SSSP is quick.

Solution Quality. As reference records of solution quality, Table 1 shows the expected **total traveling time** of all robots (aka. sum-of-costs), after applying the postprocessing introduced in Sec. 3.4 to all methods. Compared to the other baselines, the total traveling time of SSSP is not the best but comparable. We will discuss optimality at the end of the paper.

4.3 Scalability Test

We next assess the scalability of SSSP about the number of robots $|A|$, varied by 10 increments. For each $|A|$, 100 *Point2d* instances were prepared with smaller robots' radius (see Fig. 5). PP with tuned parameters (see the appendix) was also tested as a baseline, which relatively scored high among the other baselines in the scenario with many robots. Figure 5 shows that, with larger $|A|$, SSSP takes longer but still acceptable time for planning, compared to the baseline.

	SSSP	PRM	RRT	RRT-C	PP	CBS
Point2d 54/100	1.93 (1.75, 2.10)	3.67 (2.95, 4.25)	3.42 (3.02, 3.81)	3.04 (2.62, 3.43)	**1.45** (1.37, 1.52)	**1.43** (1.36, 1.49)
Point3d 26/100	2.88 (2.56, 3.16)	N/A	N/A	4.58 (4.14, 4.99)	**1.91** (1.73, 2.06)	**1.72** (1.59, 1.84)
Line2d 34/100	2.68 (2.42, 2.92)	N/A	N/A	5.89 (5.14, 6.62)	1.80 (1.69, 1.90)	**1.62** (1.56, 1.69)
Capsule3d 61/100	2.34 (2.12, 2.53)	N/A	N/A	2.89 (2.55, 3.23)	**1.84** (1.79, 1.90)	2.33 (2.24, 2.43)
Arm22 30/100	**1.57** (1.35, 1.75)	2.46 (2.13, 2.78)	2.23 (1.96, 2.48)	**1.73** (1.43, 2.00)	**1.60** (1.43, 1.73)	**1.51** (1.35, 1.64)
Arm33 94/100	**2.31** (2.22, 2.40)	N/A	N/A	2.97 (2.72, 3.21)	2.74 (2.64, 2.83)	2.75 (2.68, 2.82)
Dubins2d 30/100	1.52 (1.42, 1.62)	3.84 (3.31, 4.32)	3.06 (2.78, 3.33)	2.07 (1.80, 2.32)	1.28 (1.23, 1.32)	1.39 (1.34, 1.45)
Snake2d 55/100	**3.17** (2.91, 3.42)	N/A	N/A	4.62 (4.20, 5.02)	**3.28** (3.08, 3.48)	**3.25** (3.04, 3.44)

Table 1: Total traveling time. Scores are averaged over instances successfully solved by all methods, and normalized by $\sum_{i \in A} \mathsf{dist}(q_i^{\text{init}}, q \in \mathcal{Q}_i^{\text{goal}})$. The numbers below on the scenario names are those numbers of instances. To obtain meaningful values, some cases are excluded from the calculation due to the low success rates ($\leq$20%; denoted as N/A). We also show 95% confidence intervals of means. Bold characters are based on overlaps of the intervals.

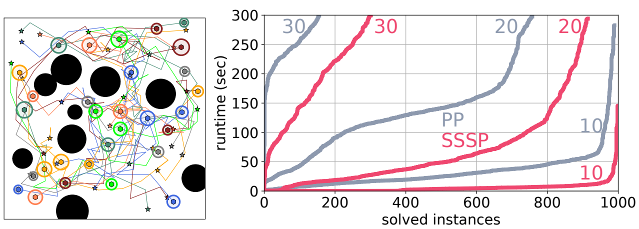

Figure 5: Scalability test in *Point2d*. *left*: An example instance ($|A| = 30$) and its solution from SSSP. *right*: The number of solved instances with annotation of $|A|$.

4.4 Which Elements are Essential?

We next address another question that asks which technical components are essential to SSSP. Specifically, we evaluated degraded versions that omit the following components: *(i)* initial roadmap constructions (Line 1), *(ii)* search node scoring (replaced by random values), *(iii)* vertex expansion (Lines 11–15) *(iv)* distance thresholds check (Line 14), *(v)* steering (by setting $\lambda = 1$), and *(vi)* integrated sampling and search. The last one rated SSSP without vertex expansion ($m = 0$) on robot-wise PRMs.

Table 2 reveals that all these components contribute to the performance of SSSP. Among them, involving the appropriate node scoring is particularly critical to achieving high success rates within a limited time, as well as vertex expansion. The initial roadmap construction is effective when single-robot motion planning itself is difficult (*Snake2d*). The steering effect was non-dramatic because the workspace is small relative to robots; we discuss this in the appendix, together with scores of total traveling time.

4.5 Robot Demo

We applied SSSP to 32 robots (https://toio.io/) modeled as *Point2d* in a dense situation (Fig. 6). The robots evolve on a specific playmat and are controllable by instructions of absolute coordinates. Even though an experimenter randomly placed the robots as the initial states (see the movie), the planning was done in about 30 s, and then all robots eventually reached their goal. Importantly, this demo was based on only five utility functions of connect, sample, dist, steer, and collide, without any prepared environmental representations.

	SSSP	random score	no init roadmap	no vertex expansion	no dist check	λ=1	on PRM
Point2d	**880**	512	793	498	737	875	807
Arm22	**586**	101	565	388	523	584	398
Snake2d	**710**	0	379	674	519	677	39

Table 2: Ablation study. The numbers of instances solved within 5 min are shown.

Figure 6: Robot demo. From left to right, the pictures show the initial, intermediate, and final configurations, which eventually constitute the characters 'SSSP.'

5 Conclusion and Discussion

This paper introduced the SSSP algorithm that rapidly solves MRMP. The main idea behind the algorithm was to unite techniques developed for SBMP and search techniques for MAPF. The former had been studied mainly in the robotics community, while the latter in the AI community. Bringing them together brings promising results, as extensively demonstrated in our experiments. In the remainder, we provide further discussions and future directions.

Why is SSSP Quick? We provide two qualitative explanations: one from the MAPF side, and another from the SBMP side. The quickness of SSSP relies on both factors.

Branching Factor: During the search, SSSP decomposes successors into search nodes corresponding to at most one robot taking a motion. Compared to search styles allowing all robots to move simultaneously, the decomposition significantly reduces *branching factor*, i.e., the number of successors at each node. In general, the average branching factor b largely determines the search effort [Edelkamp and Schrodl, 2011]. Assume that each robot has k possible motions from each configuration on average. Coupled with perfect heuristics and a perfect tie-breaking strategy, allowing all robots to move simultaneously results in $b = k^{|A|}$ and generates $\left(k^{|A|}\right) \cdot l$ nodes, where l is the depth of the search. In contrast, SSSP results in $b = k$ and enables searching the equivalent node with only $(k|A|) \cdot l$ nodes generation. This is a key trick of A* with operator decomposition [Standley, 2010] for MAPF; we based this idea to develop SSSP.

Imbalanced Roadmaps: PRM-based methods (i.e., PRM, PP, and CBS in our experiments) have no choice other than to construct roadmaps uniformly spread in each configuration space, making search spaces huge. Such drawbacks might be relieved with biased sampling but representing good bias for MRMP is not trivial; indeed, existing studies use machine learning [Arias *et al.*, 2021; Okumura *et al.*, 2022]. In contrast, owing to carefully-designed components, SSSP naturally constructs sparse roadmaps in important regions for each robot as seen in Fig. 2. Consequently, the search space for SSSP is kept small, which also contributes to quick MRMP.

Kinodynamic MRMP. We untreated dynamics constraints, therefore, *kinodynamic MRMP* is an interesting direction. Similarly to RRT [LaValle and Kuffner Jr, 2001] or EST [Hsu *et al.*, 1997] that are applicable to kinodynamic planning, we consider that SSSP is also applicable to such planning. The adaptation is by considering planning with a *state* x instead of a configuration $q \in \mathcal{C}_i$ for robot i, which comprises a configuration q and its derivative $\dot{q}$ (i.e., $x := (q, \dot{q})$). Some parts require care; in usual kinodynamic MRMP, $\mathsf{connect}_i(x, x) = \bot$ as we see in cars that cannot stop instantly. This means that sequential solutions are not allowed. In this case, it is necessary to regard $|A|$ successive search nodes in SSSP as "one block" to enable concurrent motions of multiple robots.

Optimal MRMP. SSSP prioritizes solving MRMP itself, rather than solution quality. However, it is possible to reflect quality by modifying node scoring, which is currently designed as a greedy search. With terminologies of A* search [Hart *et al.*, 1968], SSSP only uses h-value (i.e., estimation of cost-to-go). A promising direction is to incorporate g-value (i.e., cost-to-come), letting SSSP asymptotically optimal without degrading the performance of solvability. We also point out that, to be optimal, rewiring the search tree is mandatory as seen in SBMP or MAPF studies [Karaman and Frazzoli, 2011; Okumura, 2023a]. Another possibility is to incorporate iterative refinement schemes [Okumura *et al.*, 2021; Li *et al.*, 2021] developed for MAPF, although there is no theoretical guarantee.

Further Integration of SBMP and MAPF. The concept behind the paper was developing robot-wise roadmaps according to the multi-agent search progress, turning out to be promising. We consider this direction should be further investigated. A very-recent study [Kottinger *et al.*, 2022] explores this direction for CBS. Many powerful MAPF algorithms exist not limited to A* adaptation or CBS, such as [Li *et al.*, 2022; Okumura, 2023b]. Therefore, integrating them with SBMP may fruit practical methodologies for MRMP.

Acknowledgments

We thank François Bonnet and the anonymous reviewers for their many insightful comments on the manuscript. This work was partly supported by JSPS KAKENHI Grant Number 20J23011, 21K11748, 21H03423, 20K11685 and JST ACT-X Grant Number JPMJAX22A1. Keisuke Okumura thanks the support of the Yoshida Scholarship Foundation when he was a Ph.D. student.

References

[Arias *et al.*, 2021] Felipe Felix Arias, Brian Ichter, Aleksandra Faust, and Nancy M Amato. Avoidance critical probabilistic roadmaps for motion planning in dynamic environments. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[Barer *et al.*, 2014] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proceedings of Annual Symposium on Combinatorial Search (SOCS)*, 2014.

[Cáp *et al.*, 2013] Michal Cáp, P. Novák, J. Vokrínek, and M. Pechoucek. Multi-agent rrt: Sampling-based cooperative pathfinding. In *Proceedings of International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, 2013.

[Choset *et al.*, 2005] Howie Choset, Kevin M Lynch, Seth Hutchinson, George A Kantor, and Wolfram Burgard. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[Cohen *et al.*, 2019] Liron Cohen, Tansel Uras, TK Satish Kumar, and Sven Koenig. Optimal and bounded-suboptimal multi-agent motion planning. In *Proceedings of Annual Symposium on Combinatorial Search (SOCS)*, 2019.

[Dayan *et al.*, 2021] Dror Dayan, Kiril Solovey, Marco Pavone, and Dan Halperin. Near-optimal multi-robot motion planning with finite sampling. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[Dobson and Bekris, 2014] Andrew Dobson and Kostas E Bekris. Sparse roadmap spanners for asymptotically near-optimal motion planning. *International Journal of Robotics Research (IJRR)*, 2014.

[Donald *et al.*, 1993] Bruce Donald, Patrick Xavier, John Canny, and John Reif. Kinodynamic motion planning. *Journal of the ACM (JACM)*, 1993.

[Dubins, 1957] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 1957.

[Edelkamp and Schrodl, 2011] Stefan Edelkamp and Stefan Schrodl. Introduction. In *Heuristic search: theory and applications*, chapter 1. 2011.

[Elbanhawi and Simic, 2014] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *Ieee access*, 2014.

[Erdmann and Lozano-Perez, 1987] Michael Erdmann and Tomas Lozano-Perez. On multiple moving objects. *Algorithmica*, 1987.

[Feng *et al.*, 2020] Zhi Feng, Guoqiang Hu, Yajuan Sun, and Jeffrey Soon. An overview of collaborative robotic manipulation in multi-robot systems. *Annual Reviews in Control*, 2020.

[Geraerts and Overmars, 2007] Roland Geraerts and Mark H Overmars. Creating high-quality paths for motion planning. *International Journal of Robotics Research (IJRR)*, 2007.

[Han *et al.*, 2018] Shuai D Han, Edgar J Rodriguez, and Jingjin Yu. Sear: A polynomial-time multi-robot path planning algorithm with expected constant-factor optimality guarantee. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[Hart *et al.*, 1968] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 1968.

[Hearn and Demaine, 2005] Robert A Hearn and Erik D Demaine. Pspace-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science (TCS)*, 2005.

[Hönig *et al.*, 2016] Wolfgang Hönig, TK Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*, 2016.

[Hönig *et al.*, 2018] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics (T-RO)*, 2018.

[Hopcroft *et al.*, 1984] John E Hopcroft, Jacob Theodore Schwartz, and Micha Sharir. On the complexity of motion planning for multiple independent objects; pspace-hardness of the warehouseman's problem. *International Journal of Robotics Research (IJRR)*, 1984.

[Hsu *et al.*, 1997] David Hsu, J-C Latombe, and Rajeev Motwani. Path planning in expansive configuration spaces. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 1997.

[Kala, 2013] Rahul Kala. Rapidly exploring random graphs: motion planning of multiple mobile robots. *Advanced Robotics*, 2013.

[Karaman and Frazzoli, 2011] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research (IJRR)*, 2011.

[Kavraki *et al.*, 1996] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 1996.

[Kottinger *et al.*, 2022] Justin Kottinger, Shaull Almagor, and Morteza Lahijanian. Conflict-based search for multi-robot motion planning with kinodynamic constraints. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022.

[Kuffner and LaValle, 2000] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2000.

[LaValle and Kuffner Jr, 2001] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *International Journal of Robotics Research (IJRR)*, 2001.

[LaValle, 1998] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Computer Science Department, Iowa State University (TR 98–11), 1998.

[LaValle, 2006] Steven M LaValle. *Planning algorithms*. Cambridge University Press, 2006.

[Le and Plaku, 2018] Duong Le and Erion Plaku. Cooperative, dynamics-based, and abstraction-guided multi-robot motion planning. *Journal of Artificial Intelligence Research (JAIR)*, 2018.

[Li *et al.*, 2021] Jiaoyang Li, Zhe Chen, Daniel Harabor, P Stuckey, and Sven Koenig. Anytime multi-agent path finding via large neighborhood search. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2021.

[Li *et al.*, 2022] Jiaoyang Li, Zhe Chen, Daniel Harabor, Peter J Stuckey, and Sven Koenig. Mapf-lns2: Fast repairing for multi-agent path finding via large neighborhood search. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

[Okumura *et al.*, 2021] Keisuke Okumura, Yasumasa Tamura, and Xavier Défago. Iterative refinement for real-time multi-robot path planning. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[Okumura *et al.*, 2022] Keisuke Okumura, Ryo Yonetani, Mai Nishimura, and Asako Kanezaki. Ctrms: Learning to construct cooperative timed roadmaps for multi-agent path planning in continuous spaces. In *Proceedings of International Joint Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, 2022.

[Okumura, 2023a] Keisuke Okumura. Improving lacam for scalable eventually optimal multi-agent pathfinding. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, 2023.

[Okumura, 2023b] Keisuke Okumura. Lacam: Search-based algorithm for quick multi-agent pathfinding. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2023.

[Pan *et al.*, 2012] Jia Pan, Sachin Chitta, and Dinesh Manocha. Fcl: A general purpose library for collision and proximity queries. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[Reif, 1979] John H Reif. Complexity of the mover's problem and generalizations. In *Proceedings of Annual Symposium on Foundations of Computer Science (FOCS)*, 1979.

[Sánchez and Latombe, 2002] Gildardo Sánchez and Jean-Claude Latombe. On delaying collision checking in prm planning: Application to multi-robot coordination. *International Journal of Robotics Research (IJRR)*, 2002.

[Sharon *et al.*, 2015] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence (AIJ)*, 2015.

[Shome *et al.*, 2020] Rahul Shome, Kiril Solovey, Andrew Dobson, Dan Halperin, and Kostas E Bekris. drrt∗: Scalable and informed asymptotically-optimal multi-robot motion planning. *Autonomous Robots (AURO)*, 2020.

[Silver, 2005] David Silver. Cooperative pathfinding. In *Proceedings of AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2005.

[Solis *et al.*, 2021] Irving Solis, James Motes, Read Sandström, and Nancy M Amato. Representation-optimal multi-robot motion planning using conflict-based search. *IEEE Robotics and Automation Letters (RA-L)*, 2021.

[Solovey *et al.*, 2016] Kiril Solovey, Oren Salzman, and Dan Halperin. Finding a needle in an exponential haystack: Discrete rrt for exploration of implicit roadmaps in multi-robot motion planning. *International Journal of Robotics Research (IJRR)*, 2016.

[Spirakis and Yap, 1984] P. Spirakis and C. Yap. Strong np-hardness of moving many discs. *Information Processing Letters*, 1984.

[Standley, 2010] Trevor Scott Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2010.

[Stern *et al.*, 2019] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of Annual Symposium on Combinatorial Search (SOCS)*, 2019.

[Švestka and Overmars, 1998] Petr Švestka and Mark H Overmars. Coordinated path planning for multiple robots. *Robotics and autonomous systems*, 1998.

[Van Den Berg and Overmars, 2005] Jur P Van Den Berg and Mark H Overmars. Prioritized motion planning for multiple robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[Wagner and Choset, 2015] Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence (AIJ)*, 2015.

[Wagner *et al.*, 2012] Glenn Wagner, Minsu Kang, and Howie Choset. Probabilistic path planning for multiple robots with subdimensional expansion. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012.

[Wurman *et al.*, 2008] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 2008.