algorithms

MDPI

Review

On the Intersection of Computational Geometry Algorithms with Mobile Robot Path Planning

Ehsan Latif and Ramviyas Parasuraman *

School of Computing, University of Georgia, Athens, GA 30602, USA; ehsan.latif@uga.edu
* Correspondence: ramviyas@uga.edu

Abstract: In the mathematical discipline of computational geometry (CG), practical algorithms for resolving geometric input and output issues are designed, analyzed, and put into practice. It is sometimes used to refer to pattern recognition and to define the solid modeling methods for manipulating curves and surfaces. CG is a rich field encompassing theories to solve complex optimization problems, such as path planning for mobile robot systems and extension to distributed multi-robot systems. This brief review discusses the fundamentals of CG and its application in solving well-known automated path-planning problems in single- and multi-robot systems. We also discuss three winning algorithms from the CG-SHOP (Computational Geometry: Solving Hard Optimization Problems) 2021 competition to evidence the practicality of CG in multi-robotic systems. We also mention some open problems at the intersection of CG and robotics. This review provides insights into the potential use of CG in robotics and future research directions at their intersection.

Keywords: computational geometry; robotics; multi-robot systems; path planning; optimization

Citation: Latif, E.; Parasuraman, R. On the Intersection of Computational Geometry Algorithms with Mobile Robot Path Planning. *Algorithms* **2023**, *16*, 498. https://doi.org/10.3390/a16110498

Academic Editors: Michele Scquizzato and Gopal Pandurangan

Received: 27 September 2023
Revised: 23 October 2023
Accepted: 25 October 2023
Published: 27 October 2023

1. Introduction

Computing geometric (CG) attributes of groups of geometric objects in space, such as the straightforward above/below connection of a given point concerning a particular line, is the focus of CG. CG is more broadly concerned with creating and evaluating geometric problem-solving algorithms. In the mathematical discipline of CG [1], practical algorithms for resolving geometric input and output issues are designed, analyzed, and put into practice. It can also refer to the solid modeling methods used to control surfaces, curves, and pattern recognition. The late 1970s saw the creation of this field, which quickly advanced through the 1990s to reach its current state. The study of sorting and searching algorithms employed in one-dimensional spaces to address problems involving multi-dimensional inputs led to the development of computation-based geometry. Its story also influenced insights from computational graph theories used in natural geometric settings. Problems in two-dimensional and, rarely, three-dimensional space were the primary emphasis of this field in the beginning [2].

Most researchers assume that the space's dimension is a tiny constant when considering mathematical issues within multi-dimensional environments. However, scholars who concentrated on discrete algorithms rather than numerical analysis were the ones who founded this field. They also focused on the characteristics of geometric problems rather than conventional continuous questions. As a result, most of the objects in this field are flat and straight, including lines, polygons, planes, and line segments. It occasionally employs curved things like circles, but avoids solid modeling issues with intricate curves and surfaces [3].

Robotics is the study of the creation and application of robots. Robots are geometric items that function in a three-dimensional space or the actual world; therefore, it stands to reason that geometry issues emerge frequently. The design of robots and the work cells in which they must function present additional geometric challenges. The majority of

industrial robots are fixed-base robot arms. The parts that the robot arm manipulates must be provided in a way that makes it simple for the robot to grip them. It might be necessary to paralyze some components for the robot to work on them. Before the robot can work on them, they might need to be turned toward a known orientation. All of these issues are geometric, occasionally with a kinematic element.

The path planning issue, in which a robot must discover a path across an environment with obstacles, is discussed in detail in a later section of this review. We examine simple, complex path-planning problems in this article. The broader issue of task planning has several sub-problems, one of which is motion planning. Giving a robot high-level tasks, such as "vacuuming the room", and then letting the robot decide how to carry them out is desirable. Planning motions, the sequence in which to complete sub-tasks, and other things fall within this category.

Methodology: This is a brief review of the intersection of CG and mobile robot path planning. To provide a structured overview, this paper adopts a systematic review methodology as discussed below:

1. First, the scope of path planning is clearly demarcated into two primary areas: single-robot and multi-robot systems;
2. The review predominantly focuses on the literature published between 2016 and 2023, considering papers using CG in mobile robots, ensuring relevance and incorporation of the latest advancements in the field; however, to discuss the foundational work on computational geometry, we have considered pioneering works from the years 1969 to 2008;
3. We also have taken articles from the winners of CG-SHOP 2021, as they are the most relevant and recent works on the discussed topic;
4. During the literature search, keywords such as "computational geometry in robot path planning", "single-robot path planning", "multi-robot coordination", and "kino-dynamic path planning" were employed. These were searched across renowned databases like IEEE Xplore, Google Scholar, and ScienceDirect;
5. Subsequently, the selected literature was examined based on specific criteria such as relevance to the topic, citation count, and the novelty of the proposed solutions;
6. The extracted information was then organized and synthesized to offer a comparative analysis, using features such as algorithmic complexity, optimality, scalability, and real-world applicability for comparing various solutions.

Through this methodical approach, this review aims not only to present a thorough understanding of existing solutions, but also to highlight the gaps and potential avenues for future research, underscoring the authors' contributions to this evolving domain.

This review primarily focuses on CG-based complex optimization solutions for path-planning problems in single- and multi-robot systems.

1.1. Computational Geometry

Currently, computational geometry concerns the computational complexity of geometric problems within algorithm analysis. However, the phrase has also been used in at least two other contexts. Forrest refers to this field as "computational geometry" in his study [4] since Riesenfeld [5] investigated geometric modeling using spline curves and surfaces. However, this subject is more closely related to numerical analysis than geometry.

The term "Computational Geometry" has already been used in at least two other situations by Lee et al. [6], and geometric algorithms have been created in various settings such as B-spline curves and surfaces [7]. Here, we try to examine these related activities from a proper perspective and compare them to the connotation that is currently in use:

- Bézier, Forrest, and Riesenfeld have successfully handled the problem of geometric modeling using spline curves and surfaces [7], which is more closely related to numerical analysis than geometry in terms of spirit;
- Many computer scientists believe that Michael Shamos' Ph.D. thesis at Yale University in 1978 [8] or perhaps his earlier paper on geometric complexity [9] are the two works

that gave rise to the field of computational geometry, which is frequently referred to as a new one in the area of computing science;
- Others would argue that it started 10 years earlier with Robin Forrest's Ph.D. thesis at Cambridge University in 1968 [10], or possibly with his later papers on computational geometry [4];
- Finally, others would argue that the formal examination of Minsky and Papert [11] into which geometric properties of a figure can and cannot be identified (computed) using different neural network models of computation is where it all started.

Fundamental theories and algorithms mentioned in the Handbook on Computational Geometry [2] are convex hull computations, Voronoi diagrams, Delaunay triangulation, arrangements, triangulation and mesh generation, polygons, shortest paths, proximity algorithms, visibility graphs, geometric reconstruction, curve and surface reconstruction, computational convexity, and computational and quantitative real algebraic geometry. Here, we demonstrate the application of computational geometry in robotics to solve specific path-planning problems for single- and multi-robotic systems.

To solve a problem using computational geometry, the following components are required:

- Size of input to the algorithm;
- Dimension of the problem;
- Properly defined constraints;
- Objective functions;
- Modality of the algorithm.

Figure 1 depicts the overview of CG-based problem-solving specific to robotics and path-planning problems. The CG-based solutions can be extended to a path-planning problem for multiple robots by considering a distributed algorithm running on each robot for achieving common objectives.

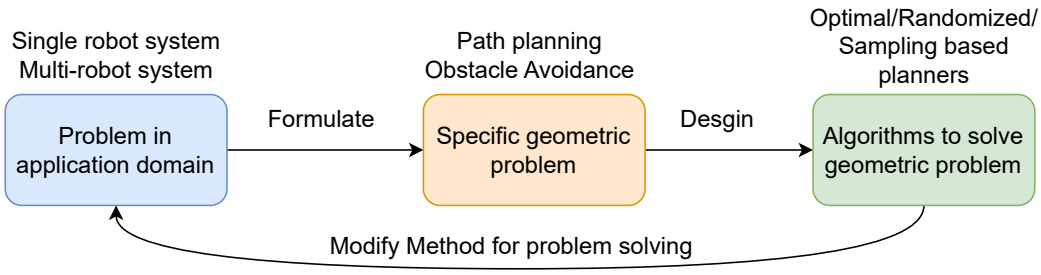

Figure 1. An overview of the computational geometric way of solving a problem.

1.2. Robotics

Creating autonomous robots—robots that can be instructed on what to do without being trained on how to do it—is one of robotics' ultimate objectives. This necessitates a robot having the ability to arrange its movements.

A robot needs to have some understanding of the environment through which it is passing to be able to plan its motion. A mobile robot moving around in a plant, for instance, needs to be aware of where impediments are. A floor plan can provide information, such as where walls and equipment are situated. The robot will have to rely on its sensors for more details. It should recognize barriers that are not shown on the floor plan, such as humans. The robot must move to its objective position using the environment's information without encountering any obstacles.

Whenever a robot of any kind wishes to move to a physical location, this path-planning problem must be resolved. The explanation above presupposes that we have an autonomous robot roaming a manufacturing setting.

2. Path Planning

Mobile robots, unmanned aircraft, and autonomous vehicles use path-planning algorithms to find the safest, most effective, collision-free, and most minor expensive routes between two points. Point-to-point navigation can be made safe and efficient by selecting the correct path-planning algorithm. However, the best algorithm depends on the robot geometry and computing constraints, including statically/holonomically and dynamically/non-holonomically constrained systems, and it necessitates a thorough understanding of current solutions.

A continuous path connecting a robot from an initial to a final goal configuration must be found in path planning, a non-deterministic polynomial-time ("NP") hard problem [12]. As the robot's degrees of freedom rise, the problem becomes more difficult. Then, the best course of action (the optimal path) will be chosen based on restrictions and requirements, such as the shortest distance between two points or the shortest time to travel without colliding. Sometimes, goals and limits coexist, such as when attempting to reduce energy use without increasing journey time past a certain point. Figure 2 delineates the path planning with obstacle avoidance along with the computational geometric planners.

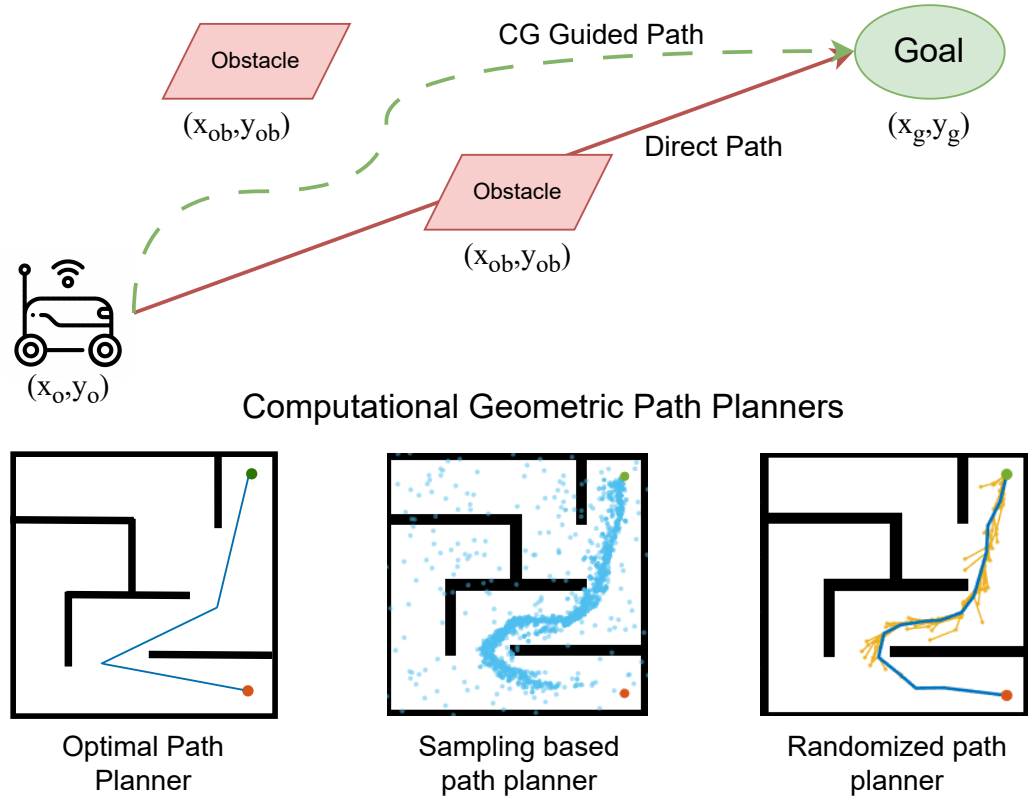

Figure 2. Pictorial view of robotic path planning and obstacle avoidance using computational geometry.

Four requirements must be met for a path-planning algorithm to be effective. First, in realistic static situations, the motion-planning technique must always be able to identify the best path. Second, it also needs to be scalable to dynamic contexts. Third, it must continue to be beneficial to and compatible with the selected self-referencing strategy. Finally, the complexity, data storage, and computing time must all be kept to a minimum [13]. The most prevalent path-planning algorithms for robots and AVs are presented in this paper, and we examine which method is best suited for static and dynamic environments. A taxonomy of different CG-based algorithms applied to mobile robot path planning is illustrated in Figure 3.

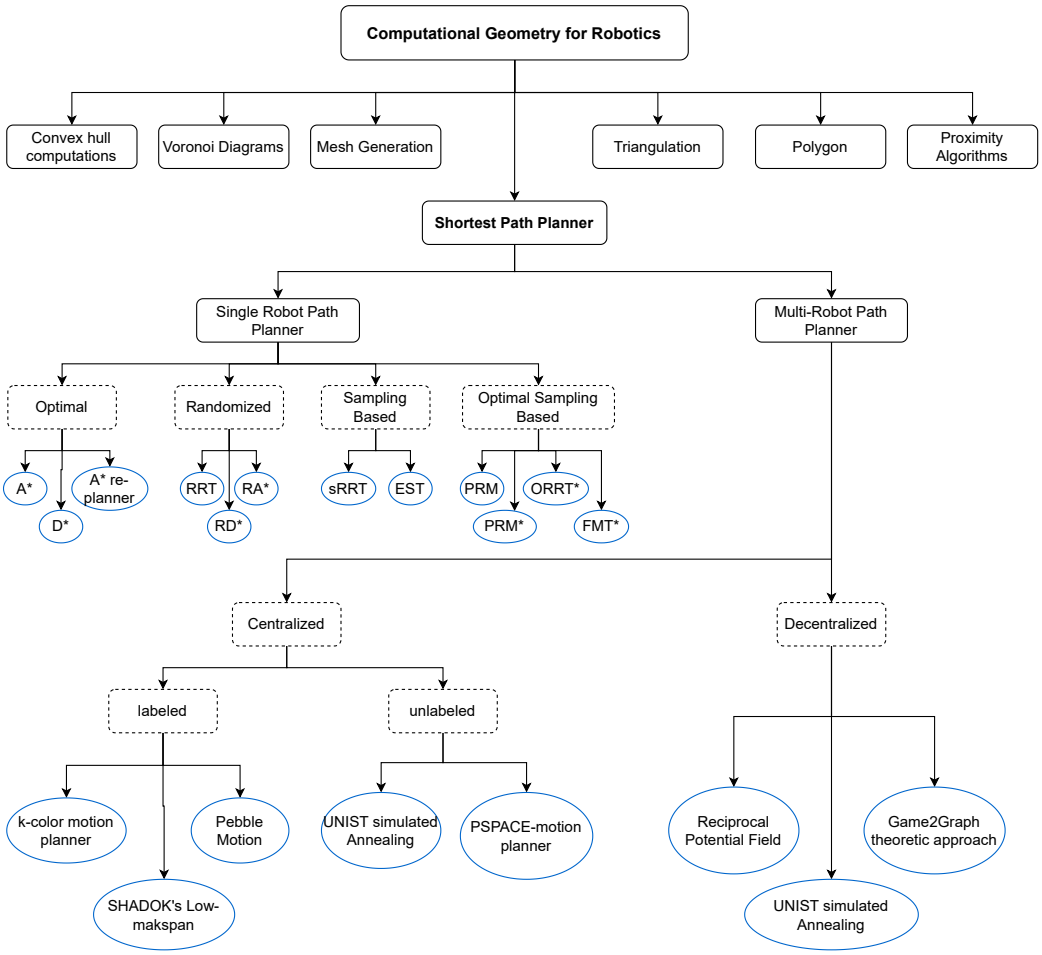

Figure 3. Taxonomy of robotic algorithms using computational geometry for path planning.

2.1. CG-Based Single-Robot Path Planning

Due to the difficulty in path planning for robots with multiple degrees of freedom (more than 4 or 5), sampling-based algorithms have been created that compromise completeness and practical application. Using such methods avoids the explicit geometric depiction of the open space. Instead, in their most basic form, they generate a sizable number of configuration sample sets and exclude samples that reflect the robot in an unallowable configuration.

Then, an effort is made to connect close configurations using a metric set across the configuration space. Finally, a roadmap-like graph structure is created as a result of this construction. It has the property that every path connecting any two of its vertices corresponds to an accessible path connecting the corresponding configurations in the configuration space. Although these methods are intrinsically imperfect, it is frequently possible to demonstrate that they are probabilistically complete—guaranteed, given enough samples, to identify a solution with a high probability if one exists.

Problem Statement: In single-robot path planning, our primary objective is to find a feasible and optimal path for a robot navigating a given environment. This entails determining a path within a configuration space C, a subset of the n-dimensional Euclidean space, denoted as $C \subset \mathbb{R}^n$. Within this space, there exists an obstacle space C_{obs}, such that $C_{\text{obs}} \subset C$. For a given starting configuration q_{start}, which belongs to C but not to C_{obs}, and a goal configuration q_{goal}, which also belongs to C but outside C_{obs}, the challenge is to determine a continuous path $\pi : [0,1] \rightarrow C$. This path must start at the given starting configuration, $\pi(0) = q_{\text{start}}$, and end at the goal configuration, $\pi(1) = q_{\text{goal}}$, ensuring that, for every $t \in [0,1]$, $\pi(t) \notin C_{\text{obs}}$. An optimal path is typically characterized by minimizing

the integral $\int_0^1 ||\pi'(t)||dt$, where $||\pi'(t)||$ signifies the velocity or rate of change of the robot's configuration.

Optimal Planners: The shortest path evaluation for the known static environment is a two-level problem that entails the selection of feasible node pairs. The most straightforward path evaluation is based on the obtained possible node pairs [14]. Therefore, A* and re-planner are used for shortest path evaluation based on the information regarding the obstacles present in the environment. However, the approach is ineffective and unworkable in dynamic contexts since neither of the above conditions exists. The algorithm uses a heuristic to estimate the cost from the current node to the goal. Let $G = (V, E)$ be a graph where V is the set of vertices and E is the edges. For a given node n, let $g(n)$ denote the cost of the shortest path from the start node to n and $h(n)$ be the heuristic estimate of the cost from node n to the goal. The A* algorithm evaluates nodes by combining $g(n)$ and $h(n)$ to form $f(n)$, defined as

$$f(n) = g(n) + h(n). \tag{1}$$

The objective of the A* algorithm is to minimize $f(n)$, ensuring that the algorithm is both complete (it will find a solution if one exists) and optimal (it will find the shortest possible path), provided that the heuristic $h(n)$ is admissible (never overestimates the actual cost to reach the goal) and consistent (satisfies the triangle inequality).

D* [15] and its variants are suggested practical tools for speedy re-planning in congested contexts to facilitate path planning in dynamic environments.

Randomized Planners: Rapidly exploring random trees (RRTs) [16], and a hybrid strategy that combines Relaxed A* (RA*) [17] with one meta-heuristic algorithm as D*, and its variations do not guarantee solution quality in significant dynamic contexts. In RRT, given a configuration space C, the algorithm iteratively builds a tree T rooted at the starting configuration q_{start}. For each iteration, a random sample q_{rand} is drawn from C, and the nearest node q_{near} from the tree T to q_{rand} is determined. A new node q_{new} is then added to the tree T by moving from q_{near} towards q_{rand} by a specified distance. The process is repeated until a termination condition is met, e.g., reaching q_{goal} or after a certain number of iterations.

The hybrid technique consists of two phases: an initialization phase where the algorithm is initialized using RA* and a post-optimization degree where the quality of the solution identified in the earlier stage is improved using one heuristic method. Like A*, RA* maintains a heuristic function $h(n)$ for estimating the cost from node n to the goal. The function $f(n)$ in RA* is defined similarly as in A*:

$$f(n) = g(n) + h(n) \tag{2}$$

However, RA* can operate in two phases: the search phase and the repair phase. In the search phase, RA* behaves like A*, finding an initial path. During the repair phase, if the costs of edges change, RA* updates the costs in its open list and re-plans, repairing the path without re-exploring the entire space. The Genetic, Ant colony, and Firefly algorithms—three meta-heuristic algorithms—are also counted as randomized planners. These seek to offer valuable features in support of a hybrid approach to path planning.

Sampling-based Planners: A sampling-based method of path planning [18] eliminates non-free configurations from a set of randomly chosen configurations. The edges are utilized to represent edges and are placed on the roadmap as collision-free paths. The sampling density can be increased in certain areas to hasten the roadmap connection. For instance, the oft-employed RRT generates a random configuration at each step and identifies the configuration most similar to the sample in the current tree. Other sampling techniques, such as expansive-space trees (EST) [19] and rapidly exploring random trees (RRTs) [16], presuppose that the initial and target configurations are known. EST evenly distributes samples in the undiscovered regions of the configuration space and then attempts to connect them to the tree, in contrast to RRTs' quick exploration of the space by extending from a tree towards random samples.

The EST algorithm aims to grow a tree in the configuration space C by placing more samples in less-explored regions. Let T be a tree rooted at the starting configuration q_{start}. The algorithm proceeds with the following general steps:

1. Select a random sample q_{rand} from C;
2. Determine a region R in C such that the probability of selecting a sample from R is inversely proportional to the volume of C already explored by T in R;
3. If q_{rand} lies in R, attempt to connect q_{rand} to the nearest node q_{near} in the tree T;
4. Add q_{rand} to T if a valid connection is established.

Mathematically, the probability $P(R)$ of selecting a sample in region R can be represented as

$$P(R) = \frac{V_{\text{unexplored}}(R)}{V_{\text{total}}(R)} \tag{3}$$

where $V_{\text{unexplored}}(R)$ denotes the volume of the region R that remains unexplored by T and $V_{\text{total}}(R)$ is the total volume of R.

Conventional sampling-based planners cannot determine the absence of a solution to a given problem. Disconnection proofs, which can establish that two given configurations dwell in two different free-space components in specific circumstances, are the subject of several works [20].

Optimal Sampling-based Planners: Probabilistic Roadmap (PRM) and RRT*, variations of PRM, are discussed, demonstrating their asymptotical optimality, in [21]. Additional neighbors for connection and a rewiring strategy are needed for RRT to be converted into an optimal planner. This guarantees that the resulting roadmap has a tree structure. A general paradigm for the asymptotic analysis of sampling-based planners is described in a recent paper [22].

PRM constructs a roadmap in the configuration space C by randomly sampling configurations and attempting to connect them. Let $G = (V, E)$ be a graph where V is the set of vertices (sampled configurations) and E is the set of edges (valid paths).

1. For $i = 1$ to n, sample a configuration q_{rand} from C;
2. If q_{rand} is collision-free, add it to V;
3. For each $q \in V$, attempt to connect q to k nearest neighbors in V that are collision-free;
4. Add valid connections to E.

After constructing the roadmap, search algorithms like A* can be used to find a path from q_{start} to q_{goal}.

On the other hand, RRT* is an optimization over the traditional RRT to ensure asymptotic optimality. Given a tree T rooted at q_{start}:

1. Sample a random configuration q_{rand} from C;
2. Find the nearest node q_{near} in T to q_{rand};
3. Attempt to connect q_{rand} to T through q_{near}, resulting in q_{new};
4. If the connection is valid, look for nodes in T within a radius r of q_{new} and attempt to rewire the tree to minimize the path cost.

The rewiring ensures that T not only spans the configuration space, but also converges to an optimal solution as the number of samples increases.

The framework takes advantage of a connection between such planners and conventional models of random geometric graphs, which have been thoroughly studied for many years and have many of their characteristics figured out by this point. Asymptotic features, or those with a high probability as the number of samples goes to infinity, are the focus of most analyses for sampling-based algorithms. For example, recent research [23] offers bounds on the likelihood of obtaining a PRM* solution that is close to optimal after a finite number of iterations. Additionally, we point out that the Fast Marching Technique (FMT*) [24] examines a setup comparable to this one for the algorithm, albeit for a specific situation of an obstacle-free workspace.

2.2. Transitioning from a Single-Robot to Multi-Robot Path Planning

As delineated in the single-robot path-planning problem, the objective revolves around the generation of an optimal and feasible path in a given configuration space. The essence of this approach lies in the computation of paths in the presence of obstacles, ensuring the absence of any intersection with the obstacle space C_{obs}. With a clear understanding of the single-robot paradigm, transitioning to the realm of multi-robot path planning necessitates the recognition and accommodation of additional complexities.

Constraints and Challenges: While the foundational principles of single-robot path planning can be employed in the multi-robot scenario, several key constraints emerge:

1. **Inter-Robot Collisions:** Unlike the single-robot scenario, the risk of collisions is not limited to static obstacles. Dynamic collisions between robots present a substantial challenge. The movement of one robot might obstruct the path of another, necessitating continuous reevaluation and adaptation;
2. **Coordinated Movement:** Ensuring that all robots reach their respective destinations within optimal time frames demands a high degree of coordination. This often involves sacrificing individual robot efficiency for collective efficiency;
3. **Increased Configuration Complexity:** The configuration space exponentially grows with the addition of each robot. This expansion augments the computational overhead and complicates the search for optimal paths.

A Bridge: CG serves as a bridge to extend single-robot path-planning solutions to cater to multiple robots. Several techniques come into play:

- **Minkowski Sum [25]:** This can be employed to compute the configuration space obstacles when dealing with moving robots. By considering one robot as the primary agent and treating other robots as moving obstacles, the problem can be reduced to a single-robot scenario in an augmented environment;
- **Voronoi Diagrams [26]:** These can assist in partitioning the workspace, providing each robot with a distinct region to navigate. This ensures a degree of separation between the robots, minimizing potential collisions;
- **Visibility Graphs [27]:** For environments with multiple robots and obstacles, visibility graphs can be extended to incorporate dynamic inter-robot constraints. This ensures that, while navigating from source to destination, the robot considers both static and dynamic elements in its path.

Role of Kinematics and Dynamics: In multi-robot systems, purely geometric considerations, although fundamental, may not suffice for effective path planning, especially in scenarios involving intricate robotic systems such as autonomous cars. The kinematic and dynamic parameters, including velocities and accelerations, play a pivotal role. Ignoring these parameters can lead to unrealistic path solutions that a physical robot cannot execute due to constraints in acceleration, turning radius, or speed limitations. For instance, two autonomous cars approaching an intersection may need to adjust their speeds to avoid collision, even if their geometric paths do not intersect. This emphasizes the need for path-planning algorithms to be kinodynamically aware, accounting for both the geometric constraints of the environment and the kinematic/dynamic constraints of the robots. Integrating these considerations ensures the generated paths are not only collision-free but also feasible, given the robots' physical characteristics and capabilities. Thus, a more holistic path-planning approach for multi-robot systems incorporates both geometric algorithms, as provided by computational geometry, and kinodynamic considerations to cater to the real-world complexities and constraints of robotic systems. The solutions discussed below consider both CG and kinodynamic systems for multi-robot path planning.

In essence, while multi-robot path planning inherently possesses greater complexities, the principles of computational geometry provide a framework to effectively leverage single-robot strategies, adapting and extending them for multi-agent scenarios.

2.3. Multi-Robot Path Planning

Several robots work together in a common workspace during multi-robot motion planning. The objective is to transfer the robots to their designated target positions without colliding with obstacles or other robots by keeping the overall schedule optimal concerning objective functions (minimize the makespan and minimize the total distance traveled by all robots). The issue can be seen as a single-robot scenario where the robot comprises multiple independent elements (robots). Let us say, for example, that the goal is to plan the movements of n robots; let S_i represent the configuration space of the ith robot. This can thus be rephrased as "*planning the motion of a single robot*" with a configuration space of $S = \{S_1, S_2, ..., S_n\}$. This discovery enables the direct application of single-robot tools to the multi-robot case. Such a simplistic approach, however, tends to be ineffective in theory and practice since it ignores the unique aspects of the current problem. Algorithm 1 generalizes the pseudocode of multi-robot path planning in different approaches. Different methods were created, especially for this issue. Table 1 depicts the comparative performance and complexity analysis of state-of-the-art multi-robot path planners.

Algorithm 1 Generalized Algorithmic Overview for Multi-Robot Path Planning

```
procedure MULTIROBOTPATHPLANNER(robots, environment)
    for each robot r_i in robots do
        start_i ← starting position of r_i
        goal_i ← goal position of r_i
    end for
    while not all robots have reached their goals do
        for each robot r_i in robots do
            if r_i has not reached goal_i then
                path_i ← PATHPLANNER(start_i, goal_i, environment)
                MOVEROBOTALONGPATH(r_i, path_i)
            end if
        end for
        HANDLECOLLISIONS(robots, environment)
    end while
end procedure
procedure PATHPLANNER(start, goal, environment)
    Centralized/Decentralized/Distributed path planners
    Anealing for UNIST [28]
    Path optimization for Gitastrophe [29]    ▷ Generate a path for a single robot from start to goal
    return path
end procedure
procedure MOVEROBOTALONGPATH(robot, path)    ▷ Move the robot along the generated path
end procedure
procedure HANDLECOLLISIONS(robots, environment)    ▷ Resolve any potential collisions among robots
end procedure
```

Centralized Planner (labeled/colored): The robots are divided into k groups (colors), with each group having replaceable robots, and a centralized algorithm for k-color multi-robot motion planning is given in [30]. Each robot must go to one of the target positions designated for its group, ensuring that, after the motion, precisely one robot occupies each target position. The continuous multi-robot issue and a discrete variation of it known as pebble motion on graphs are related in this study.

Consider that the robots are partitioned into k groups or colors. Robots within each group are interchangeable. The motion-planning problem for these k-colored robots is

presented in [30]. Each robot aims to move to one of the target positions designated for its group, ensuring that

$$\forall \text{ target positions } p, \exists \text{ exactly one robot } r, \text{ such that } r \rightarrow p \tag{4}$$

The multi-robot motion-planning problem in continuous space is closely related to its discrete counterpart: pebble motion on graphs, as discussed in [31]. In this discrete problem, the challenge is to navigate pebbles from a set of source vertices to a set of target vertices, respecting certain movement rules. The method to solve the continuous multi-robot motion planning problem is decomposing the k-color problem into multiple smaller pebble motion problems. Solutions to these sub-problems can then be merged to derive a solution to the original continuous problem.

Table 1. Comparative performance analysis of multi-robot path planners.

Algorithm	Category	Best Case	Worst Case	Average Case	Optimality	Update Cost	Abbreviations
Labeled [30]	Centralized	$O(n \times m)$	$O(2n \times m)$	$O(b \times n \times m)$	complete	low	b—no. of obstacles, m—no. of robots, n—no. of cells
Unlabeled [32]	Centralized	$O(logn)$	$O(n^3)$	$O(n^{2/3})$	complete	low	n—number of cells
Reciprocal [33]	Decentralized	$O(\mu \times n \times m)$	$O(\mu \times n \times m)$	-	sub	high	μ—makespan factor
Game-theoretic [34]	Decentralized	$O(\epsilon \times n)$	$O(\epsilon \times n)$	-	sub	high	ϵ—depth factor
Shadok [35]	Distributed	$O(w^2)$	$O(w^2)$	-	sub	high	w—number of labeled units
UNIST [28]	Distributed	$O(w^3)$	$O(w^3)$	-	sub	high	w—maximum space dimension
Gitastrophe k-opt [29]	Distributed	$O(k \times w)$	$O(k \times w)$	-	complete	moderate	k—number of robots and w—number of squares

Centralized Planner (unlabeled): Unlabeled multi-robot motion planning refers to the specific situation of the k-color issue with $k = 1$. When a sampling-based algorithm was originally proposed, it was first examined in [32]. This suggests that all robots in set R have the same potential target positions, represented by set P. Each target position should be occupied by one and only one robot at the end of the movement:

$$|R| = |P| \tag{5}$$

$$\forall p \in P, \exists! r \in R, \text{ such that } r \rightarrow p \tag{6}$$

The first exploration of this problem using a sampling-based algorithm was documented in [32]. Over the years, more efficient solutions were introduced, as highlighted in [36]. However, the inherent complexity of this problem was proven to be PSPACE-hard, according to [37]. Many solution algorithms operate on simplified assumptions, with common ones related to constraints between the start and target positions of the robots.

Recent years have introduced effective and comprehensive algorithms for the unlabeled problem [36]. For the problem to be proven to be PSPACE-hard [37], these methods simplify assumptions regarding the distance between the start and goal positions.

Decentralized Planner: A decentralized strategy that contends robot disputes should be settled locally and without centralized coordination. In the reciprocal technique [33], the individual robots compute their future trajectories while considering other robots' current position and velocity to prevent collisions. The research in [34] adopts a game-theoretical methodology. It produces a reward function that minimizes the time and resources required for coordination and increases the time between disputes. Based on the sociology of pedestrian interaction, several publications [38] create collision-avoidance

strategies. Most recent approaches use CG for multi-robot path planning while considering kino-dynamic constraints. For instance, Agarwal et al. [39] introduced a novel approach to multi-robot motion planning for unit discs utilizing revolving areas. Leveraging computational geometry, their strategy effectively handles robots as moving discs, focusing on the dynamic rearrangement of space while avoiding collisions. Although the approach excels in handling dynamic obstacles, its complexity tends to grow rapidly with the number of robots, posing scalability challenges.

On the other hand, Şenbaşlar and Sukhatme [40] proposed an asynchronous real-time decentralized planner that emphasizes the real-time nature of multi-robot systems. By avoiding central synchronization, their approach offers significant speedups, but at the potential cost of suboptimal paths. Choi et al. [41] explored parabolic relaxation techniques for motion planning, providing a unique intersection between computational geometry and optimization. However, the method may struggle with environments having numerous narrow passages due to its relaxed nature. Wang et al. [42] presented a coordination-free approach that integrates topological reasoning. By focusing on congestion reduction, their strategy ensures smoother navigation even in crowded scenarios, though it may sometimes lead to longer paths. Lastly, Zhang et al. [43] delved into an MIP-based approach for both geometric tasks and motion planning, offering a comprehensive solution. While their method stands out for addressing both task allocation and motion planning, it demands high computational resources, especially for larger teams of robots. These approaches are considered geometric as well as physical constraints of the multi-robot systems and are applicable under certain conditions. The following are the distributed approaches used in CG-SHOP 2021 for multi-robot path planning.

2.4. Distributed Approaches from CG-SHOP 2021

The 2021 CG-SHOP Challenge [44] was centered on the issue of calculating polygons with minimum areas whose vertices were a predetermined set of points in the plane. Numerous research teams from the computational geometry and combinatorial optimization communities responded enthusiastically to this challenge, which led to an active discussion of potential solutions. The challenge has two objective functions: minimize the makespan and minimize the total distance traveled by all nodes. There were three apparent front-runners: Team GITASTROPHE finished first in the total distance traveled objective and third in the makespan objective and, followed by Team UNIST in the second place in both the makespan and total distance objectives. Team SHADOKS achieved first place in the makespan and third place in total distance traveled.

Shadoks Approach to Low-Makespan: The Shadoks approach [35] utilized a few algorithms considering a coordinated motion planning issue in the two-dimensional grid Z^2. The objective is to transport a collection of robots, a set of n labeled unit squares, between the specified start and target grid cells without causing any collisions. The goal of the problem is to reduce the *makespan* as much as possible. The longest L_1 distance a robot can travel between its starting point and objective while avoiding obstacles is a trivial lower bound to the makespan. The objective function we maximize is the sum of all the variables with weight defined as

$$weight(r, P) = (\delta_r(p(0)) - \delta_r(p(k))) \times \left(\delta_r(p(0))^2 + 1\right), \tag{7}$$

Therein, $p(0)$ and $p(k)$ are the first and last positions of the path P, and $\delta_r(p)$ is the obstacle-avoiding distance from a point p to the target of the robot r. We can always swap start-target positions and reverse the pathways, preserving the best solution identified, because the problem is symmetric over time. Their strategy begins with transforming path planning into SAT problems to make it tractable, then applying A* for trivial optimization along with the cross, Cotty catcher, dichotomy, and escape strategies. Later, they improved their solution by using feasible and conflict optimizers. All the strategies work in parallel; overall computation complexity is polynomial.

UNIST Simulated Annealing Approach: The UNIST method [28] consists of three components. A feasible solution is first calculated to move each robot from its beginning to its goal point through a middle point. This is performed by setting intermediate points outside the minimum bounding box created by the input positions of the robots and obstacles. The second uses a straightforward local search strategy that involves regularly deleting and inserting a random robot along an ideal path. As a result of the robots no longer having to pass through the midway locations, the quality of the solution is improved. Finally, simulated annealing is used to enhance this workable solution further. To improve the feasible solution, UNIST has picked a robot i, deleted it from G, and inserted it again using the feasible solution. It may greatly improve this robot, as it no longer has to go through its middle-point m_i. UNIST calls this operation a tightening move (see Figure 4). UNIST used two alternative types of moves: either tightening the robot's whole trajectory or stretching it between two places by having the robot pass through a third intermediate point that was randomly created. They have run the deletion and insertion procedures $2w$ times and obtained a feasible solution in $O(w^3)$ time, where w is the maximum-sized dimension of world space.

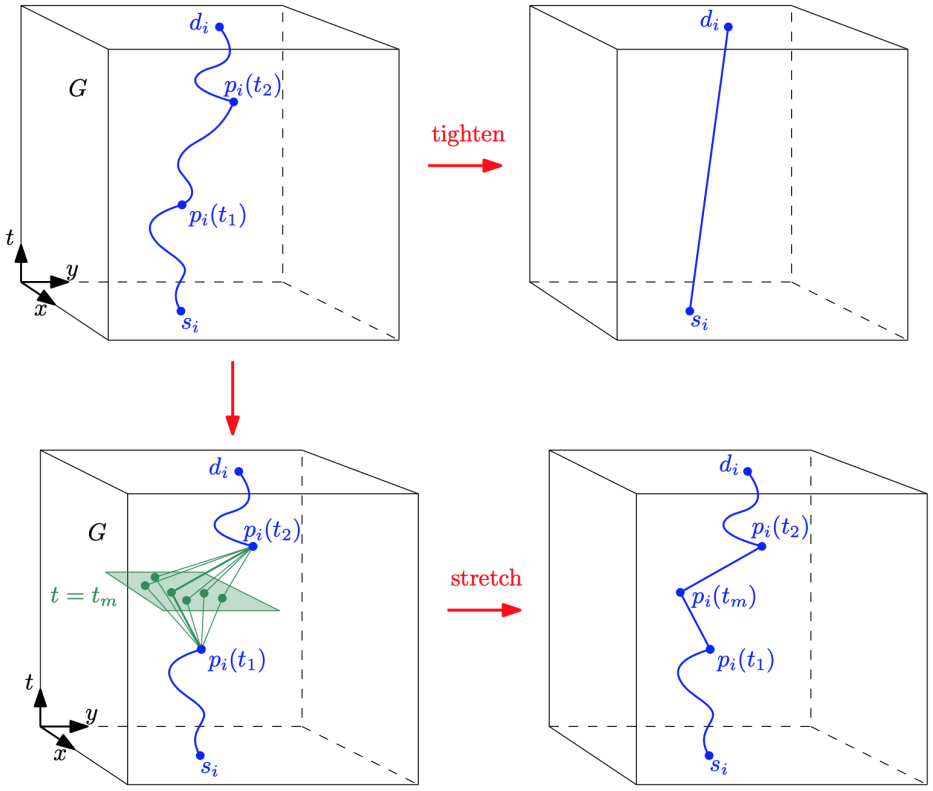

Figure 4. UNIST's tightening and stretching moves.

Gitastrophe Randomized k-Opt: An excellent initial solution is found during the initialization phase of the Gitastrophe technique [29], and this solution is then optimized during the k-opt local search phase. Gitastrophe's main contribution was to approximate the joint optimization using k unique single-path optimizations. The sorting analogy inspired this: given a set of k robots, route them, one at a time, to their destinations, with the jth robot routing respecting the path of robots $1, 2, \ldots, j-1$. Routing each robot sequentially from start to target prevents the exponential expansion of the state space, which is crucial. Finding a suitable ordering for the k robots to redirect becomes the only issue to solve. Simply arranging the robots to decrease completion time was where we had the most results. An ordering was abandoned if it was impossible (for instance, if a portion of the k robots entirely blocked the passage for the remaining robots) or not an improvement.

3. Future Directions

CG has excellent potential for optimally solving robotic path-planning problems, as evidenced by the approaches above. However, there is still space to work on exploiting CG for multi-robotic systems. Future studies should concentrate on the following areas:

Optimality and time complexity tradeoff: Optimal methods, such as DFS, A*, and D*, have a high computational cost in achieving optimal answers; randomized and sampling-based algorithms, on the other hand, produce sub-optimal and approximation solutions in less time. To achieve overall path-planning efficiency, the correct balance between optimality and execution time is essential;

Robot adaptability and cost efficiency: The dynamic environmental parameters of the robot may impact its movements and cause unwanted performance reduction. Robots may lack adaptability and become stuck in classic deadlock scenarios. To seek a collision-free path in an unknown environment with uncertain obstacles, a robot with the ability to change operating behavior over time is required. In addition to increasing the number of onboard sensors when dealing with complicated surroundings, the computation cost may be considerable, due to their high memory needs and processing expense. As a result, the computation cost considerations must be addressed when creating a proper CPP environment model. The hybrid algorithm is an intriguing invention for managing environmental change at the lowest possible cost;

Path Smoothness: In networked multi-robotic systems, communication dependability and connectivity coverage are critical. Because of its limited communication and sensory capabilities, the robot cannot regenerate the optimum path if the unexpected occurs, lowering the effective planning ratio. One of the issues that must be handled is a kinematic constraint of the robot, such as path curvature. Trajectory smoothing on a quick turn enables fast-moving robots such as drones to deliver an effective inertia motion transfer to decrease power consumption and prevent premature mechanical damage. As a result, a smooth path must be projected while adhering to the planned route;

Open Questions: There are open questions that can be addressed through the exploitation of CG in robotics. For example, we can use CG to determine if it is appropriate to employ an optimizing sampling-based planner or a non-optimizing sampling-based planner. Further, CG-based optimization techniques can be applied to optimize the resulting path for low computationally powered robots. We can also analyze convergence rates for sampling-based algorithms and give a defined length of time. Finally, CG can be helpful in solving the problems related to path planning for controlling non-holonomic and under-controlled robots.

4. Conclusions

Practical algorithms for addressing geometric input and output problems are devised, examined, and applied in the mathematical field of CG. It can also describe the solid modeling techniques for modifying surfaces and curves and allude to pattern recognition. CG is a broad field that includes ideas to address challenging optimization issues like path planning for robotic systems. The foundations of CG are discussed, along with how it can be used to solve well-known automated path-planning problems in both single-robot and multi-robot systems. We also go over three CG-SHOP 2021 competition winners' algorithms to demonstrate the applicability of CG in multi-robotic systems and highlight open CG-/robotics-related problems.

Author Contributions: Conceptualization, E.L.; writing—draft, review, and editing, E.L. and R.P.; supervision, R.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

CG	Computational Geometry
PRM	Probabilistic Road Map
DFS	Depth First Search
RRT	Rapidly-exploring Random Tree
EST	Expansive-Space Tree
FM	Fast Marching

References

1. Mark, D.B.; Otfried, C.; Marc, V.K.; Mark, O. *Computational Geometry Algorithms and Applications*; Spinger: Berlin/Heidelberg, Germany, 2008.
2. Berg, M.D.; Kreveld, M.V.; Overmars, M.; Schwarzkopf, O. Computational geometry. In *Computational Geometry*; Springer: Berlin/Heidelberg, Germany, 1997; pp. 1–17.
3. Preparata, F.P.; Shamos, M.I. *Computational Geometry: An Introduction*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
4. Forrest, A.R. Computational geometry-achievements and problems. In *Computer Aided Geometric Design*; Elsevier: Amsterdam, The Netherlands, 1974; pp. 17–44.
5. Riesenfeld, R.F. *Applications of b-Spline Approximation to Geometric Problems of Computer-Aided Design*; Syracuse University: Syracuse, NY, USA, 1973.
6. Lee, D.T.; Preparata, F.P. Computational geometry: A survey. *IEEE Trans. Comput.* **1984**, *33*, 1072–1101. [CrossRef]
7. Gordon, W.J.; Riesenfeld, R.F. B-spline curves and surfaces. In *Computer Aided Geometric Design*; Elsevier: Amsterdam, The Netherlands, 1974; pp. 95–126.
8. Shamos, M.I. *Computational Geometry*; Yale University: New Haven, CT, USA, 1978.
9. Shamos, M.I. Geometric complexity. In Proceedings of the Seventh Annual ACM Symposium on Theory of Computing, Albuquerque, NM, USA, 5–7 May 1975; pp. 224–233.
10. Forrest, A.R. Curves and Surfaces for Computer-Aided Design. Ph.D. Thesis, University of Cambridge, Cambridge, UK, 1968.
11. Minsky, M.; Papert, S. *Perceptron: An Introduction to Computational Geometry*; MIT Press: Cambridge, MA, USA, 1969.
12. Knoblauch, J.; Husain, H.; Diethe, T. Optimal continual learning has perfect memory and is np-hard. In Proceedings of the International Conference on Machine Learning, PMLR 2020, Virtual, 13–18 July 2020; pp. 5327–5337.
13. Campbell, S.; O'Mahony, N.; Carvalho, A.; Krpalkova, L.; Riordan, D.; Walsh, J. Path planning techniques for mobile robots a review. In Proceedings of the 2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE), Barcelona, Spain, 12–15 February 2020; pp. 12–16.
14. Dudi, T.; Singhal, R.; Kumar, R. Shortest Path Evaluation with Enhanced Linear Graph and Dijkstra Algorithm. In Proceedings of the 2020 59th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Chiang Mai, Thailand, 23–26 September 2020; pp. 451–456.
15. Patle, B.; Pandey, A.; Parhi, D.; Jagadeesh, A. A review: On path planning strategies for navigation of mobile robot. *Def. Technol.* **2019**, *15*, 582–606. [CrossRef]
16. Pérez-Hurtado, I.; Martínez-del Amor, M.Á.; Zhang, G.; Neri, F.; Pérez-Jiménez, M.J. A membrane parallel rapidly-exploring random tree algorithm for robotic motion planning. *Integr. Comput.-Aided Eng.* **2020**, *27*, 121–138. [CrossRef]
17. Ammar, A.; Bennaceur, H.; Châari, I.; Koubâa, A.; Alajlan, M. Relaxed Dijkstra and A* with linear complexity for robot path planning problems in large-scale grid environments. *Soft Comput.* **2016**, *20*, 4149–4171. [CrossRef]
18. Arias, F.F.; Ichter, B.; Faust, A.; Amato, N.M. Avoidance critical probabilistic roadmaps for motion planning in dynamic environments. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 10264–10270.
19. Tahir, Z.; Qureshi, A.H.; Ayaz, Y.; Nawaz, R. Potentially guided bidirectionalized RRT* for fast optimal path planning in cluttered environments. *Robot. Auton. Syst.* **2018**, *108*, 13–27. [CrossRef]
20. Varava, A.; Carvalho, J.F.; Pokorny, F.T.; Kragic, D. Caging and path non-existence: A deterministic sampling-based verification algorithm. In *Robotics Research*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 589–604.
21. Wang, Z.; Li, Y.; Zhang, H.; Liu, C.; Chen, Q. Sampling-based optimal motion planning with smart exploration and exploitation. *IEEE/ASME Trans. Mechatron.* **2020**, *25*, 2376–2386. [CrossRef]
22. Solovey, K.; Salzman, O.; Halperin, D. New perspective on sampling-based motion planning via random geometric graphs. *Int. J. Robot. Res.* **2018**, *37*, 1117–1133. [CrossRef]
23. Gammell, J.D.; Strub, M.P. Asymptotically optimal sampling-based motion planning methods. *Annu. Rev. Control Robot. Auton. Syst.* **2021**, *4*, 295–318. [CrossRef]
24. Gao, F.; Wu, W.; Lin, Y.; Shen, S. Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, 21–25 May 2018; pp. 344–351.

25. Adler, A.; Berg, M.D.; Halperin, D.; Solovey, K. Efficient multi-robot motion planning for unlabeled discs in simple polygons. In *Algorithmic Foundations of Robotics XI*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 1–17.
26. Huang, S.K.; Wang, W.J.; Sun, C.H. A path planning strategy for multi-robot moving with path-priority order based on a generalized Voronoi diagram. *Appl. Sci.* **2021**, *11*, 9650. [CrossRef]
27. Blasi, L.; D'Amato, E.; Mattei, M.; Notaro, I. Path planning and real-time collision avoidance based on the essential visibility graph. *Appl. Sci.* **2020**, *10*, 5613. [CrossRef]
28. Yang, H.; Vigneron, A. A Simulated Annealing Approach to Coordinated Motion Planning (CG Challenge). In Proceedings of the 37th International Symposium on Computational Geometry (SoCG 2021), Buffalo, NY, USA, 7–11 June 2021.
29. Liu, P.; Spalding-Jamieson, J.; Zhang, B.; Zheng, D.W. Coordinated motion planning through randomized k-opt. *ACM J. Exp. Algorithmics (JEA)* **2022**, *27*, 1–9. [CrossRef]
30. Shome, R.; Solovey, K.; Dobson, A.; Halperin, D.; Bekris, K.E. drrt*: Scalable and informed asymptotically-optimal multi-robot motion planning. *Auton. Robot.* **2020**, *44*, 443–467. [CrossRef]
31. Li, Q.; Gama, F.; Ribeiro, A.; Prorok, A. Graph neural networks for decentralized multi-robot path planning. In Proceedings of the 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 25–29 October 2020; pp. 11785–11792.
32. Lurz, H.; Recker, T.; Raatz, A. Spline-based Path Planning and Reconfiguration for Rigid Multi-Robot Formations. *Procedia CIRP* **2022**, *106*, 174–179. [CrossRef]
33. Bareiss, D.; Van den Berg, J. Generalized reciprocal collision avoidance. *Int. J. Robot. Res.* **2015**, *34*, 1501–1514. [CrossRef]
34. Smyrnakis, M.; Qu, H.; Veres, S.M. Improving multi-robot coordination by game-theoretic learning algorithms. *Int. J. Artif. Intell. Tools* **2018**, *27*, 1860015. [CrossRef]
35. Crombez, L.; da Fonseca, G.D.; Gerard, Y.; Gonzalez-Lorenzo, A.; Lafourcade, P.; Libralesso, L. Shadoks approach to low-makespan coordinated motion planning. *ACM J. Exp. Algorithms (JEA)* **2021**, *27*, 1–17. [CrossRef]
36. Debord, M.; Hönig, W.; Ayanian, N. Trajectory planning for heterogeneous robot teams. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 7924–7931.
37. Solovey, K.; Halperin, D. On the hardness of unlabeled multi-robot motion planning. *Int. J. Robot. Res.* **2016**, *35*, 1750–1759. [CrossRef]
38. Bajcsy, A.; Herbert, S.L.; Fridovich-Keil, D.; Fisac, J.F.; Deglurkar, S.; Dragan, A.D.; Tomlin, C.J. A scalable framework for real-time multi-robot, multi-human collision avoidance. In Proceedings of the 2019 International Conference on Robotics and Automation (ICRA), Montreal, QC, Canada, 20–24 May 2019; pp. 936–943.
39. Agarwal, P.K.; Geft, T.; Halperin, D.; Taylor, E. Multi-robot motion planning for unit discs with revolving areas. *Comput. Geom.* **2023**, *114*, 102019. [CrossRef]
40. Şenbaşlar, B.; Sukhatme, G.S. Asynchronous Real-time Decentralized Multi-Robot Trajectory Planning. In Proceedings of the 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 23–27 October 2022; pp. 9972–9979.
41. Choi, C.; Adil, M.; Rahmani, A.; Madani, R. Multi-robot motion planning via parabolic relaxation. *IEEE Robot. Autom. Lett.* **2022**, *7*, 6423–6430. [CrossRef]
42. Wang, X.; Sahin, A.; Bhattacharya, S. Coordination-free multi-robot path planning for congestion reduction using topological reasoning. *J. Intell. Robot. Syst.* **2023**, *108*, 50. [CrossRef]
43. Zhang, H.; Chan, S.H.; Zhong, J.; Li, J.; Koenig, S.; Nikolaidis, S. A mip-based approach for multi-robot geometric task-and-motion planning. In Proceedings of the 2022 IEEE 18th International Conference on Automation Science and Engineering (CASE), Mexico City, Mexico, 22–26 August 2022; pp. 2102–2109.
44. Fekete, S.P.; Keldenich, P.; Krupke, D.; Mitchell, J.S. Computing coordinated motion plans for robot swarms: The cg: Shop challenge 2021. *ACM J. Exp. Algorithmics (JEA)* **2022**, *27*, 1–12. [CrossRef]