Resolution-Complete Multi-agent Motion Planning with Arbitrarily-Shaped Obstacles

Biao Jia, Liang He, Dinesh Manocha

***Abstract*—We present a novel algorithm for homogeneous multi-agent motion planning in a continuous workspace with arbitrarily-shaped obstacles. Our approach extracts the medial-axis of the workspace and takes advantage of its geometric characteristics to spatially divide the agents into multiple clusters. For each cluster, we arrange the agents into a circular pattern and compute their movements using a combination of inter-loop and intra-loop movements. In practice, we approximate the medial-axis and using a finite number of boundary samples to show that our planning algorithm is resolution-complete. We highlight its performance on challenging 2D benchmarks and highlight the benefits over prior methods.**

***Index Terms*—Multi-agent, resolution-complete, Arbirarily-Shaped, Medial-Axis.**

I. Introduction

Multi-agent motion planning has been studied extensively in artificial intelligence, robotics and computer games. The objective is to navigate multiple agents from a set of start positions to corresponding goal positions while avoiding inter-agent collisions as well as collisions with obstacles. At a broad level, prior approaches can be classified into two categories: decentralized and centralized methods. For decentralized algorithms, the planner computes the trajectory for each agent separately for a short time horizon and uses some sort of coordination to avoid intersection between these local trajectories. On the other hand, the centralized methods combine the configuration degrees of each agent into one large high DOF (degree-of-freedom) system, which computes their paths together. Such centralized methods can provide completeness, including resolution-completeness or probabilistic-completeness guarantees. However, they are practical for scenarios with only a few low DOF agents.

Many previous centralized algorithms are limited to discrete workspaces [1], [2]. These methods are limited to homogeneous agents and assume that the workspace is divided into a finite number of discrete units, e.g. uniform grids, and that each agent moves one unit to another in each step. Other planning algorithms have been proposed for a continuous workspace, but they are limited to simple workspaces or assume that the obstacles have piece-wise linear boundaries. [3], [4], [5], [6], [7]. Furthermore, the running time of these algorithms is governed by the combinatorial complexity of the 2D obstacles, i.e. the number of edges. In many applications, the agents move in a continuous space and the obstacles in the scene can have arbitrary shapes, including non-convex polygons or curved boundaries. In such scenarios, using a discretization of the continuous workspace into a finite number of units can result in errors or the planner not being able to compute a feasible solution.

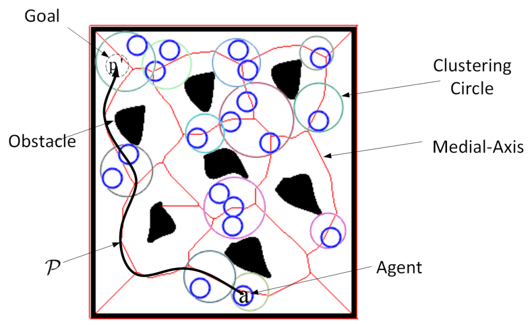

Figure 1: **Medial Axis based Multi-Agent Planning:** *We provide an overview of our approach. We show the computed path $\mathcal{P}$ for one of the agents moves toward its goal position. Given the black obstacles, we compute the medial axis of the workspace (shown in red) and use that to compute clusters of blue agents. The properties of medial axis are used to design resolution-complete multi-agent planner for arbitrary obstacles. Our approach results in speedups improvement over prior methods.*

Main Results: We present a novel and resolution-complete algorithm for multi-agent motion planning problem in a continuous 2D workspace with arbitrarily-shaped obstacles. We make no assumptions about the environment and the obstacles can be represented using piecewise lines or curves. Our approach utilizes the properties of the medial axis of the workspace. In particular, we compute an approximation of the Blum's medial axis using sampling methods. Based on the medial axis, we represent the free-space, i.e. the collision-free subset of the workspace, using a finite set of *elements* connected by *paths*. An element is a subset of free-space that has the same shape as the agent and the path connects two or more elements in the free space. Moreover, we use *circle packing* algorithms to represent a group of clustered agents. Based on this representation and decomposition, we transform the continuous planning problem into a discrete planning problem on graph, with each element being a node and each path being an edge of the graph.

Our method is able to handle the homogeneous disc-like agents. We prove that our approach is resolution-complete, if an approximate medial-axis of the workspace at any resolution, as well as a solution to the circle packing solution is available. In practice, we sample the boundaries of the obstacles using points and compute the approximate media-axis. In such cases, we show our multi-agent planning algorithm is resolution complete. We have evaluated our algorithm on 2D benchmarks with $40-100$ agents and we highlight the significant improvement in running time (up to 100X) over prior methods.

II. Prior Work

The multi-agent path planning problem has been extensively studied in different areas. We give a brief overview of prior work in this area.

A. Decentralized Methods

In this approach, each agent moves independently and the method computes a collision-free trajectory for a short time horizon. These techniques are commonly used in games, crowd simulation, and robotics. Some of the widely used methods are based on social forces [8], rule-based methods [9], or geometric optimization [10], [11], etc. However, there are no guarantees that these methods can find a collision-free solution, if one exists. Moreover, the agents can get stuck in deadlocks, though many techniques have been proposed to overcome these problems [12], [13].

Another class of methods is based on decoupled algorithms. These techniques first calculate the agents' trajectories without considering each other as obstacles and then schedule their trajectories such that the agents avoid each other [14], [15], [16], [17]. These algorithms can handle a continuous workspace, but cannot provide completeness guarantees. Other decoupled algorithms [2], [18], [19] use different kind of movements when agents get close to each other. These are mostly limited to discrete workspaces.

B. Centralized Methods

These methods theoretically treat all of the agents' configurations as one unified high-DOF system and compute the collision-free paths for them. Different methods to compute the paths have been proposed. These include traditional searching [1], [20], [21], [22] and probabilistic path planning [23], [24], [25]. In practice, since these methods compute the paths for all agents at once, so they are limited to a few agents.

Another category of centralized methods is based on geometric decomposition techniques. [26], [27], [28] divide the workspace into several sub-areas, and agents move from one to another until they reach their goals. These algorithms control the agents' movements so that they avoid each other when they are in close proximity. While these algorithms work in a continuous workspace, the boundaries of this workspace are limited to polygonal shapes and the complexity of the algorithms increases as a function of the number of obstacle edges. [29] present an optimal algorithm, but the complexity is $O(n^2m^2)$, where n and m are the number of agents and the number of obstacles' edges, respectively. In their benchmarks, the algorithm [29] takes hundreds or thousands of seconds to calculate a solution. Our approach is also a geometric decomposition method. We can handle arbitrarily-shaped obstacles and exhibit up to 100X speedup over [29].

III. Background and Notation

In this section, we introduce the notation used in the rest of the paper. We define a 2-D workspace $\mathcal{W}$, and a set of obstacles $\mathcal{O} \subseteq \mathcal{W}$. Every element in $\mathcal{O}$ can have any arbitrary shape. We assume that there are n homogeneous disc-like agents, $\{\mathbf{a_i}, i \in [1, ...n]\}$, with radius r in the workspace. We use $\mathbf{p_{a_i}}$ to represent their positions. The objective of multi-agent motion planning is to find paths for all agents to goal positions from start positions while avoiding collision with each other as well as the obstacles. If there is a solution, the complete algorithms should be able to compute it.

We also define $\mathcal{G}$ as the extracted Blum medial-axis [30] from a 2-D workspace. For any point in $\mathcal{G}$, there are more than one closest point on the obstacles' boundaries. According to this definition, for each point $\mathbf{v} \in \mathcal{G}$, there is a corresponding circle $A(\mathbf{v}, ran)$, where $\mathbf{v}$ is the center and ran represents the radius. A circle $A(\mathbf{v}, ran)$ is tangent to at least two different edges of the workspace, including boundaries of obstacles . $\mathcal{P}(\mathbf{p}, \mathbf{p}') \subseteq \mathcal{G}$ represents a path in $\mathcal{G}$, which includes two points $\mathbf{p} \in \mathcal{G}, \mathbf{p}' \in \mathcal{G}$ and every point in between on this path of $\mathcal{G}$.

In this paper, we use the symbol $\mathbf{A}$ to represent one cluster of agents. For each agent $\mathbf{a} \subseteq A(\mathbf{v}, ran)$, we define $\mathbf{a} \in \mathbf{A}$, which means every point on the agent is covered by the corresponding $A(\mathbf{v}, ran)$. We can say that, if an agent $\mathbf{a} \subseteq A(\mathbf{v}, ran)$ then $||\mathbf{p_a} - \mathbf{v}|| \leq ran - r$.

If the exact medial-axis is available, we show that our algorithm is resolution-complete. In practice, most techniques compute an approximate medial-axis by taking samples on the obstacle boundaries [31]. As a result, our multi-agent planner is resolution-complete, as its accuracy increases with the number of samples.

IV. Medial-axis and Circle Packing

Our goal is to design an algorithm for multi-agent motion planning in continuous 2D workspaces. In our approach, we use the medial-axis of the workspace to cluster the agents. Theoretically, this clustering step results in each agent being treated as a discrete element in the continuous space and also ensures that no feasible solution would be missed, up to the resolution of the medial-axis. Overall, our algorithm uses a combination of global and local computations to compute the path for each agent. As part of the global computation, we make use of the property of the medial-axis to divide agents into different clusters. As part of localized-cluster operations, we decompose $A(\mathbf{v}, ran)$ into several sub-areas if a large agent-cluster is associated with that region. Our motion planning algorithm is based on these agent clustering and area decomposition computations. Moreover, all of the agent movements (or the resulting) can be decomposed into intra-cluster and inter-cluster movements. First, we present some theorems and lemmas to illustrate the relationship between agent positions and the medial-axis. The details related to these theorems and the proof are given in the appendix (as part of the supplementary document).

Theorem 1. *For an agent, if it does not overlap with any obstacle, there will be at least one circle $A(\mathbf{v}, ran)$ that includes* *this agent. That is,* $\forall \mathbf{a}, \mathbf{a} \cap \mathcal{O} = \varnothing, \exists A(\mathbf{v}, ran), \mathbf{a} \subseteq A(\mathbf{v}, ran)$

We cluster all of the agents in the following manner. First, we randomly select one of the agents $\mathbf{a}$ and also select one of the points $\mathbf{v}$ on the medial-axis, that satisfy this condition $\{\mathbf{v} \mid argmin(||\mathbf{v} - \mathbf{p_a}||), \mathbf{v} \in \mathcal{G}\}$. Next, we compute an agent $\{\mathbf{a}' \mid argmin(||\mathbf{p_{a'}} - \mathbf{p_a}||)\}$. We compute a path by moving $\mathbf{v}$ along with the $\mathcal{G}$ toward the $\{\mathbf{v}' \mid argmin(||\mathbf{v}' - \mathbf{p_{a'}}||, \mathbf{v}' \in \mathcal{G})\}$, until $A(\mathbf{v}, ran)$ is tangent to $\mathbf{a}'$ and we stop moving $\mathbf{v}$. In other words, if we move $\mathbf{v}$ towards $\mathbf{p_{a'}}$, then $\mathbf{a}$ won't be covered by $A(\mathbf{v}, ran)$. We compute all the agents that $\forall \mathbf{a} \subset A(\mathbf{v}, ran), \mathbf{a} \in \mathbf{A_0}$. We repeat this computation till all the agents are part of a cluster (see Algorithm 1 for details). We also show the results of clustering in Figure. 1 for a simple scenarios.

```
Algorithm 1: Clustering the agents based on medial axis.
This is used in Algorithm 2, Step 2.
input : A workspace $\mathcal{W}$ with a set of agents
        $\mathbf{a}_0, \mathbf{a}_1, ..., \mathbf{a}_n$
output: A set of clustered agents $\mathbf{A}_0, \mathbf{A}_1, ..., \mathbf{A}_m$.
1  Build the medial-axis $\mathcal{G}$ of $\mathcal{W}$
2  $\mathcal{A}$ = All agents. /* repeat until all the
       agents are clustered */
3  while $\mathcal{A} \neq \varnothing$ do
4      Find $\mathbf{a} \in \mathcal{A}$
5      Find $\{\mathbf{v} \mid argmin(||\mathbf{v} - \mathbf{p_a}||, \mathbf{v} \in \mathcal{G})\}$
6      Find $\{\mathbf{a}' \mid argmin(||\mathbf{p_{a'}} - \mathbf{p_a}||)\}$
7      Find $\{\mathbf{v}' \mid argmin(||\mathbf{v}' - \mathbf{p_{a_n}}||, \mathbf{v}' \in \mathcal{G})\}$
8      while $A(\mathbf{v}, ran)$ is not tangent with $\mathbf{a}$ do
9          Move $\mathbf{v}$ along the $\mathcal{G}$ toward $\mathbf{v}'$
10         if $\mathbf{v}$ is overlap with $\mathbf{v}'$ then
11             Find $\{\mathbf{a}'' \mid argmin(||\mathbf{p_{a''}} - \mathbf{p_{a'}}||)\}$
12             Find $\{\mathbf{v}'' \mid argmin(||\mathbf{v}'' - \mathbf{p_{a'}}||, \mathbf{v}'' \in \mathcal{G})\}$
13             Replace the $\mathbf{a}'$ and $\mathbf{v}'$ with $\mathbf{a}''$ and $\mathbf{v}''$ and
               continue.
14     Remove all agents $\forall \mathbf{a} \subset A(\mathbf{v}, ran)$ from $\mathcal{A}$ and put
       them in a new cluster $\mathbf{A}$
```

Moreover, we remove all the vertices from $\mathcal{G}$, if the radius of the corresponding circles (based on medial axis computation) is smaller than that of the agents, since no agent can be in a collision-free configuration in such circles.

A. Agents Arrangement in a Cluster

In terms of efficiency, we want to use a general data structure to represent a group of clustered agents along with their incident circular area $A(\mathbf{v}, ran)$. This data structure is used as a layout that arranges each agent in a certain area of the incident circle $A(\mathbf{v}, ran)$. In other words, any arrangement of agents in the $A(\mathbf{v}, ran)$ can be transitioned into this general layout. This reduces to *circle packing* [32] problem. It corresponds to computing how to fill a given domain with a maximum number of uniquely-sized circles. According to [32] every circular area can only accommodate a certain number of agents and this number can be easily calculated. Notice that

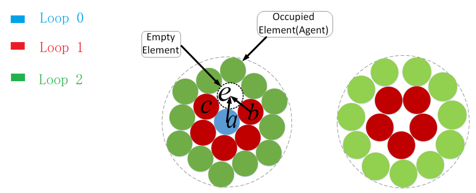

Figure 2: *This figure shows a circle of the medial-axis and a set of clustered agents. The elements are arranged tightly, and we use different color to represent the elements in different loops. The dotted circle represents an empty element* $\mathbf{e}$. *The two arrows,* $\mathbf{a}$ *to* $\mathbf{e}$ *and* $\mathbf{b}$ *to* $\mathbf{e}$ *show the inter-loop and intra-loop movements, respectively. The right figure shows another tight arrangement with different number of elements.*

this packing arrangement for a maximum number of agents is unique, and every agent connects to at least two other agents in such an arrangement. Such an arrangement is called a *tight arrangement* or a *tight pattern*. Note that when several clusters and their incident circular areas partially overlap with each other, the domain of clusters is not circle-like anymore. For example, when $A(\mathbf{v}, ran) \cap A'(\mathbf{v}, ran) \neq \varnothing$, the shape of $A(\mathbf{v}, ran) \cup A'(\mathbf{v}, ran)$ is not a circle. Currently, to best of our knowledge, optimal circle packing in a domain of any shape is a difficult problem with a high complexity. [33] presents an approximate method to fill the uniquely sized circles in any given domain with linear running time. In practice, their algorithm can find the approximate tight arrangement in tens of seconds. This running speed depends on the sampling resolution of the domain. We present a simplified method to compute a faster approximate solution to check if a combined circular domain is a tight arrangement in the supplemental document. In the following discussion, we first present our algorithm for a single domain.

We define two agents as connected if they can be tangent to each other along a moving path, without colliding with any other agents. We represent two such connected agents as $a \sim a'$. Assume an area with a group of disc-like agents. Any agent in the circle can find a path, which starts from one of it's connected agents and ends with another agent. Such a path only visits these agents once. We then call this path *loop* $\mathbf{T}$. With these formulations, we state the following lemma.

Lemma 1. *Assuming an area with a group of disc-like agents, then all of the agents in the circle can find a loop. That is, we have:* $\forall \mathbf{a} \subset A(\mathbf{v}, ran), \exists \mathbf{T}, \mathbf{a} \in \mathbf{T}$

The proof of Lemma 1 and the minimum number of connections theorems can be deducted from [32]. We call the arrangement based on such a loop as the *circular pattern*, which is composed of a set of *elements*. This arrangement be occupied by an agent or an empty element (i.e. without an agent) in each loop. We show two such possible layouts in Figure 2. Specifically, we label a cluster incident circle as *full* if it cannot cover any new agent. Otherwise, it is a *non-full* circle. Moreover, any set of clustered agents in a circle can be transited into a circular pattern by adjusting their positions if and only if the number of agents is not over the maximum capability of the area according to the analysis of [32]. Based on this formulation, the overall multi-agent algorithm is given in Algorithm 2. Figure 1 shows how to move an agent to its goal position through Algorithm 2.

```
Algorithm 2: Multi-Agent Planning Overall Algorithm
input : A workspace $\mathcal{W}$ with a set of agents
        $\mathbf{a}_0, \mathbf{a}_1, ..., \mathbf{a}_n$ and their goal position
        $\mathbf{p'}_0, \mathbf{p'}_1, ..., \mathbf{p'}_n$
output: Move all agents to their goal position or report
        no solution can be found.
1 Compute the medial-axis $\mathcal{G}$ of $\mathcal{W}$
2 Cluster the agents according to their positions and size.
3 for $\mathbf{a_i}, i = 0, 1, 2, ...n$ do
4     Find the cluster $\mathbf{A_k}$ that $\mathbf{a_i}$ belongs to.
5     Find the path $\mathcal{P}$ which starts from the $\mathbf{A_k}$ and ends
      with $\mathbf{A}_{k+t}$, which includes goal position the $\mathbf{p'}_i$.
      /* agent moves from cluster $\mathbf{A_k}$ to
         $\mathbf{A}_{k+t}$                                   */
6     for All clusters that $\mathcal{P}$ overlaps with do
7         $\mathbf{a_i}$ moves using the intra-loop and inter-loop
          exchange algorithm.
      /* Now agent $\mathbf{a_i} \subseteq \mathbf{A}_{k+t}$ and move to its
         goal position                                       */
8     $\mathbf{a_i}$ moves to $\mathbf{p'}_i$ using intra-loop and inter-loop
      exchange algorithm.
```

V. Element Movements

In this section we show how to move an element, that is performed as part of the intra-loop and inter-loop exchange algorithms (Algorithm 2, Steps 7 and 8). We first present the theorem, which illustrates the spatial relationship of two nearby elements' movements.

Theorem 2. *An element $\mathbf{c}$ can be moved to another nearby element$\mathbf{c}'$'s position if and only if both of them are connected to an empty element or at least one of them, $\mathbf{c}$ or $\mathbf{c}'$ is empty.*

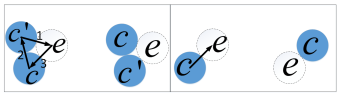

Figure 3: *This figure shows how Theorem 2 formulation can be used for moving/exchanging an element $\mathbf{c}$ with $\mathbf{c}'$. The left picture shows how two non-empty elements can exchange their positions, based on the arrows. To move a non-empty element $\mathbf{c}$ to the other non-empty element $\mathbf{c}'$, it has to be an empty element to place the $\mathbf{c}'$ and then move $\mathbf{c}$ to the position $\mathbf{p}'_\mathbf{c}$. This is shown as the third step with the arrow, which moves $\mathbf{c}'$ to $\mathbf{p_c}$.*

Theorem 2 illustrates the spatial requirements in terms of packing to move an element. Assuming both nearby elements $\mathbf{c}$ and $\mathbf{c}'$ are not empty in terms of agents, if $\mathbf{c}$ wants to move to $\mathbf{p}'_\mathbf{c}$ then there has to be at least one extra empty element $\mathbf{e}$ that connects both $\mathbf{c}$ and $\mathbf{c}'$ to complete the movement. The process involves moving $\mathbf{c}'$ to $\mathbf{p_e}$ in order to make its position free, and then moving $\mathbf{c}$ to $\mathbf{p}'_\mathbf{c}$. The other situation arises when either $\mathbf{c}$ or $\mathbf{c}'$ represents empty position, in which case we can directly move the agent to an empty position. We illustrate both of these situations in Figure 3.

Note that the above discussion assumes that $\mathbf{c}$ and $\mathbf{c}'$ are close to each other. However, if $\mathbf{c}$ and $\mathbf{c}'$ are not in close proximity, then $\mathbf{c}$ has to repeatedly try such a movement to move towards $\mathbf{p}'_\mathbf{c}$. During this computation (Algorithm 2, Steps 7 and 8), several agents are moved from their positions correspondingly. This could generate a problem because some of these agents being moved may already be at their goal positions. Ideally, we do not want to move such agents. Therefore, the design of an efficient algorithm is meant to move $\mathbf{c}$ to its goal position while keeping the irrelevant agents static. We present a general exchange algorithm. The input is two elements, and the output is these two elements exchanging their position while the rest of the agents are not moved. Fortunately, the exchange algorithms still meet the minimum spatial requirement in Theorem 2. That is, the only single connected empty element is involved in the whole processing.

Lemma 2. *Two elements that are in the same cluster can exchange their positions if and only if one of them is an empty element or there is the third empty element that connects at least one of them.*

In the following part of this section we first introduce the intra-loop exchange algorithm and then deduct the inter-loop exchange algorithm.

A. Intra-loop Exchange

The goal of intra-loop exchange is to exchange two elements' positions that are in same loop $\mathbf{c} \in \mathbf{T}, \mathbf{c}' \in \mathbf{T}$. This needs an extra empty element in a nearby loop $\mathbf{e} \in \mathbf{T}', \mathbf{T} \neq \mathbf{T}'$. The overall algorithm is shown in Algorithm 3. In line 1, $\mathbf{c}$ moves to the nearby empty position, thus emptying its own position. Note here that the loop $\mathbf{T}$ has an extra empty element because $\mathbf{c}$ has moved to anther loop. The algorithm moves all of the agents between $\mathbf{c}$ and $\mathbf{c}'$ by one step to adjust this empty element from $\mathbf{p_c}$ to $\mathbf{p}'_\mathbf{c}$. Figure 4 shows such an example.

```
Algorithm 3: Intra-loop Exchange: This is used in Algo-
rithm 2, Steps 7 and 8.
input : An element $\mathbf{c} \in \mathbf{T}$ with its initial position $\mathbf{p_c}$
        and the goal element $\mathbf{c}'$ position $\mathbf{p}'$ in the loop.
        An empty element, $\mathbf{e} \notin \mathbf{T}$, $\mathbf{e} \sim \mathbf{c}$
output: $\mathbf{c}$ and $\mathbf{c}'$ exchange their positions
1 Move $\mathbf{c}$ to $\mathbf{p_e}$; Move $\mathbf{e}$ at $\mathbf{p_c}$ to $\mathbf{p}'_\mathbf{c}$
2 while $\mathbf{c}$ and $\mathbf{c}'$ are not connected do
3     Simultaneously move all of elements $\in \mathbf{T}$.
4 Move $\mathbf{c}$ to $\mathbf{c}'$'s nearby empty element
5 Move $\mathbf{c}'$ to $\mathbf{p_e}$; Move $\mathbf{c}$ to $\mathbf{p}'_\mathbf{c}$
6 Move $\mathbf{c}'$ to $\mathbf{p_c}$ meanwhile return all agents $\in \mathbf{T}$ to their
  primitive positions
```

B. Inter-cluster Exchange

The inter-loop exchange represents two elements that are in different loops exchanging their positions. Again, we need an

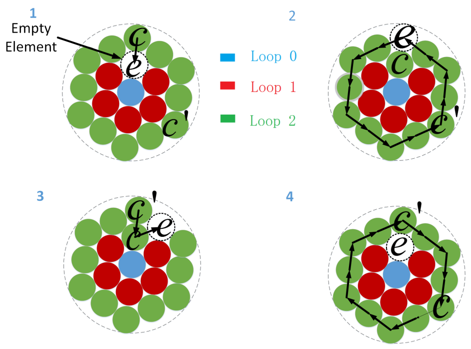

Figure 4: *Sequences of intra-loop exchange processing: The upper left picture shows two elements* $\mathbf{c} \in \mathbf{T}$ *and* $\mathbf{c}' \in \mathbf{T}$ *with an empty element* $\mathbf{e} \in \mathbf{T}'$. $\mathbf{c}$ *moves to* $\mathbf{p_e}$ *(shown with an arrow) in this picture. In the upper right picture, all of the agents in* $\mathbf{T}$ *push each other to move simultaneously so that* $\mathbf{c}$ *could connect to* $\mathbf{e}$ *in* $\mathbf{T}$. *In the bottom left picture,* $\mathbf{c}$ *moves back to loop* $\mathbf{T}$ *and* $\mathbf{c}'$ *temporarily move to loop* $\mathbf{T}'$. *In the bottom right picture, all the agents push each other back to their primitive position and* $\mathbf{c}'$ *moves towards* $\mathbf{T}$, *thus finishing the exchange processing step.*

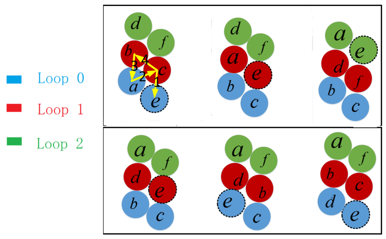

Figure 5: *This figure shows the sequence of how elements* $\mathbf{a}$ *and* $\mathbf{d}$ *exchange their positions when they are not in the nearby loop. In this example, there are three loops, and we highlight each of them in different colors. The upper row shows how* $\mathbf{a}$ *moves towards* $\mathbf{p_d}$ *with an empty element* $\mathbf{e}$. *In each inter-loop movement, the pairs of elements are moved to another loop. After* $\mathbf{a}$ *reaches its goal, the two pairs of elements are moved away from their original loops. In the bottom row, the algorithm uses the same strategy as the left column to return* $\mathbf{e}$ *back to its original position. Throughout this process, every element that has been moved returns to its original position.*

extra empty element $\mathbf{e}$ to connect one of the two elements $\mathbf{c}$ and $\mathbf{c}'$. We assume that they have the following relationships at the start state: $\{\mathbf{e}, \mathbf{c}, \mathbf{c}' | \mathbf{e} \sim \mathbf{c}, \mathbf{e} \in \mathbf{T}, \mathbf{c} \in \mathbf{T}, \mathbf{c}' \in \mathbf{T}', \mathbf{T}' \neq \mathbf{T}\}$. Our approach performs inter-loop exchanging, and we illustrate the approach in Figure 5.

VI. Resolution-Completeness

In this section, we present an algorithm for an agent's intra-cluster movements and prove its completeness. Next we present a criterion to connect all possible clusters. Based on checking whether a cluster $\mathbf{C}$ can be connected with another cluster whose incident circle $A(\mathbf{v}, ran)$ includes the goal position of an agent $\mathbf{a} \in \mathbf{C}$, we conclude whether there is path for $\mathbf{a}$ to reach its goal.

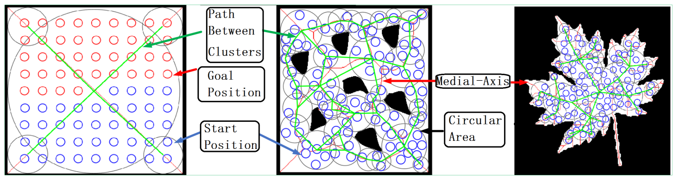

Figure 6: *We show three challenging benchmarks with* $40-100$ *agents, where techniques based on decentralized methods may not work well. We highlight the start positions (blue agents), the goal positions (red agents) and the medial-axis, the path between the clusters (green). Our algorithm can compute the paths for all agents in* $3-8$ *seconds.*

A. Completeness of Intra-cluster Computation

We extend Lemma 2 to the following lemma, which gives the minimum spatial requirement to implement an exchange algorithm inside of a circle.

Lemma 3. *Assume a cluster* $\mathbf{A}$ *and its incident circle* $A(\mathbf{v}, ran)$ *is non-full.* $\forall \mathbf{a} \in \mathbf{A}$, *we can find a complete path from its start position* $\mathbf{p_a} \in A(\mathbf{v}, ran)$ *to the goal position* $\mathbf{p_a'} \in A(\mathbf{v}, ran)$ *and we refer to that cluster* $\mathbf{A}$ *is complete.*

B. Connect Clusters

So far we have shown that for any agent, if both its start position and goal position are in a non-full circle, then our algorithm can find a path to reach its goal. In this section, we present an algorithm that connects such clusters together to extend the inter-loop exchange algorithm to inter-cluster level. We show that all the agents can move around in the connected cluster. Based on this criterion, the problem of path existence and computation for an agent reduces to checking whether the agent's goal position is within the connected cluster.

Theorem 3. *Suppose we have a path* $\mathcal{P}(\mathbf{v}, \mathbf{v}') \subseteq \mathcal{G}$, *that connects two areas* $A(\mathbf{v}, ran)$ *and* $A(\mathbf{v}', ran')$ *with their corresponding clusters* $\mathbf{A}$ *and* $\mathbf{A}'$. *Then, if* $\{\mathcal{P}(\mathbf{v}, \mathbf{v}') \mid \forall A(\mathbf{p}, ran''),\ \mathbf{p} \in \mathcal{P}(\mathbf{v}, \mathbf{v}'),\ \forall \mathbf{a} \notin \mathbf{A} \cup \mathbf{A}',\ \mathbf{a} \cap A(\mathbf{p}, ran) = \varnothing\}$, *we label the path* $\mathcal{P}(\mathbf{v}, \mathbf{v}')$ *as clean.*

Note that we mentioned that $\mathcal{G}$ does not include the points whose incident circles' radius are smaller than the agent's radius. Thus, for any incident circle of $\mathcal{P}(\mathbf{v}, \mathbf{v}')$, it has the capability to accommodate at least one agent. With this assumption, Theorem 3 can be explained as, if $A(\mathbf{v}', ran)$ is non-full, then it has an element $\exists \mathbf{c} \in A(\mathbf{v}, ran)$ that can move along $\mathcal{P}(\mathbf{v}, \mathbf{v}')$ to reach $A(\mathbf{v}', ran)$ and it does not touch any other cluster's agent during this movement.

Definition 1. For two areas $A(\mathbf{v}, ran)$ and $A(\mathbf{v}', ran')$, we call them connected if there exists a clean path $\mathcal{P}(\mathbf{v}, \mathbf{v}')$ that connects them.

Lemma 4. *If two areas* $A(\mathbf{v}, ran)$ *and* $A(\mathbf{v}', ran')$ *are connected, and at least one of them is non-full, then all of their elements can reach any position if* $\mathbf{p} \in A(\mathbf{v}, ran) \cup A(\mathbf{v}', ran')$
.

Lemma 5. *Suppose all clusters are connected. If there is one cluster whose area is not-full, or we can remove an agent from a full area to a new location without impacting the clearance of all the paths between the clusters, then we can find a solution to move all agents to their goals. Otherwise, there is no solution (i.e. path) for the current scenario.*

	# Agents	MAT	Clusters	Movements	Total
#1	40	0.24	0.410	2.35	3.001
#2	100	0.315	0.398	7.43	8.143
#3	100	0.270	0.331	7.21	7.811

Table I: *The performance of our algorithm on three challenging benchmarks (Benchmark #1, #2, #3. We highlight the running time (in seconds) of medial axis computation (MAT), cluster computation (Clusters), agent movements (Movement) and the total running time.*

The last lemma proves the completeness of our algorithm.

VII. Analysis

In this section, we analyze the computational complexity of our algorithm. The overall algorithm includes two parts, medial-axis computation and multi-agent motion planning. Since we use a sample-based approach to compute the medial axis, its complexity is $O(klogk)$ [31], for k samples. For path computation, assume we have n agents and we move one agent at a time and the agent's movement takes m steps to reach the goal position. In the worst case, $n-1$ agents may have to move temporarily for this one agent and that takes a steps. As a result, the overall complexity is bounded by $O(an^2)$. So the overall complexity is bounded by $O(klogk + an^2)$.

VIII. Implementation and Performance

In this section, we describe our implementation and highlight the performance on three different benchmarks. In the first benchmark, we compare our approach with a recent algorithm on continuous space centralized planning. [29]. The other two benchmarks represent challenging scenarios. We have implemented our algorithms in C++ on a Intel Core i7 CPU running at 3.30GHz with 16GB of RAM and running Windows 7. All of the timing results are generated on a single core. All of the running times are shown in the table I and we show the time spent in different stages of the algorithm.

Benchmark 1: We simulate the scenario shown in [29]. There are 40 agents in a rectangle scenario and we highlight their start and goal positions. We can compute all the paths in 3 seconds, as opposed to taking 311 seconds in [29].

Benchmark 2: We simulate a rectangle-shaped scenario with some complex-shaped obstacles. There are 100 agents in this scenario. Each agent randomly selects another agents' start position as its goal position. We assume that the agents' goal positions are not overlapping. We can compute collision free paths in about 8 sec.

Benchmark 3: We simulate a leaf-like scenario which has a complex boundary shape with a lot of edges. Prior methods will not work in such a scenario. There are 100 agents in this scene and each agent randomly pick another agents' start position as its goal position. We assume that the goal positions are not overlapping and our algorithm takes about 8 seconds.

IX. Limitations and Future Work

We present a novel method to compute collision-free paths for the homogeneous multi-agent motion planning with arbitrarily-shaped obstacles. We use the medial-axis of the workspace and our approach is resolution-complete. To the best of our knowledge, this is the first approach that can provide such guarantees for arbitrarily-shaped obstacles and we observe up to 100X speedup over prior methods. Our approach has some limitations. It is limited to homogeneous disc-like agents and does not take into account dynamics constraints. Furthermore, we can't provide any optimality guarantees. There are many avenues for future work. In addition to overcoming these limitations, we would like to design new methods based on medial-axis to simulate human-like behaviors or planning for high-DOF agents. Instead of moving one agent at a time, we would like to develop approaches that allow multiple agents to move simultaneously in complex scenarios.

References

[1] M. Katsev, J. Yu, and S. M. LaValle, "Efficient formation path planning on large graphs," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3606–3611.

[2] R. Luna and K. E. Bekris, "Push and swap: Fast cooperative path-finding with completeness guarantees," in *IJCAI*, 2011, pp. 294–300.

[3] M. Turpin, K. Mohta, N. Michael, and V. Kumar, "Goal assignment and trajectory planning for large teams of aerial robots." in *Robotics: Science and Systems*, 2013.

[4] A. Krontiris, R. Luna, and K. E. Bekris, "From feasibility tests to path planners for multi-agent pathfinding," in *Sixth Annual Symposium on Combinatorial Search*, 2013.

[5] I. Karamouzas, R. Geraerts, and A. F. van der Stappen, "Space-time group motion planning," in *Algorithmic Foundations of Robotics X*. Springer, 2013, pp. 227–243.

[6] M. Turpin, N. Michael, and V. Kumar, "Trajectory planning and assignment in multirobot systems," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 175–190.

[7] ——, "Concurrent assignment and planning of trajectories for large teams of interchangeable robots," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 842–848.

[8] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[9] N. Pelechano, J. M. Allbeck, and N. I. Badler, "Controlling individual agents in high-density crowd simulation," in *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 2007, pp. 99–108.

[10] H. Yeh, S. Curtis, S. Patil, J. van den Berg, D. Manocha, and M. Lin, "Composite agents," in *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2008, pp. 39–47.

[11] J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics research*. Springer, 2011, pp. 3–19.

[12] L. He and J. van den Berg, "Meso-scale planning for multi-agent navigation," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 2839–2844.

[13] A. Kimmel, A. Dobson, and K. Bekris, "Maintaining team coherence under the velocity obstacle framework," in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 247–256.

[14] R. Cui, B. Gao, and J. Guo, "Pareto-optimal coordination of multiple robots with safety guarantees," *Autonomous Robots*, vol. 32, no. 3, pp. 189–205, 2012.

[15] G. Sanchez and J.-C. Latombe, "Using a prm planner to compare centralized and decoupled planning for multi-robot systems," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 2112–2119.

[16] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," *The International Journal of Robotics Research*, vol. 24, no. 4, pp. 295–310, 2005.

[17] P. Velagapudi, K. Sycara, and P. Scerri, “Decentralized prioritized planning in large multirobot teams,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 4603–4609.

[18] B. de Wilde, A. W. ter Mors, and C. Witteveen, “Push and rotate: cooperative multi-agent path planning,” in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2013, pp. 87–94.

[19] Q. Sajid, R. Luna, and K. E. Bekris, “Multi-agent pathfinding with simultaneous execution of single-agent primitives.” in *SOCS*, 2012.

[20] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[21] R. E. Korf, “Depth-first iterative-deepening: An optimal admissible tree search,” *Artificial intelligence*, vol. 27, no. 1, pp. 97–109, 1985.

[22] A. Felner, M. Goldenberg, G. Sharon, R. Stern, T. Beja, N. R. Sturtevant, J. Schaeffer, and R. Holte, “Partial-expansion a* with selective node generation.” in *AAAI*, 2012.

[23] S. Carpin and E. Pagello, “On parallel rrts for multi-robot systems,” in *Proc. 8th Conf. Italian Association for Artificial Intelligence*, 2002, pp. 834–841.

[24] D. Ferguson, N. Kalra, and A. Stentz, “Replanning with rrts,” in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1243–1248.

[25] C. Ferner, G. Wagner, and H. Choset, “Odrm* optimal multirobot path planning in low dimensional search spaces,” in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 3854–3859.

[26] K. Dresner and P. Stone, “A multiagent approach to autonomous intersection management,” *Journal of artificial intelligence research*, vol. 31, pp. 591–656, 2008.

[27] G. Calinescu, A. Dumitrescu, and J. Pach, “Reconfigurations in graphs and grids,” *SIAM Journal on Discrete Mathematics*, vol. 22, no. 1, pp. 124–138, 2008.

[28] P. Švestka and M. H. Overmars, “Coordinated path planning for multiple robots,” *Robotics and autonomous systems*, vol. 23, no. 3, pp. 125–152, 1998.

[29] K. Solovey, J. Yu, O. Zamir, and D. Halperin, “Motion planning for unlabeled discs with optimality guarantees,” *CoRR*, vol. abs/1504.05218, 2015. [Online]. Available: http://arxiv.org/abs/1504.05218

[30] H. Blum and R. N. Nagel, “Shape description using weighted symmetric axis features,” *Pattern recognition*, vol. 10, no. 3, pp. 167–180, 1978.

[31] J. Giesen, B. Miklos, and M. Pauly, “The medial axis of the union of inner voronoi balls in the plane,” *Computational Geometry*, vol. 45, no. 9, pp. 515–523, 2012.

[32] R. L. Graham, B. D. Lubachevsky, K. J. Nurmela, and P. R. Östergård, “Dense packings of congruent circles in a circle,” *Discrete Mathematics*, vol. 181, no. 1, pp. 139–154, 1998.

[33] S. I. Galiev and M. S. Lisafina, “Linear models for the approximate solution of the problem of packing equal circles into a given domain,” *European Journal of Operational Research*, vol. 230, no. 3, pp. 505–514, 2013.