UNIVERSITY OF CALIFORNIA SAN DIEGO

Collision-Free Task Assignment and Trajectory Planning for Multi-Robot Systems

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Engineering Sciences (Aerospace Engineering)

by

Yasuhiro Toyoda

Committee in charge:

Professor Sonia Martínez, Chair
Professor Robert Bitmead
Professor Maurício de Oliveira

2020

The thesis of Yasuhiro Toyoda is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California San Diego

2020

DEDICATION

To Yuka, Keiji, and Karin, for always being there for me.

EPIGRAPH

For a successful technology, reality must take precedence over public relations,

for nature cannot be fooled.

—Richard P. Feynman

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to professor Sonia Martínez, you have been a tremendous mentor for me. She has given an understanding of the purpose I came here, which learns multi-agent systems, and suggested a suitable research topic. Furthermore, she proposed class selections to fulfill my goal, and I was able to learn a wide range of control theory from the basics to leading-edge trends. It's all because of her precise guidance that I had a meaningful experience and was able to grow significantly as a control engineer through this master's program.

I would also like to thank my other committee members, professor Robert Bitmead and professor Maurício de Oliveira for their time and valuable feedback.

I acknowledge Aamodh Suresh for engaging conversations and for giving me insightful feedback and suggestions. I sincerely wish him the greatest success with his doctoral program.

I would like to express my gratitude to Mitsubishi Heavy Industries for their financial support that enabled me to focus on my research and classwork. I would also like to appreciate Yoshihiro Sera, Motonari Tsunekawa, Kenji Hakamada, and Akira Watanabe for allowing me to study in the US.

Finally, I must express my very profound gratitude to my wife, Yuka, who supported me throughout this journey. To my kids, Keiji and Karin, I believe that they will come back here to study at UCSD in the future as they want.

VITA

2007	Bachelor of Engineering in Engineering Science, Osaka University, Japan
2009	Master of Engineering in Aeronautics and Astronautics, the University of Tokyo, Japan
2009-	Senior System Engineer, Mitsubishi Heavy Industries, Ltd.
2020	Master of Science in Mechanical and Aerospace Engineering, University of California San Diego

ABSTRACT OF THE THESIS

Collision-Free Task Assignment and Trajectory Planning for Multi-Robot Systems

by

Yasuhiro Toyoda

Master of Science in Engineering Sciences (Aerospace Engineering)

University of California San Diego, 2020

Professor Sonia Martínez, Chair

In this paper, we address the problem of assigning tasks and generating trajectories in a collision-free manner for multi-robot systems moving along a predefined roadmap. First, we propose centralized algorithms, which assign tasks to minimize the total travel distance or the collisions and then resolve the collisions with replanning the trajectory by local graph modification. Especially, to minimize the collisions in the initial task assignment, we formulate an optimization problem whose objective function includes the number of collisions based on a conflict graph that encodes all possible collisions among the system. We also develop a method to resolve remaining collisions in the initial task assignment. In this method, each robot has a graph

representing the roadmap, which is used to generate its trajectory. When a collision occurs on the trajectory, the robot modifies the graph to regenerate the trajectory that resolves the collision.

We then propose several decentralized algorithms, extending the centralized methods, which assign tasks initially before robots start moving in such a way to minimize the total travel distance, minimize collisions, or deploy randomly, and then avoid collisions with local graph modification. In the initial task assignment minimizing collisions, each robot calculates the expected value of collisions based on the local conflict graph, which encodes possible collisions among the neighborhood, and choose a task to minimize the expectation while coordinating to resolve a conflict of the assignment with the neighborhood.

The paper finally reports on simulations for systems of several tens of robots to evaluate the performance of the proposed centralized and decentralized algorithms.

Chapter 1

Introduction

1.1 Motivation

Coronaviruses have caused a tremendous impact on the global economy and have transformed the lives of people. The United States and a lot of other countries around the world have declared a state of emergency for COVID-19 and ordered stay at home and work from home to prevent the spread of infection, and people are living while maintaining a social distance. Even in such a situation, for essential work such as the healthcare, grocery stores, and delivery service, they are permitted to continue the business, and the safety of employees against infection is one of the most significant issues. In order to minimize the risk of infection for employees, to restart stagnant economic activities, and to reconstruct a more sustainable economic against such unprecedented situation, the coronavirus recession could bring about a spike to automation in such a way that most tasks that currently done by human workers are automated by utilizing robots[1]. Especially, the contribution of mobile robots covers a wide range of businesses, exemplified in Figure 1.1.

Mobile robots are robots that can move from one place to another autonomously, that is, without assistance from external human operators [2]. For example, many people are shifting

(a) (b) (c)

Figure 1.1: Mobile robots,(a)warehouse robot[5], (b)hospital robot[4], (c)driverless delivery vehicle[3]

to e-commerce to get essential goods instead of buying at physical stores due to coronaviruses outbreak. Along with that, the work in the warehouse has increased significantly. On the other hand, the spread of infection in warehouses has become a major issue, therefore the demand for autonomous mobile robots that transport inventory around the warehouses is increasing. The field of robotics is expanding in the medical field as well. Robot nurses are a good example, which provides appropriate treatment for patients while protecting doctors and nurses from infection [3]. Another type of robot in the hospital may deliver drugs and clean linens and meals to patients while carting away medical waste and soiled sheets [4]. Thus, various robots will be introduced in the medical field.

With increasing the demand to minimize contact with other people and to mitigate risks for such situations, the adoption of mobile robots will be accelerating in a wide range of business fields such as medical care, industry, and logistics. In an environment where multiple robots share a workspace, the robots are required to collaborate with others to achieve the whole tasks effectively as a group of robots, rather than independently performing only predetermined tasks. In comparison with single robots, multi-robot systems demonstrate several advantages: faster completion of all tasks by performing tasks in parallel, higher robustness against the faults of individual robots and small subgroups, and easier adaptation to a variety of different applications and missions. However, in order to realize the multi-robot system, we need to solve several

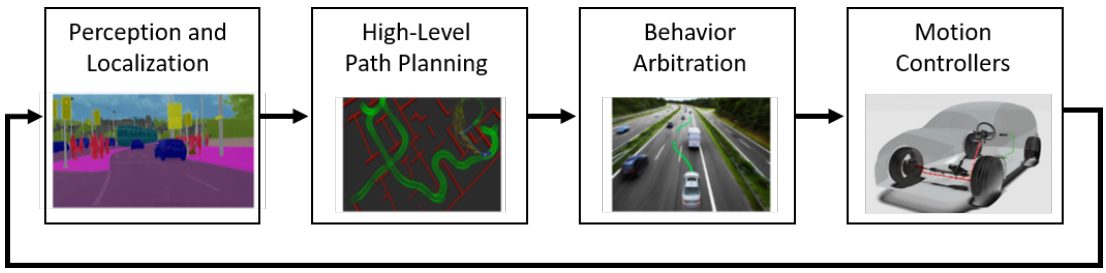


Figure 1.2: Block diagram of mobile robot

specific problems that do not appear in the considerations of the mobile robots acting individually: preserving the activity of the swarm as a unit, communication and interactions between the robots, collision avoidance, and allocation of tasks[6]. In this paper, we primarily consider the collision-free trajectory planning as well as task allocation.

An abstract block diagram of the autonomous mobile robot is shown in Figure 1.2. The robot processes observations coming from on-board sources such as cameras, radars, LiDARs, GPS, and/or inertial sensors to estimate its position and perceive the surrounding environment. Based on such information, each robot decides a task that will be accomplished and plans a trajectory to the task. Then, local actions are decided based on behavior arbitration [7]. Multi-robot systems can communicate and coordinate to make better decisions during each process. In this paper, we focus on collision avoidance in the path planning phase and consider the problem of task assignment and path planning for multi-robot systems without any collision.

1.2 Related Works

Several works can be found in the literature that addresses task assignment and trajectory planning in which robots do not collide. In [8], a centralized task assignment algorithm that makes use of the Hungarian algorithm under the assumption about the initial locations of robots and goals was proposed. The robot can move around N dimensional Euclidean space with no obstacle. With the assumption, the proposed algorithm can guarantee the solution is collision-free. In addition, a

decentralized algorithm is formulated based on their centralized algorithm. In the decentralized algorithm, robot exchanges assigned goals with other robots within the communication radius in such a way that the sum of distance squared for the pair of robots decreases. The work is extended in [9] to remove the assumption for the initial location and proposes a decentralized task assignment and trajectory generation based on methods related to switched systems. When a robot is about to result in colliding with others, it changes policy dynamically to avoid the collision, which is based on a Lyapunov-like barrier function.

In [10], authors consider the collision-free task assignment for multiple robots, moving along a predefined roadmap. The conflict graph which encodes possible collision over the system is introduced, and tasks are assigned to robots solving an optimization problem, which minimizes total cost under the constraints collected in the conflict graph. The Dijkstra's algorithm [11] is utilized for generating the shortest path on the roadmap, and they do not consider replanning the path to dodge the collisions. Therefore, in some specific cases, the optimization problem does not have a solution, that is, no matter how tasks are assigned, it includes collisions.

In the literature, several collision avoidance approaches are provided based on local coordination of robot motion. The coordination method for collision avoidance and deadlock detection/resolution is proposed in [12]. If a possible collision is detected, robots start to negotiate, and if necessary, one of the robots has idle time to avoid the collision. In [13], [14], the decentralized robot motion policy assumes a more realistic situation such that robots have practical dynamics preventing immediate stops or not allowing stops at all. In [15], this work introduces a collision avoidance method for aerial robots using altitude control. These works successfully avoid collisions in euclidean space. Assuming a roadmap environment, they should be allowed that robots deviate from the roadmap to avoid collisions.

Compared to the centralized multi-robot systems, the decentralized systems offer many potential advantages: robustness in terms of a single point of failure, computational resources, scalability, and flexibility of the system configuration. On the other hand, they involve challenging

issues such as coordination among robots and the handling of the distributed information. In [16], a distributed version of the Hungarian algorithm is developed considering a connected network in which each robot broadcasts information over the system.

Several works introduce market-based coordination approaches to solve the task assignment problem, which are characterized by the absence of a centralized decision-maker. [17] utilizes the contract net protocol, where task-swap is the only possible type of contract. Given initial task assignments, each robot coordinates according to the protocol and exchanges the task with a neighborhood if decreasing the cost. Decentralized task assignment algorithms exist in [18], [19], in which every task is occupied by at least one robot without being given an initial task assignment. The first work introduces a consensus-based auction approach(CBAA), in which each robot places bids on tasks, and the highest bid wins the assignment. In general, the auction algorithm requires an auctioneer to receive and evaluate bids from the neighborhood. The CBAA removes the auctioneer by combining the consensus approach to resolve conflicts of the winner. In [19], a distributed coordination algorithm based on the auction approach is proposed. Each auction is performed among neighborhoods, and a robot with the highest bid is assigned to the task. Therefore, although it is guaranteed that each robot eventually reaches a task without conflicts of assignment, this algorithm allows that robots are assigned to the same task.

1.3 Contribution

In this paper, we consider the problem of assigning tasks and generating trajectories in a collision-free manner for multi-robot systems moving along a predefined roadmap. First, we propose centralized algorithms, characterized by a two-phase approach, leading a solution that includes no collision while minimizing the total travel distance. The first phase is to determine the task assignment, which minimizes collisions over the system. To realize such a task assignment, we formulate this problem as an integer linear programming, whose objective functions are not

only the total distance but also the number of collisions encoded utilizing the conflict graph. Tasks are assigned to robots, solving this optimization problem. We also develop an algorithm to resolve remaining collisions in the assignment of the first phase. In this algorithm, each robot has a graph representing the roadmap, which is used to generate the shortest path. When a collision occurs on the trajectory, the robot modifies the graph to regenerate the trajectory, which avoids the collision.

Furthermore, decentralized algorithms are developed based on the knowledge of the centralized system. Robot calculates the expectation of the collisions for each task as an objective function, making use of the local conflict graph that encodes the collision with the neighborhood. A variable of the objective function is only its assignment, not related to assignments for other robots. Therefore, conventional market-based coordination approaches can be exploited to assign tasks that minimize each robot's objective function. After start moving to the assigned goal, robots keep communicating with their neighborhood and updating trajectories dynamically with the graph modification algorithm to avoid the collision. The proposed decentralized algorithm guarantee that all robots eventually reach to their goal without collisions.

1.4 Organization

This paper is organized as follows. Section 2 introduces preliminary notions, and the basis of graph theory related to this paper. Section 3 presents the mathematical formulation of the problem, which is addressed with the centralized and decentralized approaches. In Sections 4 and 5, the centralized and decentralized algorithm are proposed respectively. The simulation results for proposed algorithms are provided in Section 6. Section 7 summarizes our conclusions and future works.

Chapter 2

Preliminary

Let $\mathbb{R}$, $\mathbb{R}_{>0}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{Z}_{\geq 0}$ denote the set of reals, positive reals, non-negative reals, and non-negative integers, respectively. The transpose of a matrix $\mathbf{A}$ is $\mathbf{A}^T$. The identity matrix of size n is denoted by $\mathbf{I}_n$. The cardinality of a set V is the number of elements of the set, given by $|V|$. Let $\mathbf{1}_n$ be n-dimensional vector where every element is equal to one.

Graph Theory

A weighted graph $\mathcal{G} = (V, E)$ consists of a set of vertices V and a set of edges $E \subseteq V \times V$, and each edge of the graph has an associated numerical value, called a weight. The vertex set of a graph $\mathcal{G}$ is reffered to as $V(\mathcal{G})$, its edge set as $E(\mathcal{G})$. For $u, v \in V$ and $u \neq v$, the pair (u, v) denotes an edge from u to v. For an undirected graph, if $(u, v) \in E$, there exists (u, v) anytime, and the vertices u and v are adjacent, or neighboring.

Given a graph $\mathcal{G} = (V, E)$ with $V = \{v_1, v_2, \cdots, v_n\}$ and $E = \{1, 2, \cdots, e_m\}$, the adjacency matrix $\mathcal{A}(\mathcal{G})$ is defined as a $|V| \times |V|$ matrix whose (i, j)-th element $a_{i,j}$ given by:

$$a_{i,j} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E(\mathcal{G}) \\ 0 & \text{otherwise} \end{cases}$$

The unoriented incidence matrix $\mathcal{I}(\mathcal{G})$ is a $|V| \times |E|$ matrix whose (i,k)-th element $\mathfrak{u}_{i,k}$ given by:

$$\mathfrak{u}_{i,k} = \begin{cases} 1 & \text{if } v_i \in e_k \\ 0 & \text{otherwise} \end{cases}$$

where, $v \in e$ means that v is the vertex of the edge e.

The set of vertices which adjacent to a vertex $v \in V(\mathcal{G})$ is denoted by $\mathcal{N}_{\mathcal{G}}(v)$, and the degree of v is defined as $d(v) = |\mathcal{N}_{\mathcal{G}}(v)|$. We represent the set of all possible subgraph of $\mathcal{G}$ related to edges as $G = \{(V,\overline{E})|\overline{E} \subseteq E\}$. A path (or trajectory) in a graph is an ordered sequence of vertices such that any pair of consecutive vertices in the sequence is an edge of the graph, A graph is connected if there exists a path between any two vertices, and otherwise it is disconnected. For a weighted graph, the length of a path is the sum of the weights of the edges in the path. The distance $d_{\mathcal{G}}(u,v)$ denotes the length of a shortest path between u and v in $\mathcal{G}$, and Appendix A shows an algorithm for finding the shortest path between an arbitrary pair of vertices. [20],[21],[22].

In this paper, we describe that a $n \times m$ grid graph is a graph whose vertices correspond to the points on two-dimensional Euclidean space with integer coordinates, x-coordinates being in the range $1,2,\cdots,m$ and y-coordinates being in the range $1,2,\cdots,n$, and two vertices are connected by an edge whenever the corresponding points are at Euclidean distance 1, as shown in Figure 2.1.

Multi-robot Systems on Predefined Roadmap

We consider N robots system that robots move on a grid graph $\mathcal{G} = (V,E)$ with constant unit speed 1 and change the direction on only vertices, that is, are located on vertices at each discrete time $t \in \mathbb{Z}_{\geq 0}$. Let $x_i(t) \in V$ for $i \in \{1,2,\cdots,N\}$ denote the vertex where the robot occupies at t. The system state vector is given by:

$$X(k) = \{x_1(k), x_2(k), \cdots, x_N(k)\} \in V^N$$

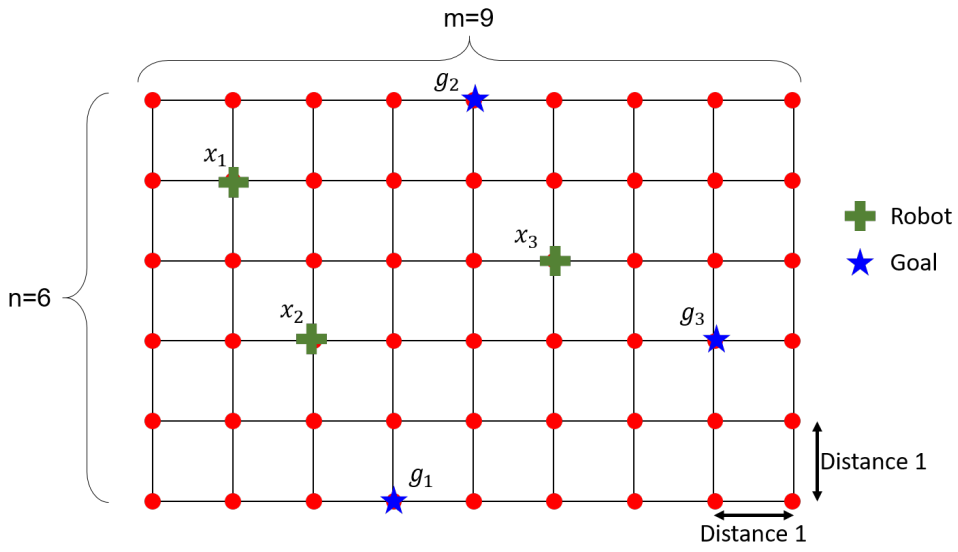


Figure 2.1: $n \times m$ grid graph ($n = 6$, $m = 9$, $N = M = 3$)

Given M stationary goals to be accomplished by the robots, let $g_j \in V$ for $j \in \{1, 2, \cdots, M\}$ denote the vertex where the goal is located. The system goal state vector is:

$$T = \{g_1, g_2, \cdots, g_M\} \in V^M$$

Let $\xi_{i,j} = \{\xi^0_{i,j}, \xi^1_{i,j}, \cdots, \xi^{\Delta_{i,j}}_{i,j}\}$ denote a trajectory from robot i to goal j. The element $\xi^t_{i,j} \in V(\mathcal{G})$ represents the location of robot i which moves to goals j at the discrete time t, and $\xi^0_{i,j} = x_i(0)$, $\xi^{D_{i,j}}_{i,j} = g_j$. In addition, the length of the trajectory $\xi_{i,j}$ is written as $l(\xi_{i,j}) = \Delta_{i,j}$. If assigned goal $\mathbf{f}_i$ of robot i is decided, we simply write ξ_i instead of $\xi_{i,\mathbf{f}_i}$, and let $\xi = \{\xi_i \mid i \in \{1, 2, \cdots, N\}\}$ be the set of trajectories for all robots.

Chapter 3

Problem Definition

Let us consider N robots with no size which move on a grid graph $\mathcal{G} = (V, E)$ and M goals. All robots are homogeneous and interchangeable with no preference of goals, and do not have any sensing or actuation error. Let $\phi_{i,j} \in \{0, 1\}$ be the decision variable that indicates whether or not goal j is assigned to robot i:

$$\phi_{i,j} = \begin{cases} 1 & \text{if robot } i \text{ is assigned to goal } j \\ 0 & \text{otherwise} \end{cases}$$

Furthermore, we define the stacked vector of the decision variables related to robot i as $\phi_i = [\phi_{i,1}, \phi_{i,2}, \cdots, \phi_{i,N}]^T$ and the stacked vector over the system as $\phi = [\phi_1^T, \phi_2^T, \cdots, \phi_N^T]^T$. Assuming the number of goals is the same as robots, that is $M = N$, we require each robot is assigned to each goal without duplication, which results in $\sum_{i=1}^{N} \phi_{i,j} = 1, \forall j \in \{1, 2, \cdots, N\}$ and $\sum_{j=1}^{N} \phi_{i,j} = 1$, $\forall i \in \{1, 2, \cdots, N\}$. Let $\mathbf{f}_i$ be the goal assigned to robot i, in other words, $\mathbf{f}_i = \{j \mid \phi_{i,j} = 1, \forall j\}$.

Definition 1. *Let $\xi_{i,j}$ be the trajectory from robot $i \in \{1, 2, \cdots, N\}$ to goal $j \in \{1, 2, \cdots, N\}$. The trajectories $\xi_{i,j}$ and $\xi_{k,h}$ collide if and only if either of the following conditions is true:*
$\forall t \in \mathbb{Z}_{\geq 0}$,

Algorithm 1: Collision-free Check between Trajectories $\xi_{i,j}$ and $\xi_{k,h}$

Input : $\xi_{i,j}$, $\xi_{k,h}$
Output : True or False

$t_{min} \leftarrow \min(l(\xi_{i,j}),\ l(\xi_{k,h}))$
// Check Definition 1(a)
if $\xi^t_{i,j} = \xi^t_{k,h} \quad t \in \{0,1,\cdots,t_{min}\}$ **then**
  return False
end
// Check Definition 1(b)
if $\xi^{t+1}_{i,j} = \xi^t_{k,h}$ *and* $\xi^t_{i,j} = \xi^{t+1}_{k,h} \quad t \in \{0,1,\cdots,t_{min}\}$ **then**
  return False
end
return True

a) $\xi^t_{i,j} = \xi^t_{k,h}$ *, or*

b) $\xi^{t+1}_{i,j} = \xi^t_{k,h}$ *and* $\xi^t_{i,j} = \xi^{t+1}_{k,h}$.

Otherwise, they are collision-free.

As described in the definition, to ensure the collision-free, we require two conditions; a)any two robots must not occupy the same vertex at the same time, and b)any robots must not move on the same edge at the same time. Algorithm 1 illustrates the pseudo code to check collision-free between a pair of trajectories.

In this paper, we address the following problem:

Problem 1. *The objective is to find an optimal set of trajectories* $\xi^* = \{\xi^*_1, \xi^*_2, \cdots, \xi^*_N\}$ *which do not collide each other while minimizing the sum of the length for all robots, given by:*

$$\xi^* = \underset{\xi}{\operatorname{argmin}} \sum_{i=1}^{N} l(\xi_i) \tag{3.1}$$

subject to:

ξ_i *and* ξ_k *are collision-free* $\forall i,k \in \{1,2,\cdots,N\}$

Chapter 4

Centralized Task Assignment and Trajectory Planning

4.1 Conceptual Approach

In this section, we consider a centralized algorithm to solve Problem 1. The centralized system requires that robots always have perfect state knowledge over the system, such as locations of all robots and goals, and goal assignment for all robots.

Problem Formulation with Graph Modification

The basic idea to find collision-free trajectories is that when a collision occurs, robots replan the trajectories to avoid passing through the collision point. The collision point represents either a vertex or an edge where robots will collide. In order to realize the idea, we introduce a modified graph that some edge is removed from the roadmap graph $\mathcal{G}$. The shortest path on the modified graph, which is computed by Dijkstra's algorithm, does not path through the removed edge. Furthermore, it is possible to generate a trajectory that does not include the collisions point by modifying the graph appropriately.

Let us consider a situation that two robots 1 and 2 collide, shown in Figure 4.1(a). To avoid the collision, robot 1 replans trajectories in the modified graph removed an edge, which includes the collision point, resulting in robot 1 has trajectories that avoid passing the collision point(see (b)). Then robots find the optimal set of trajectories to minimize the sum of length, as shown in (c).

We generalize the approach that utilizes the graph modification to manipulate the trajectory so that it can deal with multiple robots and goals, and reformulate Problem 1 for the centralized system. Let $\mathrm{G} = \{(V, E_l) | E_l \subseteq E \ \ l \in \{1, 2, \cdots, 2^{|E|}\}\}$ denote the set of subgraph related to edges, which is the set of all possible modified graphs. The modified graph of robot i is denoted as γ_i, and the set of modified graphs for the system is $\gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_N\}$. Let $D_{i,j}(\gamma_i)$ be the distance from robot i to goal j on the modified graph γ_i, that is, $D_{i,j}(\gamma_i) = d_{\gamma_i}(x_i, g_j)$. Given robot i's assignment ϕ_i, the length of trajectory to assinged goal $l(\xi_i)$ can be rewritten as:

$$l(\xi_i) = \sum_{j=1}^{N} \phi_{i,j} D_{i,j}(\gamma_i) \tag{4.1}$$

and since the trajectory can be manipulated by the modified graph, we write $\xi_i = \xi_i(\gamma_i)$.

We reformulate Problem 1 based on the approach of graph modification as follows:

Problem 2. *The objective is to find the set of modified graphs γ^* and task assignment ϕ^* which result in the collision-free while minimizing the total travel distance:*

$$\gamma^*, \phi^* = \underset{\gamma, \phi}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{N} \phi_{i,j} D_{i,j}(\gamma_i) \tag{4.2}$$

subject to:

$\xi_i(\gamma_i)$ *and* $\xi_k(\gamma_k)$ *are collision-free.* $\forall i, k \in \{1, 2, \cdots, N\}$

Considering the straightforward way to solve Problem 2, it is a brute-force approach, as shown in Algorithm 2. This algorithm solves the optimization problem repeatedly for every

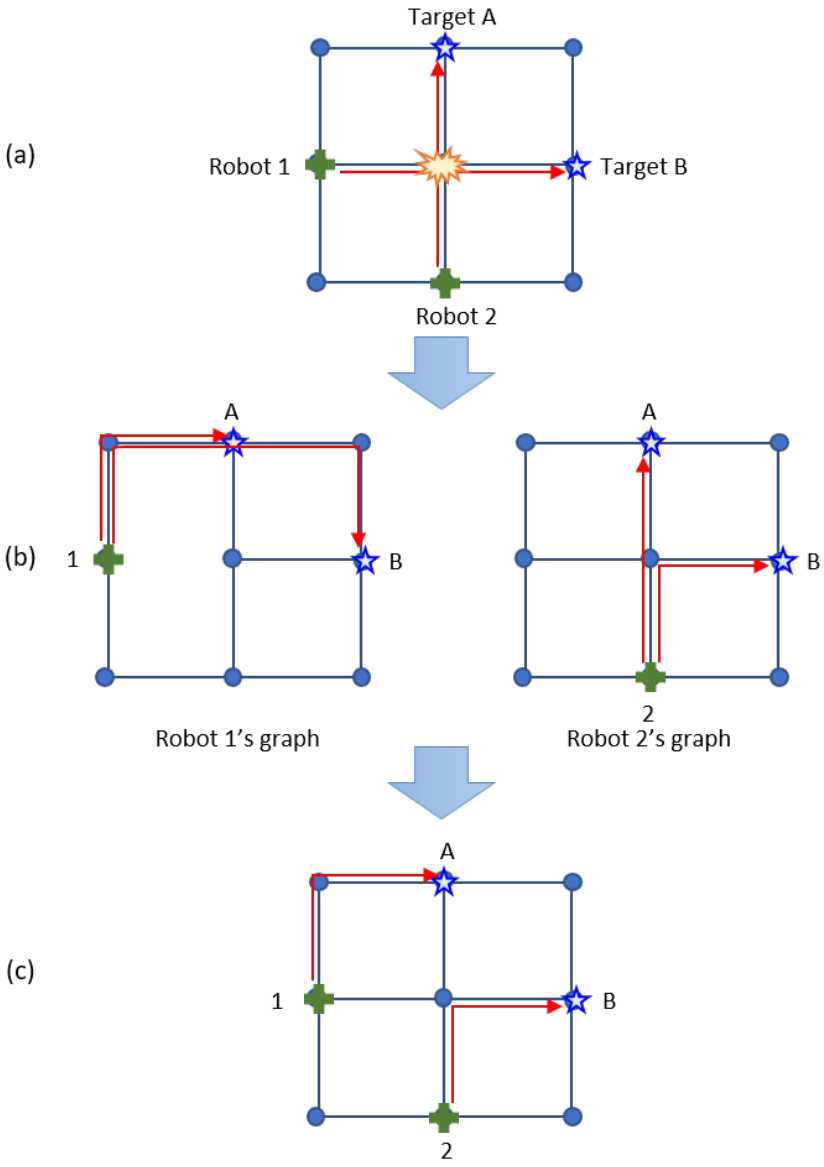


Figure 4.1: An conceptual idea to find the collision-free trajectories.(a) Target assignment and trajectories based on the original graph, but collision occurs. (b) Left: Robot 1's trajectories on the modified graph. Right: Robot 2's trajectories on the original graph. (c) Collision-free assignment and trajectories

possible modified graph, and find the desired task assignment and modified graph comparing each solution. Since each robot has $2^{|E|}$ possible modified graph, the algorithm requires to solve the optimization problem $2^{N|E|}$ times, and its time-complexity is estimated as $\mathcal{O}(2^{N|E|}(|E|N^2 + N^3 + |V|\log|V|))$. It is almost impossible to solve within a realistic time.

Algorithm 2: Centralized Brute-force Graph Modification Algorithm

Input : $\mathcal{G} = (V,E), X(0), T$
Output : $\xi_i, \mathbf{f}_i,\ i \in \{1,2,\cdots,N\}$

$J_{min} \leftarrow +\infty$
$\gamma(l) \leftarrow \{(V,E_l) | E_l \subseteq E,\ l \in \{1,2,\cdots,2^{|E|}\}\}$
for $p_1 \in \{1,\cdots,2^{|E|}\}$ **do**
  $\gamma_1 \leftarrow \gamma(p_1)$
  compute $D_{1,j}\ j \in \{1,\cdots,N\}$ by Algorithm 8 with γ_1
  for $p_2 \in \{1,\cdots,2^{|E|}\}$ **do**
    $\gamma_2 \leftarrow \gamma(p_2)$
    compute $D_{2,j}\ j \in \{1,\cdots,N\}$ by Algorithm 8 with γ_2
    $\vdots$
    for $p_N \in \{1,\cdots,2^{|E|}\}$ **do**
      $\gamma_N \leftarrow \gamma(p_N)$
      compute $D_{N,j}\ j \in \{1,\cdots,N\}$ by Algorithm 8 with γ_N
      $D \leftarrow [D_{i,j}] \quad i \in \{1,\cdots,N\}, j \in \{1,\cdots,N\}$
      compute ϕ^* by Hungarian algorithm with D
      $J \leftarrow \sum_{i=1}^{N} \sum_{j=1}^{N} \phi^*_{i,j} D_{i,j}$
      $\overline{\mathbf{f}_i} \leftarrow$ pick the index of ith robot's goal from ϕ, $i \in \{1,2,\cdots,N\}$
      $\overline{\xi_i} \leftarrow$ trajectory from x_i to $g_{\mathbf{f}_i}$ in γ_i by Algorithm 8 and 9, $i \in \{1,2,\cdots,N\}$
      if $J < J_{min}$ *and collision-free(Algorithm 1) for every pair of* $\overline{\xi_i}$ *and* $\overline{\xi_j}$ **then**
        $\xi_i \leftarrow \overline{\xi_i}\ \ i \in \{1,2,\cdots,N\}$
        $\mathbf{f}_i \leftarrow \overline{\mathbf{f}_i}\ \ i \in \{1,2,\cdots,N\}$
      end
    end
  end
end
return $\xi_i, \mathbf{f}_i \quad i \in \{1,2,\cdots,N\}$

Conceptual Approach for Centralized System

To reduce the time-complexity, instead of comparing all possible modified graphs, we propose a two-step approach, as shown in Figure 4.2. The first step is to decide an initial task assignment by solving an optimization problem, and we consider Minimum Distance Task Assignment(MDTA) and Minimum Collision Task Assignment(MCTA). MDTA assigns goals in a way to minimize the total travel distance, which is one of the most general assignment

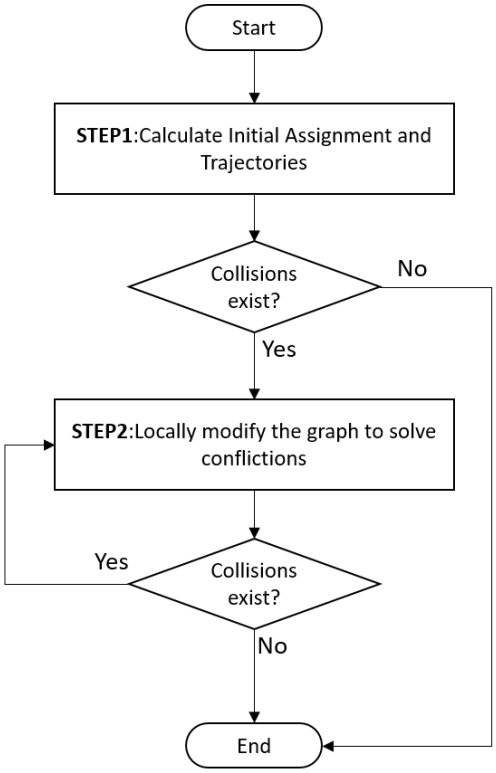


Figure 4.2: Conceptual approach for centralized system

problem as summarized in Appendix B.1. In this paper, we utilize the Hungarian algorithm to solve the MDTA problem. In Section 4.2, we formulate the MCTA problem as an integer linear programming. In this step, collision avoidance is not taken into consideration, and therefore the initial assignment and trajectory may include collisions. Once we get the initial assignment and trajectory, we resolve each collision in such a way to modify the graph locally, which is presented in Section 4.3.

4.2 Minimum Collision Task Assignment

MDTA does not consider about collision and leaves all efforts for collision-free up to the subsequent graph modification phase. For graph modification, It is anticipated that the more edges are removed, the more a deadlock situation is likely to happen. Therefore, we develop

another approach, MCTA, which finds task assignment with fewer collisions as possible while minimizing the sum of distance. With MCTA, the extent of graph modification can be reduced.

Conflict Graph

We first define a graph to identify all the possible collisions in the system, which is introduced in [10],[23]. Let $\mathcal{S}$ be the conflict graph, whose vertex $v_{i,j} \in V(\mathcal{S})$ represents an assignment of robot i to goal j. The cardinality of $V(\mathcal{S})$ is $|V(\mathcal{S})| = N^2$. The edge between $v_{i,j}$ and $v_{k,h}$ is denoted as $(v_{i,j}, v_{k,h})$, which implies that robot i assigned to goal j collides with robot k assigned to goal h, that is, $\xi_{i,j}$ and $\xi_{k,h}$ collide. In our problem, each robot cannot be assigned more than one goal, and each goal cannot be assigned more than one robot. Therefore, $(v_{i,j}, v_{i,h}) \notin E(\mathcal{S}),\ \forall j,h \in \{1,2,\cdots,N\}$ and $(v_{i,j}, v_{k,j}) \notin E(\mathcal{S}),\ \forall i,k \in \{1,2,\cdots,N\}$.

ILP with Incidence Matrix of Conflict Graph

The MCTA problem is formulated with extending the conflict-free task assignment in Appendix B.2. Let $\mathcal{I}(\mathcal{S})$ be the unoriented incidence matrix of the conflict graph $\mathcal{S}$. Given the assignment ϕ, the existence of collisions is encoded by a vector $b_I(\phi) = \mathcal{I}^T(\mathcal{S})\phi \in \{0,1,2\}^{|E(\mathcal{S})|}$. According to the definition of conflict graph and unoriented incidence matrix, if $b_{Ik} = 2$, the collision represented by the edge of conflict graph corresponding to kth column of $\mathcal{I}(\mathcal{S})$. Therefore, the number of collision in the system can be identified by counting up the element which is $b_{Ik} = 2,\ \forall k \in \{1,2,\cdots,|E(\mathcal{S})|\}$, which is formulated as $\sum_{k=1}^{|E(\mathcal{S})|} R(b_{Ik}(\phi) - 1)$. $R(*)$ is the ramp function($\mathbb{R} \to \mathbb{R}_{>0}$) defined as the mean of the independent variable and its absolute value, $R(x) := x + |x|\frac{x+|x|}{2}$. Therefore, the number of collision is written as:

$$h(x) = \sum_{k=1}^{|E(\mathcal{S}|} \frac{|b_{Ik}(x) - 1| + b_{Ik}(x) - 1}{2} \tag{4.3}$$

The objective of MCTA is to not only minimize the number of collision, but also minimize

the sum of distance. We define the objective function of the sum of distance as the follows:

$$f(\phi) = \sum_{i=1}^{N} \sum_{j=1}^{N} D_{i,j}\phi_{i,j} \tag{4.4}$$

where, $D_{i,j}$ represents the distance from robot i to goal j on the roadmap graph $\mathcal{G}$. Since there are two objective functions, the problem is classified to the multi-objective optimization problem, and we transform the multiple objectives into an aggregate objective function by multiplying each objective function by a weighting factor and summing up all weighted objective functions. The optimization problem of MCTA is formulated as follows:

$$\text{minimize:} \qquad J(\phi) = w_1 f(\phi) + w_2 h(\phi) \tag{4.5}$$

$$\text{subject to:} \qquad \sum_{i=1}^{N} \phi_{i,j} = 1, \qquad j \in \{1, 2, \cdots, N\} \tag{4.6}$$

$$\sum_{j=1}^{N} \phi_{i,j} = 1, \qquad i \in \{1, 2, \cdots, N\} \tag{4.7}$$

$$\phi \in \{0, 1\}^N \tag{4.8}$$

We linearize this problem in such way that the absolute value in the objective function $h(\phi)$ will be reformulated into two linear expressions. Let $z_k = |b_{Ik} - 1|$ be a new variable to relax the absolute value, and it can be broken down to two inequalities given by: $\forall k \in \{1, 2, \cdots, |E(\mathcal{S})|$

$$b_{Ik} - 1 \leq z_k \tag{4.9}$$

$$b_{Ik} - 1 \geq -z_k \tag{4.10}$$

and the objective function $h(x)$ is reformulated using two variables as given by:

$$h(\phi, z) = \sum_{k=1}^{|E(\mathcal{S})|} \frac{z_k + b_{Ik}(\phi) - 1}{2} \tag{4.11}$$

The optimization problem can then be rewritten as the integer linear programming problem given by:

$$\text{minimize:} \qquad J(\phi, z) = w_1 f(\phi) + w_2 h(\phi, z) \tag{4.12}$$

$$\text{subject to:} \qquad b_{Ik} - 1 \leq z_k, \qquad k \in \{1, 2, \cdots, |E(\mathcal{S})|\} \tag{4.13}$$

$$b_{Ik} - 1 \geq -z_k, \qquad k \in \{1, 2, \cdots, |E(\mathcal{S})|\} \tag{4.14}$$

$$\sum_{i=1}^{N} \phi_{i,j} = 1, \qquad j \in \{1, 2, \cdots, N\} \tag{4.15}$$

$$\sum_{j=1}^{N} \phi_{i,j} = 1, \qquad i \in \{1, 2, \cdots, N\} \tag{4.16}$$

$$\phi \in \{0, 1\}^N \tag{4.17}$$

$$z \in \{0, 1\}^{|E(\mathcal{S})|} \tag{4.18}$$

Solving this ILP problem on the computer, we realized that while the problem could be solved in a few second, it took a long time to set the problem. This is because the summation of $h(\phi, z)$ in (4.11) blows up as the number of robots and goals increase, and the number of edges of the conflict graph $\mathcal{S}$ reach at most N^4. We reformulate the number of collision in a way to use the adjacency matrix of the conflict graph $\mathcal{S}$.

ILP with Adjacency Matrix of Conflict Graph

Let $A(\mathcal{S}) = (a_{(i,j),(k,h)})$ be the $N^2 \times N^2$ adjacency matrix of the conflict graph $\mathcal{S}$, and according to the definition of the adjacency matrix, given the assignment ϕ, the number of collisions which happens on a path from robot i to goal j is given by:

$$b_{i,j}(\phi) = \sum_{k=1}^{N} \sum_{h=1}^{N} a_{(i,j),(k,h)} \phi_{k,h} \tag{4.19}$$

and therefore, the total number of collisions in the system is given by:

$$h_A(\phi) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} b_{i,j}(\phi)\phi_{i,j} \tag{4.20}$$

Next, we linearize (4.20) since it is the nonlinear function for variables ϕ. We consider to identify a linear function:

$$f(b_{i,j},\phi_{i,j}) = \begin{cases} b_{i,j} & \text{if } x_{i,j} = 1 \\ 0 & \text{otherwise} \end{cases} \tag{4.21}$$

Each assignment has at most $N-1$ collisions, in other words, the upper bound of $b_{i,j}$ is $N-1$. From this fact, $b_{i,j} - N(1-\phi_{i,j})$ is always negative for $\phi_{i,j} = 0$, and clearly shows $b_{i,j}$ for $\phi_{i,j} = 1$. With the ramp function, we can realize the linear function (4.21) as $R(b_{i,j} - N(1-\phi_{i,j}))$, and (4.20) is linearized as the following:

$$\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} R(b_{i,j} - N(1-\phi_{i,j})) \tag{4.22}$$

Since (4.22) has absolute value in the ramp function, we reformulate in similar way to linearize (4.11), as shown by:

$$h_A(\phi, z) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \frac{z_{i,j} + b_{i,j} - N(1-\phi_{i,j})}{2} \tag{4.23}$$

$$z_{i,j} = |b_{i,j} - N(1-\phi_{i,j})| \tag{4.24}$$

The MCTA problem can be formulated as the multi-objective Integer Linear Programming:

$$\text{minimize:} \qquad J_A(\phi, z) = w_1 f(\phi) + w_2 h_A(\phi, z) \tag{4.25}$$

$$\text{subject to:} \qquad b_{i,j} - N(1-\phi_{i,j}) \leq z_{i,j}, \qquad i,j \in \{1,2,\cdots,N\} \tag{4.26}$$

$$b_{i,j} - N(1-\phi_{i,j}) \geq -z_{i,j}, \qquad i,j \in \{1,2,\cdots,N\} \tag{4.27}$$

$$\sum_{i=1}^{N} \phi_{i,j} = 1, \qquad j \in \{1,2,\cdots,N\} \tag{4.28}$$

$$\sum_{j=1}^{N} \phi_{i,j} = 1, \qquad i \in \{1,2,\cdots,N\} \tag{4.29}$$

$$\phi \in \{0,1\}^N \tag{4.30}$$

This ILP can solve much faster than the former ILP (4.12), because $h_A(\phi, z)$ requires to sum N^2 terms, which is less than N^4 of $h(\phi, z)$.

To get the desirable solution, which prioritizes to minimize the number of collision, we set the weighting factor as $w_1 << w_2$.

Alternative Form of Linearization : We also introduce other way to linearize (4.20). Non-linearity of this equation comes from the product of two binary product $\phi_{i,j} \times \phi_{k,h}$. Let $y_{i,j,k,h} = \phi_{i,j} \times \phi_{k,h} \in 0,1$ be new binary variable, and (4.20) can be linearized as:

$$h_A(y) = \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} \sum_{h=1}^{N} a_{(i,j),(k,h)} y_{i,j,k,h} \tag{4.31}$$

Since possible combinations of $(\phi_{i,j}, \phi_{k,h}, y_{i,j,k,h})$ are $(0,0,0)$, $(1,0,0)$, $(0,1,0)$, and $(1,1,1)$, we add some constraint for $y_{i,j,k,h}$: $\phi_{i,j} - y_{i,j,k,h} \geq 0$ and $\phi_{k,h} - y_{i,j,k,h} \geq 0$ ensure that $y_{i,j,k,h}$ is zero if either $\phi_{i,j}$ or $\phi_{k,h}$ is zero, $1 - \phi_{i,j} - \phi_{k,h} + y_{i,j,k,h} \geq 0$ ensures that $y_{i,j,k,h}$ takes 1 if both binary

variables are set to 1. Thus, the integer linear programming can be formulated as:

$$\text{minimize:} \qquad J_A(\phi) = w_1 f(\phi) + w_2 h_A(y) \tag{4.32}$$

$$\text{subject to:} \qquad \sum_{i=1}^{N} \phi_{i,j} = 1, \qquad j \in \{1,2,\cdots,N\} \tag{4.33}$$

$$\sum_{j=1}^{N} \phi_{i,j} = 1, \qquad i \in \{1,2,\cdots,N\} \tag{4.34}$$

$$1 - \phi_{i,j} - \phi_{k,h} + y_{i,j,k,h} \geq 0, \qquad i,j,k,h \in \{1,2,\cdots,N\} \tag{4.35}$$

$$\phi_{i,j} - y_{i,j,k,h} \geq 0, \qquad i,j \in \{1,2,\cdots,N\} \tag{4.36}$$

$$\phi_{k,h} - y_{i,j,k,h} \geq 0, \qquad k,h \in \{1,2,\cdots,N\} \tag{4.37}$$

$$\phi \in \{0,1\}^N \tag{4.38}$$

$$y \in \{0,1\}^{N^4} \tag{4.39}$$

This ILP problem deals with much more variables and constraints than foregoing one. Therefore, in this paper, we solve (4.25) to get the minimum collision solution.

4.3 Local Graph Modification for Collision Avoidance

After accomplishing the initial task assignment by solving optimization problems, we resolve every collisions in a way to modify the graph locally and regenerate the trajectory, as shown in Algorithm 3.

The local graph modification first identify a pair of robots which have a collision. Assuming robot i and k collide, let p_{col} be a location of the collision on trajectories ξ_i and ξ_k, which can be identified by similar method to Algorithm 1. In order to avoid the collision p_{col}, two actions are taken into consideration; 1) exchanging assigned goals between robot i and k and 2) modifying the graph of either robot i or k which is used for computing the path. Considering the combination of the two actions, there are six options p as shown in Table 4.1. The local graph

Algorithm 3: Centralized Local Graph Modification Algorithm

Input : ξ_i, $\mathbf{f}_i$, γ_i, $\forall i \in \{1,2,\cdots,N\}$
Output : ξ_i, $\mathbf{f}_i$, γ_i, $\forall i \in \{1,2,\cdots,N\}$

while *collision exists in the system* **do**
  $i,k \leftarrow$ a pair of robots which collide
  $p_{col} \leftarrow$ collision point between ξ_i and ξ_k
  $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow$run Algorithm 4 (ξ_i, ξ_k, p_{col})
  $\xi_i \leftarrow \xi_{i,\mathbf{f}_i}(\gamma_i)$
  $\xi_k \leftarrow \xi_{k,\mathbf{f}_k}(\gamma_k)$
end

option	Assigned goal		Graph	
p	for robot i	for robot j	for robot i	for robot j
1	$\mathbf{f}_i$	$\mathbf{f}_j$	γ_i	γ_j
2	$\mathbf{f}_j$	$\mathbf{f}_i$	γ_i	γ_j
3	$\mathbf{f}_i$	$\mathbf{f}_j$	$\gamma_{i,\mathbf{f}_i}$	γ_j
4	$\mathbf{f}_i$	$\mathbf{f}_j$	γ_i	$\gamma_{j,\mathbf{f}_j}$
5	$\mathbf{f}_j$	$\mathbf{f}_i$	$\gamma_{i,\mathbf{f}_j}$	γ_j
6	$\mathbf{f}_j$	$\mathbf{f}_i$	γ_i	$\gamma_{j,\mathbf{f}_i}$

Table 4.1: The six options are evaluated in the graph modification approach to find collision-free solution.

modification approach choose optimal option p^* such that the collision p_{col} can be resolved while the sum of distance is minimized.

Algorithm 4 resolves the collision p_{col} that happens between robot i and k. We first consider modifying the graph, corresponding to option 2 and 3. Let $\gamma'_{i,\mathbf{f}_i}$ be the modified graph which is removed an edge of the graph γ_i such that the edge is included in the path $\xi_{i,\mathbf{f}_i}(\gamma_i)$ and is before or on p_{col}. In other words, if collision point p_{col} belongs to a vertex and corresponds to the vertex $\xi^t_{i,\mathbf{f}_i} \in \xi_{i,\mathbf{f}_i}(\gamma_i)$, the edge (ξ^{t-1}_i, ξ^t_i) is removed from the graph γ_i. If p_{col} is on an edge (ξ^{t-1}_i, ξ^t_i), its edge is removed. Let $J(p)$ be the sum of distances of two robots for option p, and with modified graph of robot i and k, we have $J(3) = D_{i,\mathbf{f}_i}(\gamma'_{i,\mathbf{f}_i}) + D_{k,\mathbf{f}_k}(\gamma_k)$ and $J(4) = D_{i,\mathbf{f}_i}(\gamma_i) + D_{k,\mathbf{f}_k}(\gamma'_{k,\mathbf{f}_k})$. Next, we exchange the assigned goal between robot i and k. If the trajectories $\xi_{i,\mathbf{f}_k}(\gamma_i)$ and $\xi_{k,\mathbf{f}_i}(\gamma_k)$ does not collide at the collision point p_{col}, we do not need to

Algorithm 4: Collision Avoidance with Graph Modification between ξ_i and ξ_k

Input : $\xi_i, \xi_k, \gamma_i, \gamma_k, \mathbf{f}_i, \mathbf{f}_k, p_{col}$
Output : $p^*, \gamma_i, \gamma_k, \mathbf{f}_i, \mathbf{f}_k$

$J(p) \leftarrow \infty, \ \ \forall p = \{1, 2, \cdots, 6\}$
// For original assignment
$\gamma'_{i,\mathbf{f}_i} \leftarrow$ Remove an edge before or on p_{col} from γ_i
$\gamma'_{k,\mathbf{f}_k} \leftarrow$ Remove an edge before or on p_{col} from γ_k
$J(3) \leftarrow D_{i,\mathbf{f}_i}(\gamma'_{i,\mathbf{f}_i}) + D_{k,\mathbf{f}_k}(\gamma_k)$
$J(4) \leftarrow D_{i,\mathbf{f}_i}(\gamma_i) + D_{k,\mathbf{f}_k}(\gamma'_{k,\mathbf{f}_k})$
// For swapped assignment
$\xi'_i \leftarrow$ the shortest path from x_i to $\mathbf{f}_k$ on γ_i
$\xi'_k \leftarrow$ the shortest path from x_k to $\mathbf{f}_i$ on γ_k
if ξ'_i *and* ξ'_k *collide at* p_{col} **then**
    $\gamma'_{i,\mathbf{f}_k} \leftarrow$ Remove an edge before or on p_{col} from γ_i
    $\gamma'_{k,\mathbf{f}_i} \leftarrow$ Remove an edge before or on p_{col} from γ_k
    $J(5) \leftarrow D_{i,\mathbf{f}_k}(\gamma'_{i,\mathbf{f}_k}) + D_{k,\mathbf{f}_i}(\gamma_k)$
    $J(6) \leftarrow D_{i,\mathbf{f}_k}(\gamma_i) + D_{k,\mathbf{f}_i}(\gamma'_{k,\mathbf{f}_i})$
end
else
    $J(2) \leftarrow D_{i,\mathbf{f}_k}(\gamma_i) + D_{k,\mathbf{f}_i}(\gamma_k)$
end
// Choose the best option
$p^* \leftarrow \min(\text{argmin}_{p \in \{1,2,\cdots,6\}} J(p))$
switch p^* **do**
    case *1* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k$
    case *2* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_k, \mathbf{f}_i, \gamma_i, \gamma_k$
    case *3* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_i, \mathbf{f}_k, \gamma'_{i,\mathbf{f}_i}, \gamma_k$
    case *4* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma'_{k,\mathbf{f}_k}$
    case *5* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_k, \mathbf{f}_i, \gamma'_{i,\mathbf{f}_k}, \gamma_k$
    case *6* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_k, \mathbf{f}_i, \gamma_i, \gamma'_{k,\mathbf{f}_i}$
end

consider the graph modification, and get the sum of distance $J(2) = D_{i,\mathbf{f}_k}(\gamma_i) + D_{k,\mathbf{f}_i}(\gamma_k)$. If the collision happens, the graph modification is applied to avoid collision in the similar way of option 3 and 4, resulting in $J(5) = D_{i,\mathbf{f}_k}(\gamma'_{i,\mathbf{f}_k}) + D_{k,\mathbf{f}_i}(\gamma_k)$ and $J(6) = D_{i,\mathbf{f}_k}(\gamma_i) + D_{k,\mathbf{f}_i}(\gamma'_{k,\mathbf{f}_i})$.

After computing the sum of distance $J(p)$ for all options, we choose one option such that minimizing $J(p)$ and overwrite the assignment and graph for robot i and k. This process is

iteratively executed for each collision until all collisions are resolved, and during this process, the graph for robots is modified cumulatively, that is, the modified graph may be eventually removed multiple edges. For each pair of robots, Algorithm 4 focuses on resolving the collision p_{col}. If option 2 can avoid the collision p_{col} even though it has a collision at other location, the algorithm choose option 2 as the optimal option. Then, solving the new collision, the algorithm may choose option 2 again. Once this situation occurs, it cannot exit while loop in Algorithm 3. Therefore, we implement a method that monitors goal exchange to avoid such an endless loop.

In this paper, we will make the following assumption:

Assumption 1. *There exists a path from each robot x_i to assigned goal $\mathbf{f}_i$ on the modified graph γ_i*

Proposition 1. *The centralized local graph modification algorithm results in collision-free trajectories.*

Proof. The algorithm is to identify collision points p_{col} and then resolve p_{col} by local graph modification, which has two actions: exchanging goals and removing a single edge from a graph of either robot. First, we will show that local graph modification can solve the collision p_{col}, and then show that solving the collisions one by one results in collision-free trajectories.

The collision point p_{col} denotes a location where robot i and j collide, which is on a edge or a vertex. In the case that p_{col} is on the edge, exchanging the goals may not solve the conflict p_{col}. For example, in Figure 4.3, even if swapping the goals, both trajectories still include the collision point p_{col}, that is, the collision is not resolved by swapping the goals. Removing the edge of p_{col} from the graph of either robot, the robot which is removed the edge can no longer pass through the collision point. In addition, Assumption 1 guarantees that both robots have trajectories to assigned goals.

Next, we consider the case that the conflict p_{col} happens on a vertex. Similar to the precedent case, exchanging the goals does not always solve the collision. Therefore, we show the edge removal can solve the collision. The degree of vertex p_{col} denotes $d(p_{col})$, and let v_l,

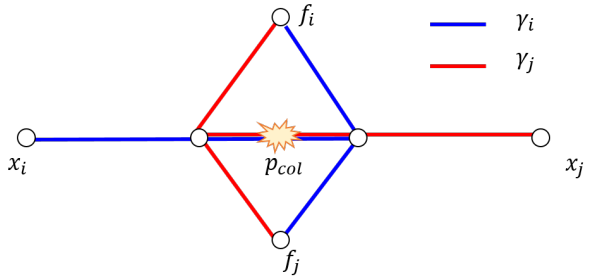


Figure 4.3: Avoiding collision on edge

$l \in [1,2,\cdots,d(p_{col})]$ be the neighboring vertex of p_{col}. We can assume without loss of generality that robot j passes through the vertex $v_{d(p_{col})}$ before the collision. Robot i passes through the vertex $v^* = v_l$ before the collision, $l \in [1,2,\cdots,d(p_{col})-1]$, because if robot i passes through the vertex $v_{d(p_{col})}$, robot i would collide with robot j at $v_{d(p_{col})}$, and the collision p_{col} would be no longer our concern. To avoid the collision, the algorithm removes the edge (v^*, p_{col}) from the modified graph γ_i and regenerates the trajectory of robot i. In some situation, robot i and j may still collide at the same vertex p_{col}. However, if all edges (v_l, p_{col}) for $l \in [1,2,\cdots,d(p_{col})-1]$ are removed, the collision p_{col} never happen anymore. Therefore, we can say the collision p_{col} can be solved by removing at most $d(p_{col})-1$ edges.

From the foregoing argument to avoid the collision p_{col}, every collisions in the system can be solved in such a way to apply local graph modification pairwisely. Although removing edges may disconnect a graph and robots may not be able to reach the goal, this algorithm guarantee to generate trajectories to assigned goal since Assumption 1, thus, it holds that this algorithm results in planning collision-free trajectories. ■

4.4 Centralized Algorithms

Now, we have two variations of the centralized algorithms to assign tasks and plan collision-free trajectories, summarized in Figure 4.4.

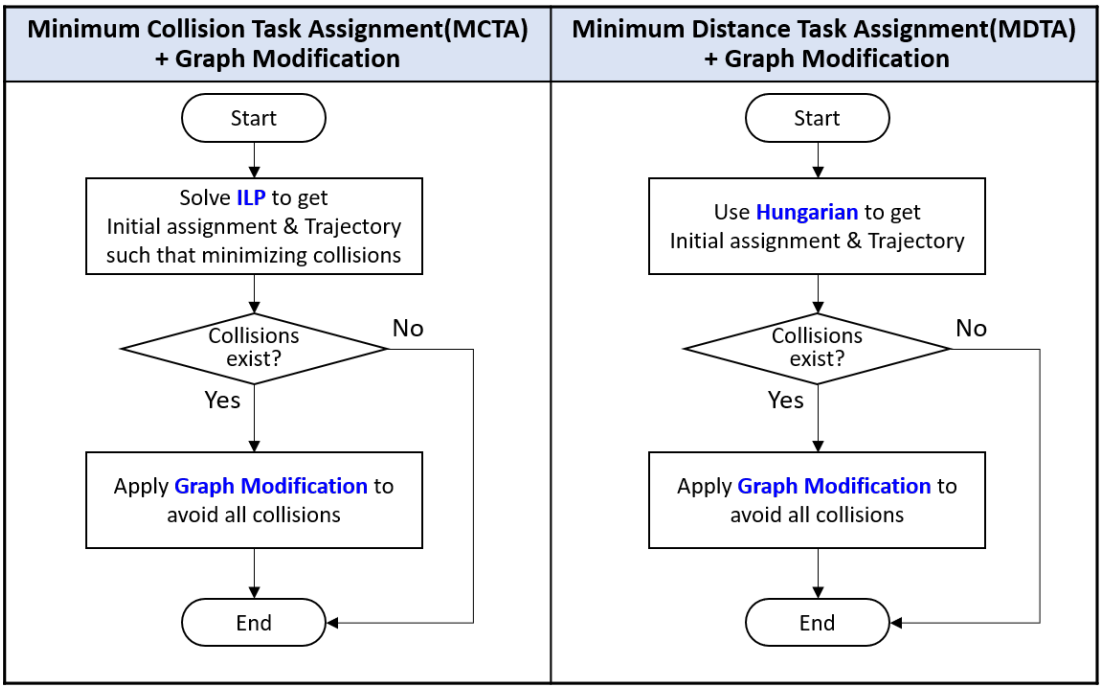


Figure 4.4: Centralized Algorithms

Minimum Collision Task Assignment(MCTA) + Graph Modification

The MCTA + graph modification first solves the ILP problem to assign goals which minimize the number of collisions, formulated in (4.25), and then avoiding the remaining collisions by the local graph modification approach in Algorithm 3.

Minimum Distance Task Assignment(MDTA) + Graph Modification

The MDTA + graph modification first assigns goals to minimize the total travel distance by the Hungarian algorithm in Appendix B.1, and then applies the local graph modification approach to avoid collisions.

Chapter 5

Decentralized Task Assignment and Trajectory Planning

5.1 Conceptual Approach

In this section, we discuss a decentralized approach to solve Problem 1 in a way to extend the knowledge related to the centralized approach. For the decentralized system, while robots cannot have state information of all other robots, they know their position and the location of all goals, and have a capability to exchange information, such as their current position, assigned goal and planned trajectory, with other robots closer than distance R_{com} on a roadmap.

Decentralized Problem Formulation

Robots constantly encounter new neighboring robots that come into the communication radius R_{com}, and learn new information from their neighborhoods. The key feature of this decentralized system is that each robot aims to minimize the total distance locally while avoiding collisions. Before we redefine Problem 1 as a decentralized problem, we first introduce some new notation.

Let $\mathcal{N}_i$ denote the set of neighborhoods with in the communication radius R_{com} of robot i:

$$\mathcal{N}_i = \{k \mid d_E(x_i, x_k) \leq R_{com}, k \neq i\} \subseteq \{1, 2, \cdots, N\} \backslash i$$

where, $d_E(x_i, x_k)$ is the euclidean distance between robots i and k. Let $\Xi_i = \{\xi_k \mid k \in \mathcal{N}_i \bigcup i\}$ denote the set of trajectories for robot i and the neighborhood. Problem 1 is reduced to distributed problem for each robot as follows:

$$\Xi_i^* = \underset{\Xi_i}{\operatorname{argmin}} \sum_{k \in \mathcal{N}_i \bigcup i} l(\xi_k) \tag{5.1}$$

subject to:

ξ_i and ξ_k are collision-free $\forall k \in \mathcal{N}_i$

We exploit the knowledge of the centralized algorithm to construct the collision-free trajectory within the neighborhood and reformulate (5.1). The assignment matrix ϕ is no longer known by any one robot, but robots can know assignment for their neighboring robots, $\Phi_i = \{\phi_k | k \in \mathcal{N}_i \bigcup i\}$ be the set of assigned vectors for robot i and its neighborhood, and similarly $\Gamma_i = \{\gamma_k \mid k \in \mathcal{N}_i \bigcup i\}$ be the set of modified graph.The set of targets assigned to the neighborhoods of robot i is given by:

$$\mathcal{T}_i = \{\mathbf{f}_k \mid k \in \mathcal{N}_i\} \subseteq \{1, 2, \cdots, N\} \backslash \mathbf{f}_i$$

We formulate the decentralized problem with the graph modification:

Problem 3. *For each robot i, the objective is to find the set of modified graphs Γ_i^* and task assignment Φ_i^* which result in the collision-free while minimizing the total travel distance:*

$$\Gamma_i^*, \Phi_i^* = \underset{\Gamma_i, \Phi_i}{\operatorname{argmin}} \sum_{k \in \mathcal{N}_i \bigcup i} \sum_{j \in \mathcal{T}_i \bigcup \mathbf{f}_i} \phi_{k,j} D_{k,j}(\gamma_k) \tag{5.2}$$

subject to:

$\xi_i(\gamma_i)$ *and* $\xi_k(\gamma_k)$ *are collision-free.* $\forall k \in \mathcal{N}_I$

The number of possible modified graphs is at most $2^{|E|}$ per robot as in the centralized problem. Since each robot has $|\mathcal{N}_i|$ neighborhoods, the robot must calculate and compare the total cost $2^{(|\mathcal{N}_i|+1)|E|}$ times in a straightforward approach. This approach is not acceptable due to computationally demanding.

Conceptual Approach for Decentralized System

The proposed decentralized algorithms combine two algorithms for the planning phase and the execution phase. The planning phase is to assign task and plan trajectories before robots start moving, and we consider three methods in this phase;Minimum Distance Task Assignment(MDTA), Minimum Collision Task Assignment(MCTA), and Random Task Assignment. Then, in the execution phase, which is after robots started moving, each robot replans the assignment and trajectory with the local graph modification to avoid collisions at every discrete time steps. In addition, since we do not assume a centralized planner, each robot are required to coordinate to resolve conflict of the decision among the neighborhood.

5.2 Initial Task Assignment

In this section, we propose two decentralized algorithms to decide an initial task assignment; MCTA and MDTA. Although both algorithms are developed based on corresponding centralized algorithms, thre are some challenges for adopting to the decentralized system. While the centralized task assignment algorithms assume a centralized planner, the decentralized algorithms need to coordinate conflicts of decisions to converge to consensus. Furthermore, since objective functions of the centralized algorithms (4.25) and (B.1) depend on neighborhood decision as well, they are not cut out for distributed coordination. We modify the objective functions to fit the decentralized system, and make use of the consensus-based auction algorithm(CBAA)

in Appendix B.3 to obtain a sub-optimal task assignment to minimize the objective function. To converge to a feasible assignment, which each robot is assigned to each goal without duplication, CBAA requires that all of the robots join the same connected communication network.

Minimum Collision Task Assignment

The centralized MCTA assigns tasks to all robots in the system by solving the ILP problem to minimize the number of collisions. The straightforward way for adopting to the decentralized algorithm will be that each robot solves the ILP problem within the neighborhood and decide assignment for the neighborhood as well as its self. In this approach, after robots broadcast the decision to their neighborhood, they need to negotiate and resolve the conflicts in terms of their own assignment and their neighborhood assignments. However, it is hard to converge to a valid assignment. Therefore, the proposed decentralized MCTA approach is that each robot focuses on deciding only its own assignment, such as minimizing the expectations of the collisions within the neighborhood and resolve the conflict if it receives information about the other robots are assigned to the same task. The conflict can be resolved by CBAA.

Based on the centralized MCTA defined in Section 3, we propose an objective function combining the expectation of the collisions and the distance to the assigned goal. First, we define the expected value of the number of collisions that happen on a path from robot i to goal j. Let $p_{k,h}$ be the probability that neighboring robot k is assigned to goal h, and we assume that it follows a discrete uniform distribution, that is, $p_{k,h} = 1/N-1$. The denominator is $N-1$ because robot i is assigned to one of N goals and robot k is assigned to remaining goal. Since robot i knows locations of all goals, computing the adjacency matrix $a_{(i,j),(k,h)}$ requires the location of robot k, that is, robot i can compute only expected value related to the neighborhoods. With the assignment probability $p_{k,h}$, we can reformulate the number of collisions which happens on the

trajectory from robot i to goal j (4.19) as the expected value, given by:

$$b_{i,j} = \frac{1}{\mathcal{N}_i} \sum_{k \in \mathcal{N}_i} \sum_{h=1}^{N} a_{(i,j),(k,h)} p_{k,h} \tag{5.3}$$

The expected value is normalized by the number of neighborhood to eliminate the influence of the number of neighborhood. Given an robot i's assignment $\phi_i \in \{0,1\}^N$, the expected value of the collisions in which robot i gets involved is shown as:

$$h_{Ai}(\phi_i) = \sum_{j=1}^{N} b_{i,j} \phi_{i,j} \tag{5.4}$$

In the decentralized MCTA, (4.4) is also reformulated such that it depends on only robot i's assignment, given by:

$$f_i(\phi_i) = \sum_{j=1}^{N} D_{i,j} \phi_{i,j} \tag{5.5}$$

This function represents the distance from robot i to assigned goal $\mathbf{f}_i$. Since (5.4) and (5.5), an objective function of the decentralized MCTA is defined as:

$$F_i(\phi_i) = w_1 f_i(\phi_i) + w_2 h_{Ai}(\phi_i) \tag{5.6}$$

To prioritize minimizing the expectation of collisions $h_{Ai}(\phi_i)$ over minimizing the distance $f_i(\phi_i)$, we set the weight value as $w_1 << w_2$. With the objective function F_i, each robot decides the initial task to locally minimize $F_i(\phi_i)$ using the CBAA.

Minimum Distance Task Assignment

The decentralized MDTA also exploits the CBAA, and its objective function is defined as the distance from robot to goal given by:

$$F_i(\phi_i) = \sum_{j=1}^{N} D_{i,j}\phi_{i,j} \tag{5.7}$$

5.3 Decentralized Local Graph Modification

After accomplishing the initial task assignment process proposed in the previous section, robots start moving to their assigned goal along with the planned path. Since the initial task assignment does not guarantee collision-free, each robot is required to implement collision avoidance locally. In this section, we propose a decentralized collision avoidance algorithm based on the graph modification discussed in Section 4.3.

Algorithm 5 and 6 show the procedure of robot i at discrete time step t. Let $\mathcal{U}_i \subseteq \mathcal{N}_i$ be the update list, which is the list of robots which robot i attempts to check collisions and applies the graph modification. The update list $\mathcal{U}_i$ is updated promptly so that robot i can check the collision with a robot that has potential collisions. At every discrete time step t, $\mathcal{U}_i$ is initialized as the list of robots that robot i newly encounters since the previous time step $t-1$. If $t = 0$, $\mathcal{U}_i$ is the same as the list of the neighborhood. Furthermore, if the assigned goal or trajectory is changed through the graph modification, the robot needs to check collisions with all of the neighborhood again(line 7 of Algorithm 5, and line 4 of Algorithm 6).

Robot i attempts Algorithm 7 to each robot according to the update list $\mathcal{U}_i$. The decentralized collision avoidance algorithm is similar to the centralized algorithm, Algorithm 2.However, while the centralized algorithm is applied to each collision, the decentralized algorithm is basically applied to all neighborhood regardless of whether the collision exists. In the decentralized graph modification algorithm, robot i first check collision between the pair of robots. Even if it turns

Algorithm 5: Decentralized Local Graph Modification Algorithm for robot i at discrete time t

Input : $\mathcal{N}_i(t), \mathcal{N}_i(t-1), x_i, \mathbf{f}_i, \gamma_i$
Output : $x_i, \mathbf{f}_i, \gamma_i$

$\mathcal{U}_i \leftarrow \mathcal{N}_i(t) \backslash \mathcal{N}_i(t-1)$
while $\mathcal{U}_i \neq \phi$ **do**
  $k \leftarrow$retrieve from $\mathcal{U}_i$
  request robot k to send $x_k, \mathbf{f}_k, \gamma_k$
  $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k, p^* \leftarrow$ run Algorithm 7$(\xi_i(\gamma_i), \xi_k(\gamma_k))$
  if *task or path are changed* ($p^* \neq 1$) **then**
    $\mathcal{U}_i = \mathcal{N}_i(t) \backslash k$
  end
  send γ_k and $\mathbf{f}_k$ to robot k
end

Algorithm 6: Receive data from robot k

Input : $\mathcal{U}_i, \mathcal{N}_i(t), \overline{\gamma_i}, \overline{\mathbf{f}_i}$
Output : $\mathcal{U}_i, \gamma_i, \mathbf{f}_i$

if *receive* $\overline{\gamma_i}$ *and* $\overline{\mathbf{f}_i}$ *from robot* k **then**
  if *task or path are changed*($\gamma_i \neq \overline{\gamma_i}$ *or* $\mathbf{f}_i \neq \overline{\mathbf{f}_i}$) **then**
    $\gamma_i, \mathbf{f}_i \leftarrow \overline{\gamma_i}, \overline{\mathbf{f}_i}$
    $\mathcal{U}_i \leftarrow \mathcal{N}_i(t) \backslash k$
  end
  else
    $\mathcal{U}_i \leftarrow \mathcal{U}_i \backslash k$
  end
end

out that the robots do not collide, robot i exchange the assigned goals to find out an option to minimize the sum of distance. Therefore Algorithm 7 contributes to the reduction of the total travel distance by exchanging the goal as well as the collision avoidance.

In general, decentralized algorithms are required asynchronous implementation. In the proposed decentralized algorithm, there is possibility that a robot receive a task assignment and modified graph, which contradict the recent information, from the neighborhood due to the communication and process delay. As one of the solution to this issue, a handshaking procedure is proposed in [24], which guarantees recent information for the robots and secure the coordination

Algorithm 7: Local Graph Modification with ξ_k for robot i

Input : $\xi_i, \xi_k, \gamma_i, \gamma_k, \mathbf{f}_i, \mathbf{f}_k, p_{col}$
Output : $p^*, \gamma_i, \gamma_k, \mathbf{f}_i, \mathbf{f}_k$

$J(p) \leftarrow \infty, \quad \forall p = \{1, 2, \cdots, 6\}$
while *True* **do**
  if ξ_i *and* ξ_k *does not collide* **then**
    $J(1) \leftarrow D_{i,\mathbf{f}_i}(\gamma_i) + D_{k,\mathbf{f}_k}(\gamma_k)$
    if $\xi_{i,\mathbf{f}_i}(\gamma_i)$ *and* $\xi_{k,\mathbf{f}_k}(\gamma_k)$ *does not collide* **then**
      $J(2) \leftarrow D_{i,\mathbf{f}_k}(\gamma_i) + D_{k,\mathbf{f}_i}(\gamma_k)$
    end
  end
  else
    $p_{col} \leftarrow$ collision point between robot i and k
    $\gamma'_{i,\mathbf{f}_i} \leftarrow$ remove an edge in $\xi_{i,\mathbf{f}_i}(\gamma_i)$ before or on p_{col} from γ_i
    $\gamma'_{k,\mathbf{f}_k} \leftarrow$ remove an edge in $\xi_{k,\mathbf{f}_k}(\gamma_k)$ before or on p_{col} from γ_k
    $J(3) \leftarrow D_{i,\mathbf{f}_i}(\gamma'_{i,\mathbf{f}_i}) + D_{k,\mathbf{f}_k}(\gamma_k)$
    $J(4) \leftarrow D_{i,\mathbf{f}_i}(\gamma_i) + D_{k,\mathbf{f}_k}(\gamma'_{k,\mathbf{f}_k})$
    if $\xi_{i,\mathbf{f}_k}(\gamma_i)$ *and* $\xi_{k,\mathbf{f}_i}(\gamma_k)$ *collide at* p_{col} **then**
      $\gamma'_{i,\mathbf{f}_k} \leftarrow$ Remove an edge in $\xi_{i,\mathbf{f}_k}(\gamma_i)$ before or on p_{col} from γ_i
      $\gamma'_{k,\mathbf{f}_i} \leftarrow$ Remove an edge in $\xi_{k,\mathbf{f}_i}(\gamma_k)$ before or on p_{col} from γ_k
      $J(5) \leftarrow D_{i,\mathbf{f}_k}(\gamma'_{i,\mathbf{f}_k}) + D_{k,\mathbf{f}_i}(\gamma_k)$
      $J(6) \leftarrow D_{i,\mathbf{f}_k}(\gamma_i) + D_{k,\mathbf{f}_i}(\gamma'_{k,\mathbf{f}_i})$
    end
    else
      $J(2) \leftarrow D_{i,\mathbf{f}_k}(\gamma_i) + D_{k,\mathbf{f}_i}(\gamma_k)$
    end
  end
  $p^* \leftarrow \min(\text{argmin}_{p \in \{1,2,\cdots,6\}} J(p))$
  switch p^* **do**
    case *1* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k$
    case *2* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_k, \mathbf{f}_i, \gamma_i, \gamma_k$
    case *3* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_i, \mathbf{f}_k, \gamma'_{i,\mathbf{f}_i}, \gamma_k$
    case *4* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma'_{k,\mathbf{f}_k}$
    case *5* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_k, \mathbf{f}_i, \gamma'_{i,\mathbf{f}_k}, \gamma_k$
    case *6* **do** $\mathbf{f}_i, \mathbf{f}_k, \gamma_i, \gamma_k \leftarrow \mathbf{f}_k, \mathbf{f}_i, \gamma_i, \gamma'_{k,\mathbf{f}_i}$
  end
  $\xi_i, \xi_k \leftarrow \xi_{i,\mathbf{f}_i}(\gamma_i), \xi_{k,\mathbf{f}_k}(\gamma_k)$
  if ξ_i *and* ξ_k *does not collide* **then**
    break
  end
end

process among the pair of robots. When we implement the algorithm into a real-world application, such kind of procedures is necessary.

The decentralized system also requires Assumption 1 in Section 4.3 to have the following proposition.

Proposition 2. *For each time step, the decentralized local graph modification algorithm determines an assignment and trajectory that the robot does not collide with its neighborhood.*

Proof. Since the graph modification process is the same as the centralized one, Proposition 1 implies that the decentralized algorithm guarantees to result in collision-free trajectories within the neighborhood. ■

We make the following assumption:

Assumption 2. *The communication radius is more than 2.*

Proposition 3. *The decentralized local graph modification algorithm enable all robots reach to goals without collisions.*

Proof. Since Proposition 2 and Assumption 2, at every time step, each robot is guaranteed not to collide with other robot within at least distance 2. Since robots move by the distance 1 at each time step, it is certain that any collision does not occur until the next time step. Therefore, as time goes on, each robot can approach the goal without collision, and then eventually all the robots reach the goal. ■

5.4 Decentralized Algorithms

Based on the discussion so far, we propose three decentralized algorithms as shown in Figure 5.1. Each algorithm assigns goals initially in the planning phase in a different way, and then all of three use the same collisions avoidance algorithm after robots start moving.

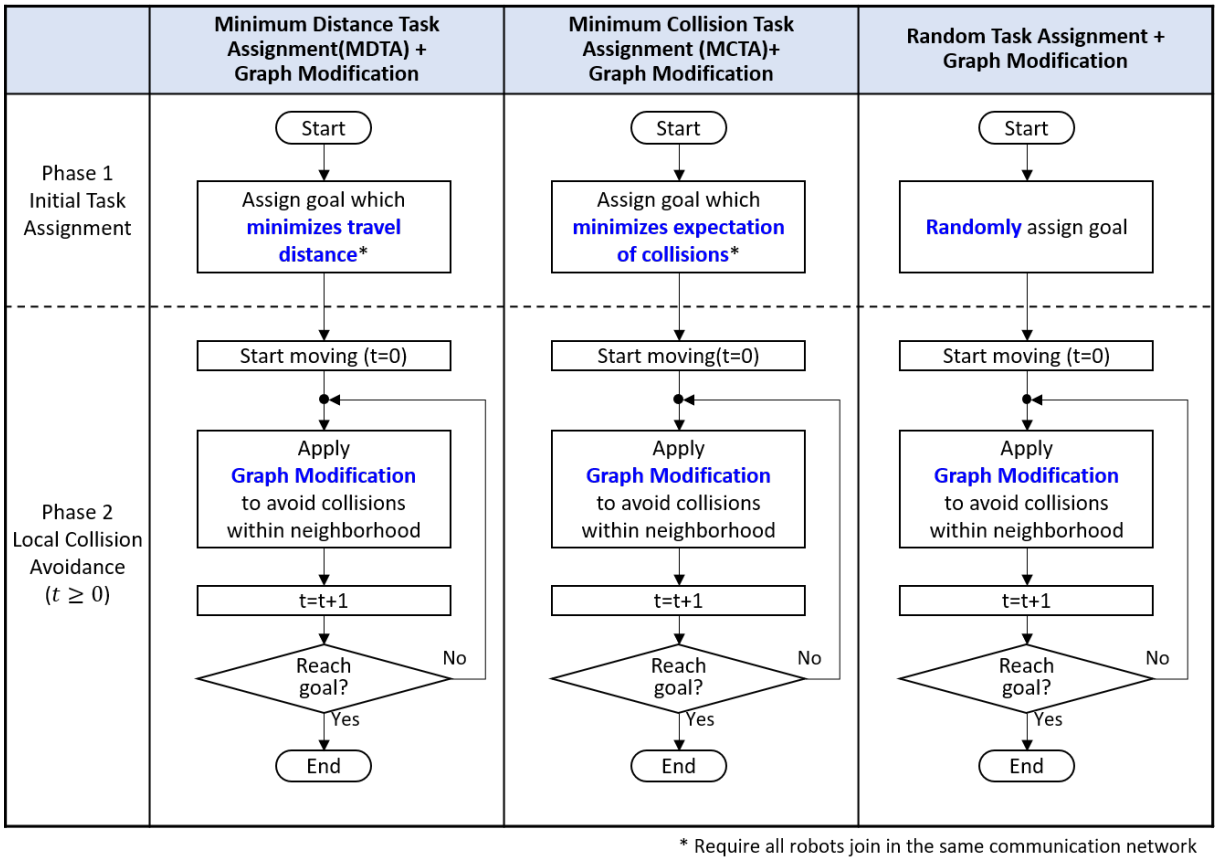


Figure 5.1: Decentralized Algorithms

Minimum Distance Task Assignment(MDTA) + Graph Modification

Before robot start moving, each robot coordinates the initial task assignment to minimize the distance (5.7). To reach the task assignment that robots are assigned to goals without duplication, it is required that all robots join one connected communication network. Then, robots move ahead to assigned goal while performing the local graph modification in Algorithm 5. In this phase, the system does not require that the communication network is connected.

Minimum Collision Task Assignment(MCTA) + Graph Modification

Each robot coordinates the initial task assignment to minimize the expectation of collisions with the neighborhood, formulated in (5.6). Once the initial task assignment is determined, robots

move to assigned goals in the same way as MDTA + Graph Modification.

Random Task Assignment + Graph Modification

Robots are randomly assigned to tasks in the planning phase. For example, each robot is assigned to the goal corresponding to its index. Then, robots start moving in the same as other algorithms. Note that this algorithms does not require the network connectivity at all.

Chapter 6

Simulation Results and Discussion

6.1 Centralized Algorithm

We evaluate the performance of two proposed algorithms in Section 4.4; Minimum Distance Task Assignment (MDTA) + Graph Modification and Minimum Collision Task Assignment (MCTA) + Graph Modification. In addition, we simulate only MDTA based on Hungarian algorithm, which provides the lower bound of the total travel distance. The MDTA results may include collisions.

Performance Evaluation per Number of Robots

The first set of simulation evaluate the performance as varying the number of robots on the two different roadmaps, shown in Figure 6.1 and 6.2. For each condition of simulation, 100 trials are performed, and for each trial, the robots and the goals are randomly deployed. The three algorithms are compared on the same set of initial locations for each condition of simulation. We assume two different types of roadmaps on which robots move: a grid graph shown in Figure 6.1(a) and a hexagonal graph in (a)-2, which have 169 and 160 nodes, respectively. The number of nodes is set to be almost the same to eliminate the relative influence. The length of each edge

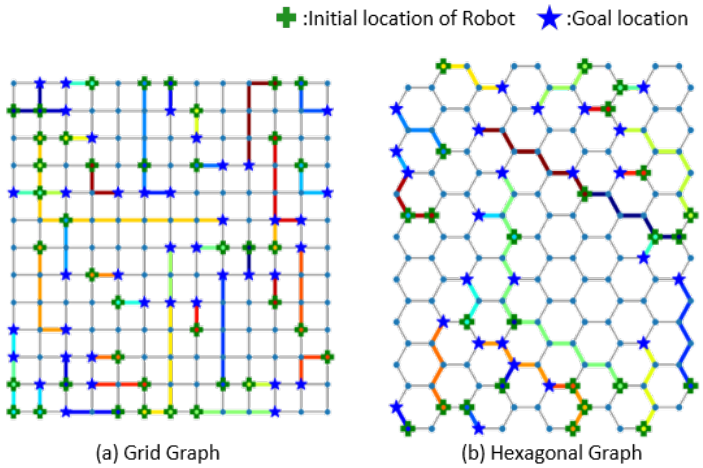


Figure 6.1: Roadmaps and trajectories with $N = M = 40$. Green cross represents initial location of robots, blue star represents goal location, and colored lines represent the planned trajectory.

is 1. Since the degree of vertices of the hexagonal graph is smaller than one of the grid graph, we expected more collisions in the hexagonal grid. But as seen from the results, there were no significant differences between both roadmaps.

Figure 6.2 (a) shows the number of collisions that MDTA has. Collisions are more likely to occur as the number of robots increases. Also, from this result, we can know the number of conflicts that MDTA + Graph Mod. and MCTA + Graph Mod. have to deal with. The box plot of the execution time is evaluated in (b). The execution time increases exponentially as the number of robots increases.

For all trials in this simulation set, MCTA resulted in collision-free task assignment, and local graph modification was not performed at all. Therefore, the execution time of MCTA + Graph Mod indicates the time taken for only MCTA process. MDTA + Graph Mod can lead results faster than MCTA + Graph Mod. Figure 6.2 (c) shows the total travel distance, which is the sum of distance which all robots travel. Despite avoiding collisions, both proposed algorithms show almost the same performance as MDTA. However, this simulation results probably depend on the underlying graph. More simulation and theoretical analysis will be needed to understand

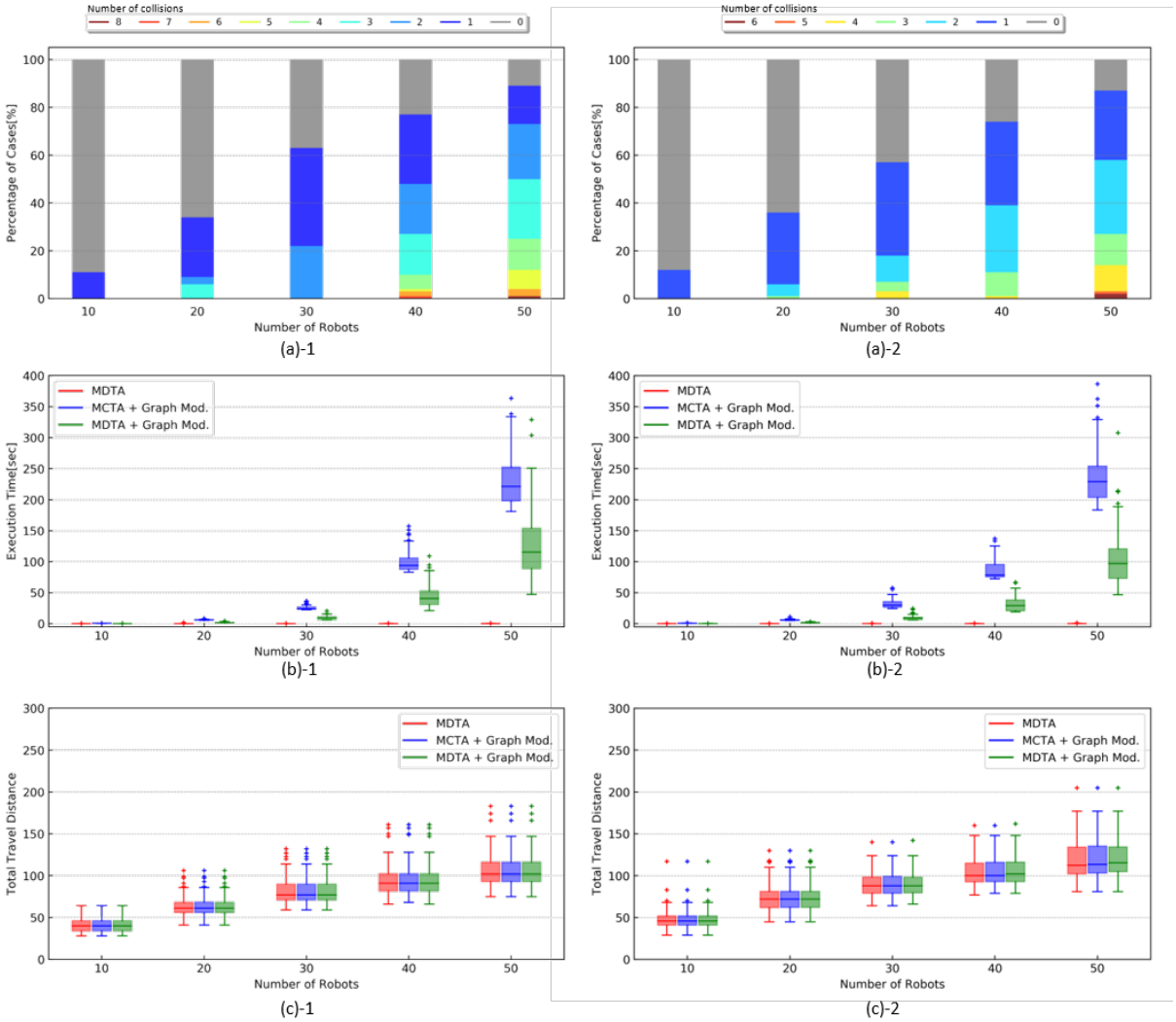


Figure 6.2: Simulation results of centralized system with varying the number of robots. (a) shows the population of trials per collisions for MDTA. (b) and (c) show the box plots of execution time and total travel distance per number of robots. The "+" marks represents statistical outliers.

the graph effects on the total travel distance.

Performance Evaluation per Number of Collisions to be Solved

We evaluate the performance per the number of collisions to be addressed, as shown in Figure 6.3. In this simulation, each robot moves along with a grid graph whose length and width are 10 each, and this graph has 100 vertices. The number of robots is 30. Figure 6.3 (a)

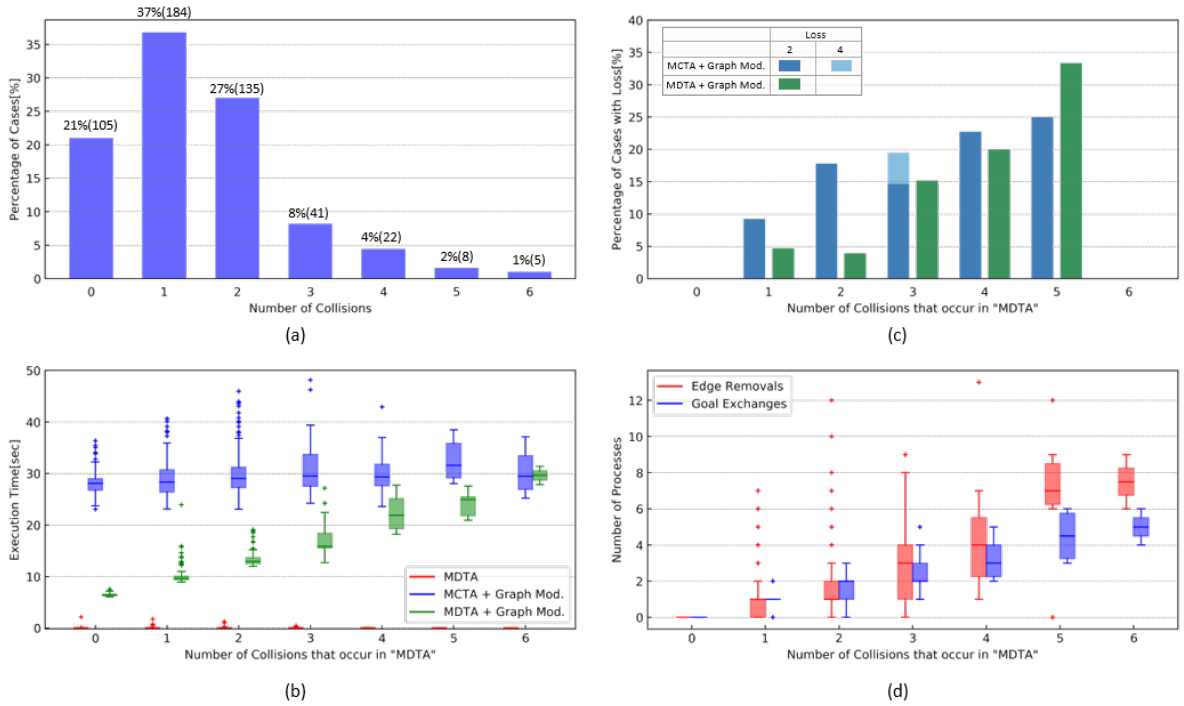


Figure 6.3: Simulation results of centralized system with $N = 30$. (a) shows the histogram of the number of collisions in MDTA. (b) shows the execution time per the number of collisions. The loss of total distance is analyzed in (b) and the number of Graph Mod. processed is shown in (d).

shows the distribution of collisions for 500 trials of MDTA, which is based on the Hungarian algorithm. From this result, we can see that 80% of the trials include collisions and, although it's a tiny percentage, the maximum number of collisions is 6. We compare the performance of each algorithm with the same initial deployments shown in Figure6.3(a). (b) depicts the execution time in terms of the number of collisions that occur in the MDTA. MDTA + Graph Mod avoids the collisions that occur in MDTA with modifying the graph, and therefore, as the number of collisions in MDTA increases, the execution time also increases. On the other hand, in MCTA + Graph Mod, although the objective of MCTA is to assign tasks in a way to minimize the number of collisions, the task assignment does not have any collisions in all trials of this simulation. Therefore, the execution time of MCTA + Graph Mod is almost constant regardless of the number of collisions in MDTA. Even when the number of collisions in MDTA is 0, the execution time of

MDTA + Graph Mod is not the same as that of MDTA. It is because MDTA + Graph Mod, which guarantees the collision-free, needs to check the collisions after the initial task assignment by the MDTA process. Next, we evaluate a loss, which is defined as the difference from MDTA in terms of the total travel distance. As shown in Figure 6.3 (c), the more collisions the system has to avoid, the more loss it suffers. However, we can see that both algorithms give the collision-free solution without the loss in most cases. In addition, this simulation reveals that loss is two in almost all cases. For six collisions in MDTA, there are no cases with the loss. It seems like not proper results because the number of trials is five cases shown in Figure 6.3 (a). Finally, Figure 6.3 (d) shows the number of edge removals and goal exchange carried out during the Graph Mod process. MCTA + Graph Mod can have a collision-free solution in the MCTA process and does not need Graph Mod process. Therefore, (d) shows the results of MDTA + Graph Mod. From this result, as the collisions in MDTA increase, more edge removals and goal exchanges are required to solve the collisions.

6.2 Decentralized Algorithm

We evaluated the performance of the three decentralized algorithms proposed in Section 5.4: MDTA + Graph Mod., MCTA + Graph Mod., and Random TA + Graph Mod. Each simulation condition is performed 100 trials, and for each trial, the goals were randomly deployed. The decentralized MDTA and MCTA require that all robots join a connected network. To obtain the initial deployment that meets the requirement, we check the connectivity of communication for robots randomly deployed. If it does not show the connectivity, the robots are randomly deployed on the roadmap repeatedly. The three algorithms are compared on the same set of initial locations for each simulation condition.

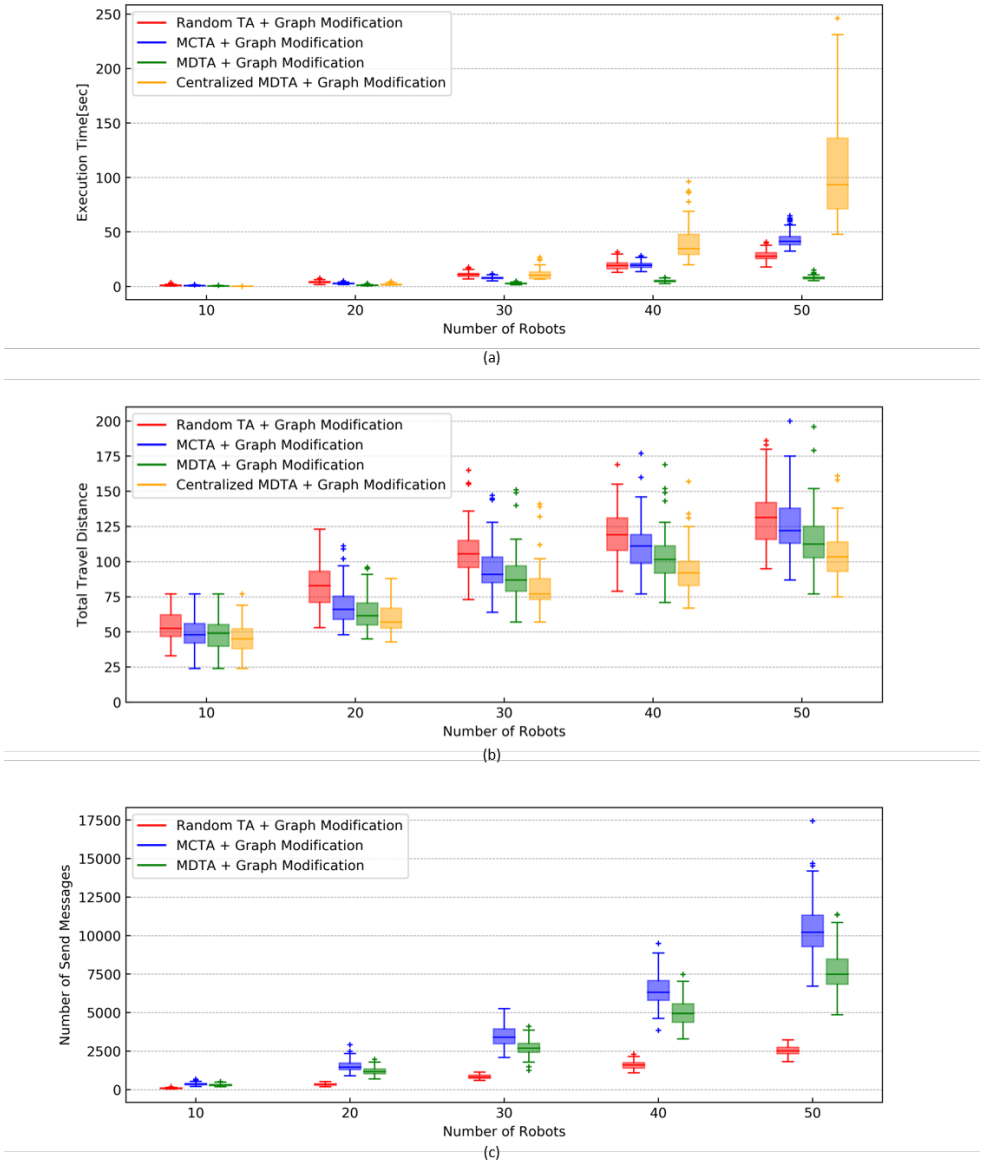


Figure 6.4: Simulation results of decentralized system with varying the number of robots for communication radius $R_{com} = 4$ on 13x13 grid graph. (a) execution time, (b) total travel distance, and (c) the number of send messages. The "+" marks in each box plot represents statistical outliers.

Performance Evaluation per Number of Robots

We evaluate performance on a 13x13 grid graph for $R_{com} = 4$. Figure 6.4(a) shows the execution time defined as the sum of cumulative time it takes to process the algorithm at each

time step for all robots. The execution time of MDTA + Graph Mod is less than other algorithms. In the case of more than 30 robots, decentralized algorithms are superior to the centralization algorithm. While the execution time of MDTA + Graph Mod. and Random TA + Graph Mod. show almost linear with the number of robots, MCTA + Graph Mod. polynomially increases.

The total number of communications is shown in Figure 6.4(b). Since random TA + Graph Mod. does not need to coordinate the initial task assignment, it requires less communication. MCTA + Graph Mod. requires sending more messages to generate a local conflict graph in the initial task assignment phase. As for the total travel distance, MDTA + Graph Mod. is the closest to the centralized results, as shown in Figure 6.4(c).

The performance of the initial task assignment is analyzed as shown in Figure 6.5. (a) shows the total travel distance planned in the initial task assignment. The decentralized MDTA results in the smallest total distance and works well as desired. Furthermore, as shown in Figure 6.5(b), the number of goal exchanges that mainly contribute to the reduction in total distance is the lowest in MDTA + Graph Mod.

Next, we show the number of potential conflicts in the initial task assignment in Figure 6.5(c). MCTA is superior to MDTA in terms of minimizing collisions. Since the less number of collisions, the graph modification performed to avoid collisions is the lowest in MCTA + Graph Mod. with respect for the median, as shown in (d).

Performance Evaluation per Communication Radius

Figure 6.6 and 6.7 show the simulation results for $N = 30$ as varying communication ranges from the minimum value of 2 to enough big value, $R_{com} = 16$. The robot is deployed on a 10x10 grid graph roadmap to make it easier to generate initial locations randomly that all robots are connected. In order to minimize the influence of initial location of robots and goals, every simulation cases use the same set of initial location for running 100 trials. Since the communication radius is equivalent to the size of the neighborhood, both execution time (a)

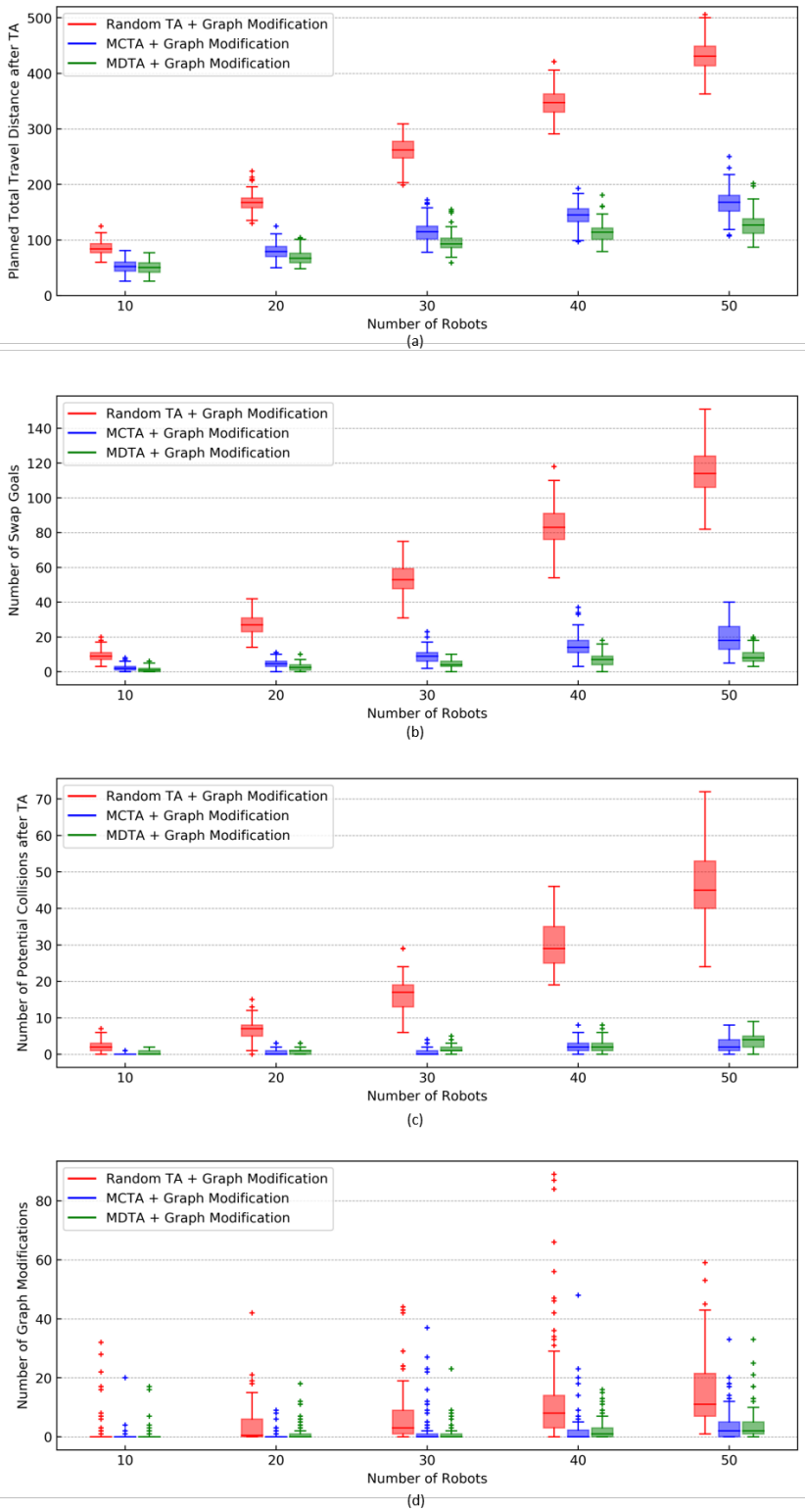


Figure 6.5: Performance evaluation for initial task assignment algorithms for $R_{com} = 4$ on 13x13 grid graph.

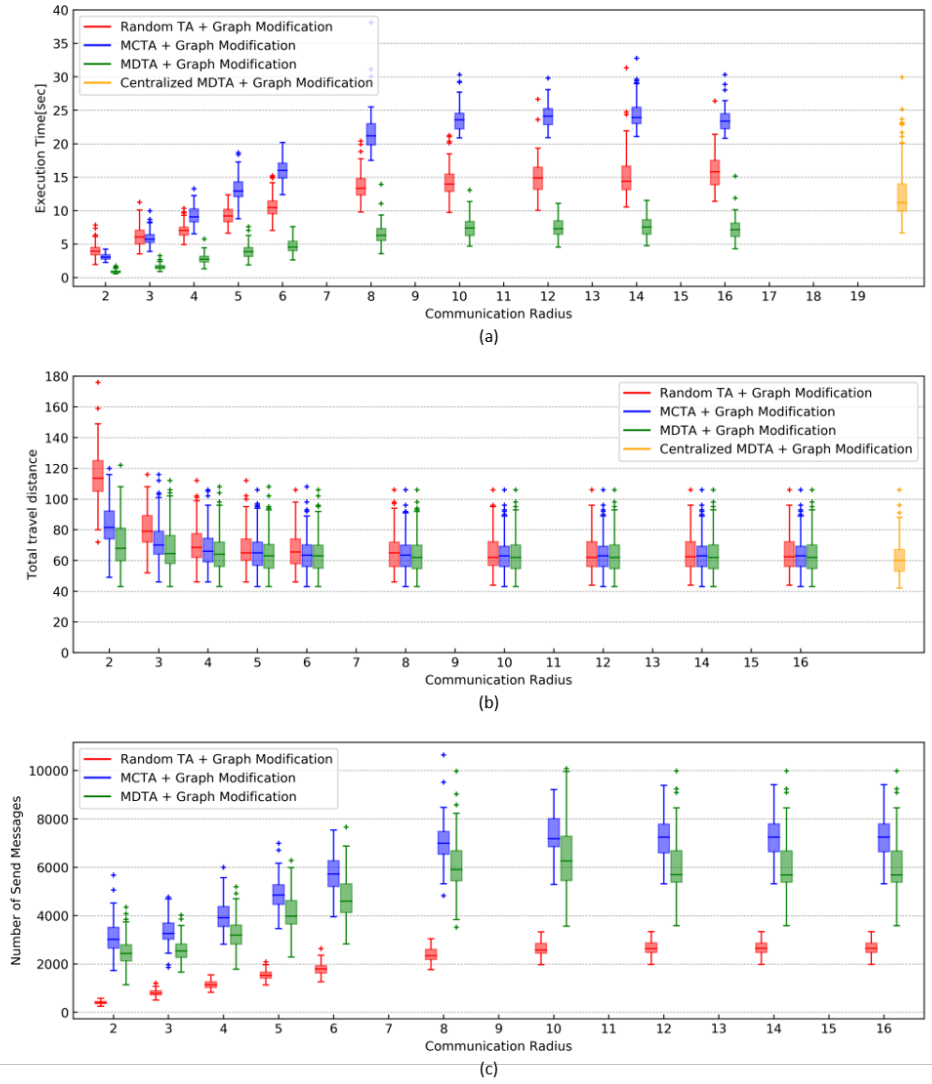
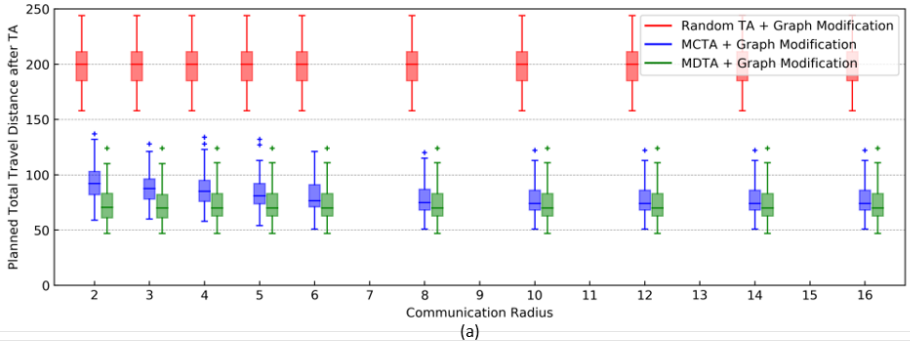


Figure 6.6: Simulation results of decentralized system with varying communication radius for $N = M = 30$ on 10x10 grid graph. (a) execution time, (b) total travel distance, and (c) the number of send messages. The "+" marks in each box plot represents statistical outliers.

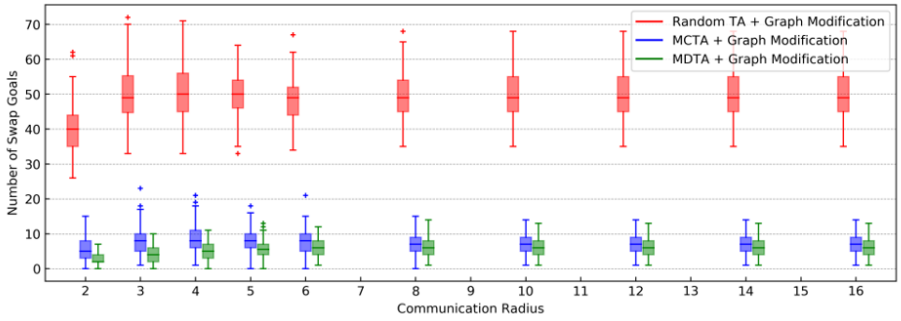
and the number of communications (b) increase and reach a plateau as the communication radius increases. In terms of the execution time, MDTA+Graph Mod. is better than the centralized algorithm even for enough large communication radius. The total travel distances are approaching to the centralized result, and a gap between each algorithm is narrowed as the communication radius increases. MDTA+Graph Mod. can obtain a solution closer to optimal even if the

communication radius is limited.

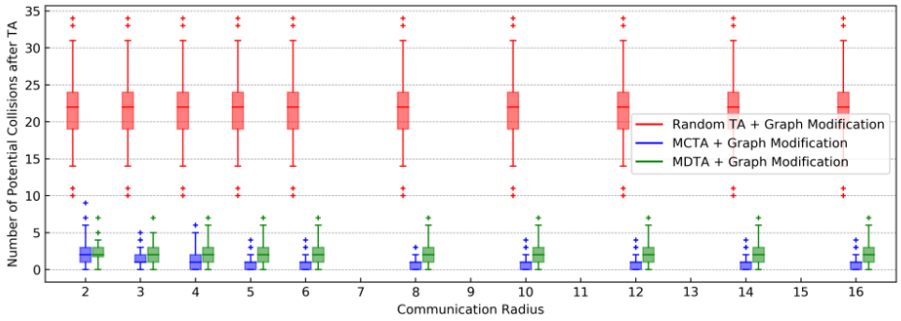
Next, we discuss the performance of initial task assignments shown in Figure 6.7. As for the total distance and potential collisions after the initial task assignment, MDTA + Graph Mod. is not affected by the communication radius. This is because the objective function of MDTA is invariable to the size of the neighborhood. On the other hand, the results of MCTA + Graph Mod. vary over the communication radius since MCTA computes the local conflict graph for the neighborhood. Note that Random TA is unrelated to the communication radius.



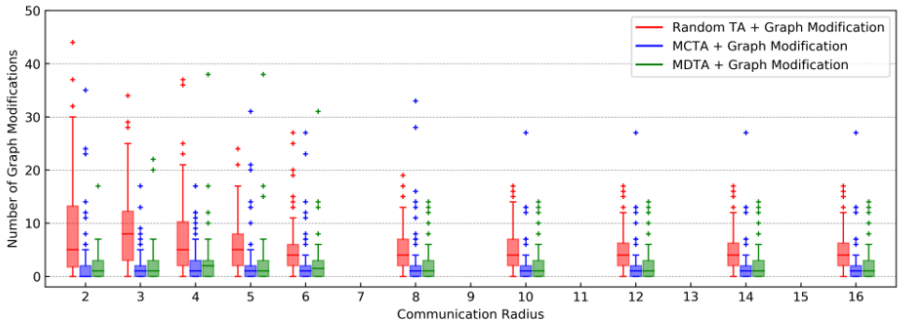
(a)



(b)



(c)



(d)

Figure 6.7: Performance evaluation for initial task assignment algorithms for $N = M = 30$ on 10x10 grid graph.

Chapter 7

Conclusion

In this paper, we considered the problem of collision-free task assignment and trajectory planning for multi-robot systems moving along a predefined roadmap.

We first developed two centralized algorithms, which assign tasks to minimize the total distance or the collisions and then resolve the collisions with replanning the trajectory by graph modification. The conflict graph, which encodes all possible collisions among the system, was utilized to formulate an optimization problem, whose solution is a task assignment to minimize the collisions while minimizing the total distance. The local graph modification approach, which reconstructs the trajectory to avoid collisions with removing an edge on the trajectory from the roadmap graph, guarantees collision-free. We show proposed algorithms can avoid all collisions at the cost of almost no increase of total distance.

We proposed three decentralized algorithms extending the centralized methods, which assign the initial task before robots start moving in such a way to minimize the total travel distance, minimize the collisions, or deploy randomly, and then avoid the collision with graph modification. To realize the minimum collision assignment, we introduced the expectation of the collisions based on the local conflict graph, which encodes possible collisions among the neighborhood and is utilized as a bid in a market-based coordination approach. The performance of the

proposed algorithms was evaluated by computer simulation. The proposed algorithms provided the suboptimal collision-free solution, and the algorithm of minimum distance assignment and graph modification showed the best performance among the proposed algorithms.

Future Works

This section briefly summarizes future works that could be considered. In terms of performance improvement, the decentralized minimum collision task assignment has the potential to improve in a way that each robot keeps updating a ”conditional” expectation as the bid based on the decision of the neighborhood's assignment. Although there is some challenge to coordinate the bids, which dynamically changes, this approach is expected to reduce collisions in the initial task assignment. Additionally, we assume the existence of the path to goal on the modified graph to guarantee to reach the collision-free solution, so that it is interesting to make it more reasonable. Finally, future work includes considering uncertainty, such as the existence of moving obstacles and dynamically changing edge weights of the roadmap.

Appendix A

The Shortest Path on Graph

There are many methods to construct the shortest path between two vertices on a graph, such that Dijkstra's algorithm [11], breadth first search [25], and Bellman-Ford algorithm [26]. Bellman-Ford algorithm can deal with a graph in which some of the edge weights are negative number, but is generally slower than others for the same problem. Breadth first search handles a unweighted graph, and Dijkstra's algorithm solves for a non-negative weighted graph. The grid graph considered in this paper is regarded as the unweighted graph or weighted graph with weight 1 for all edges. In order to keep potential for future work such as expanding our work to non-grid graph, we decided to apply Dijkstra's algorithm to solve the shortest path problem.

Dijkstra's algorithm is shown in Algorithm 8. In a graph $\mathcal{G} = (V, E)$ with non-negative weights, the algorithm finds the shortest paths from a vertex v to every vertices u. The output is distance of the shortest path from v to u, $d_{\mathcal{G}}(v,u)$, and the parent pointers, $parent(u)$, for all vertices $u \in V$. The parent pointer represents the vertex which is the in-neighbor of u in the shortest path from v to u. If $u = v$, the parent pointer will show v because v is the root and does not have in-neighbor (i.e. $parent(v) = v$). For the grid graph, the weights of edges are 1 everywhere, that is, $weight(v,u) = 1, \forall (v,u) \in E$

Given the parent pointers from Algorithm 8, we can construct the shortest path from v to

Algorithm 8: Dijkstra's algorithm

Input : $\mathcal{G} = (V, E)$, v
Output : $parent(u)$ and $d_{\mathcal{G}}(v,u)$, $\forall u \in V$

for $u \in V$ **do**
  $d_{\mathcal{G}}(v,u) \leftarrow +\infty$
  $parent(u) \leftarrow u$
end
$d_{\mathcal{G}}(v,v) \leftarrow 0$
$Q \leftarrow V$
while $Q \neq \phi$ **do**
  $u \leftarrow \text{argmin}_{u \in Q} d_{\mathcal{G}}(v,u)$
  $Q \leftarrow Q \backslash u$
  for *each vertex w connected to u by an edge in E* **do**
    if $d_{\mathcal{G}}(v,w) > d_{\mathcal{G}}(v,u) + weight(u,w)$ **then**
      $d_{\mathcal{G}}(v,w) = d_{\mathcal{G}}(v,u) + weight(u,w)$
      $parent(w) = u$
    end
  end
end
return $parent(u)$ and $d_{\mathcal{G}}(v,u)$ for all vertices u in V

any vertex w by Algorithm 9, and the output, path(v,w), is an ordered sequence of vertices on the shortest path from v to w.

Algorithm 9: Extract-path algorithm

Input : $\mathcal{G} = (V, E)$, w, $parent(u)$, $u \in V$
Output : $path(v,w)$

$path(v,w) \leftarrow w$
$u \leftarrow w$
while $parent(u) \neq u$ **do**
  $u \leftarrow parent(u)$
  insert u at the beginning of $path(v,w)$
end
return $path(v,w)$

Computing the shortest path from robot i to goal j in the graph $\mathcal{G}$ by Algorithm 8 and 9, we can get $d_{\mathcal{G}}(x_i, g_j)$. The trajectory from robot i to goal j is denoted as $\xi_{i,j} = path(x_i, g_j)$. Note that if there are two or more shortest paths whose length is the same as each other, this algorithm

chooses one of them in accordance with the order of computation, which depends on indices of robots, goals, edges, and vertices.

Appendix B

Assignment Problem

B.1 Hungarian Algorithm

In general, the assignment problem is described as the problem to find a optimal assignment of a set of robots to a set of goals(tasks) to minimize the total distances (or cost), assuming the number of robots N is equal to the number of goals M, that is, $M = N$. Each robot can be assigned to at most one task, and each task can be performed by not more than one robot. The mathematical formulation of the problem is shown as an Integer Linear Programming (ILP) given by:

$$\text{minimize:} \quad \sum_{i=1}^{N}\sum_{j=1}^{N}\phi_{i,j}D_{i,j} \tag{B.1}$$

$$\text{subject to:} \quad \sum_{i=1}^{N}\phi_{i,j} = 1, \qquad j \in \{1,2,\cdots,N\} \tag{B.2}$$

$$\sum_{j=1}^{N}\phi_{i,j} = 1, \qquad i \in \{1,2,\cdots,N\} \tag{B.3}$$

$$\phi_{i,j} \in 0,1 \tag{B.4}$$

where, $\phi_{i,j} = 1$ if the robot i is assigned to the task j, otherwise $\phi_{i,j} = 0$

The Hungarian algorithm is one of the well known method to solve the linear assignment problem with time-complexity bounded polynominally in the number of robots, $\mathcal{O}(N^3)$ [27].

B.2 Conflict-Free Task Assignment

In [10], authors propose a centralized algorithm to find the set of assignment from all the combination of robots and goals such that the solution does not have any collisions while minimizing the total cost as much as possible. In this algorithm, the conflict graph described in Section 4.2, and [10],[23], is used for identifying possible conflicts. Let $\phi_i \in \{0,1\}^N$ be the assignment vector of robot i, which is $\phi_i = \{\phi_{i,1}, \phi_{i,2}, \cdots, \phi_{i,N}\}^T$, and $\phi \in \{0,1\}^{NN}$ be the vector collecting all the assignment vector, which is $\phi = \{\phi_1^T, \phi_2^T, \cdots, \phi_N^T\}^T$. The following constraint guarantees the assignment ϕ does not have any conflict:

$$\mathcal{I}^T(\mathcal{S})\phi \leq \mathbf{1}_N \tag{B.5}$$

where, $\mathcal{S}$ and $\mathcal{I}$ represent the conflict graph and the incidence matrix respectively. A proof of this proposition is shown in [10]. Adding the constraint (B.5) into the ILP problem (B.1), the constrained optimization problem, whose solution results in collision-free assignment, can be written as follows:

$$\text{minimize:} \qquad \sum_{i=1}^{N}\sum_{j=1}^{N} \phi_{i,j} D_{i,j} \tag{B.6}$$

$$\text{subject to:} \qquad \mathcal{I}^T(\mathcal{S})\phi \leq \mathbf{1}_N \tag{B.7}$$

$$\sum_{i=1}^{N} \phi_{i,j} = 1, \qquad j \in \{1,2,\cdots,N\} \tag{B.8}$$

$$\sum_{j=1}^{N} \phi_{i,j} = 1, \qquad i \in \{1,2,\cdots,N\} \tag{B.9}$$

$$\phi_{i,j} \in 0,1 \tag{B.10}$$

However, this ILP problem cannot always be solved because in some cases, any set of assignment may have collisions, as shown in Figure B.1. Therefore, we will propose a modified version of this ILP problem such that it can give the set of assignment to minimize the number of collision while minimizing the total cost in Section 4.2.

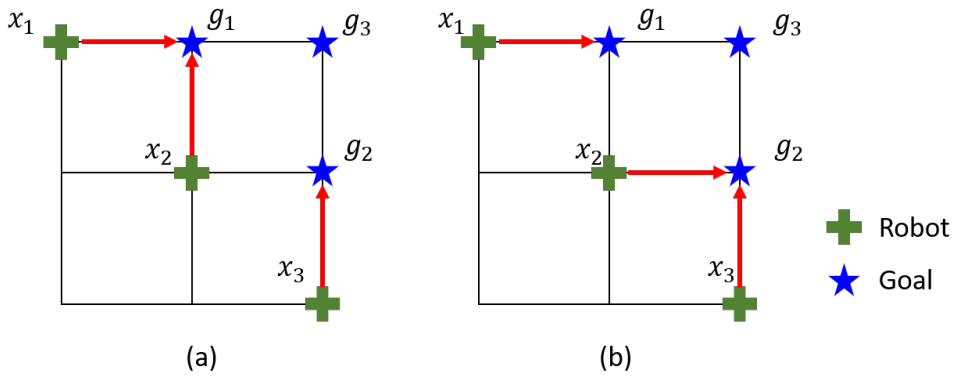


Figure B.1: A example of unsolvable configuration. For this initial condition, there are two choices for the first move; (a) and (b), but both definitely have the collision.

B.3 Decentralized Task Assignment

In this section, we introduce a decentralized task assignment algorithm, consensus-based auction approach (CBAA), proposed by Luc Brunet in [18].Some types of decentralized algorithms realize the task assignment to make use of a centralized planner that corrects state information from all of the robots in the system, compute the optimal assignments, and then distribute the results. However, in this type of algorithm, the centralized planner requires high computational capability, which leads to a lack of robustness due to the single point of failure. The CBAA makes use of two algorithms: auction algorithm and consensus algorithm, and can efficiently produce the sub-optimal solutions without such a centralized planner.

The CBAA consists of iterations between two phases. The first phase is the auction process, which is that each robot places a bid on a task, and then, the second phase, the consensus process, carries out to resolve conflicts and converge to the feasible assignment. In the consensus

process, all robots need to be connected on communication network directly or indirectly to resolve all conflicts over the system.

Auction Process

Algorithm 10 shows the procedure of robot i's auction process at iteration t where one iteration consists of a single run of both processes. In the auction process, each robot can make a bid asynchronously. Let $F_{i,j}$ be the bid that robot i places for goal j, which is equivalent to the objective functions in Section 5.2. If robot i does not have an assignment($\sum_j x_{i,j} = 0$), first it check availability vector σ_i whose jth entry is 1 if robot i is allowed to place a bid on goal j. To compute σ_i, we use the indicator function $\mathcal{I}(\cdot)$ that is unity if the argument is true and zero otherwise. Let x_i be the assignment vector, y_i be the lowest bid for each goal recognized by robot i, and z_i be the list which represents which robot is assigned to each goal. Within the set of goals which is $\sigma_{i,j} = 1$, robot chooses a goal J_i to minimize the bid, and updates x_i, y_i, and z_i for J_i.

Algorithm 10: Auction Process for robot i at iteration t

Input : $F_{i,j}, x_i(t-1), y_i(t-1), z_i(t-1)$
Output : $x_i(t), y_i(t), z_i(t), J_i$

if *t=0* **then**
  $x_{i,j}(t) = 0 \quad \forall j$
  $y_{i,j}(t) = \infty \quad \forall j$
  $z_{i,j}(t) = -1 \quad \forall j$
end
$x_i(t) = x_i(t-1)$
$y_i(t) = y_i(t-1)$
$z_i(t) = z_i(t-1)$
if $\sum_j x_{i,j}(t) = 0$ **then**
  $\sigma_{i,j} = \mathbb{I}(F_{i,j} < y_{i,j}(t)) \quad \forall j$
  $J_i = \text{argmin}_j \, \sigma_{i,j} \cdot F_{i,j}$
  $x_{i,J_i}(t) = 1$
  $y_{i,J_i}(t) = F_{i,J_i}$
  $z_{i,J_i}(t) = i$
end

Consensus Process

In the consensus process, each robot resolves a conflict of assignment and make a feasible assignment, which is that each robot is assigned to each robot without duplication. As shown in Algorithm 11, each robot first exchanges the lowest bid list y_i, and the assigned robot list z_i with neighborhoods $\mathcal{N}_i$. The lowest bid list y_i is updated in such a way that robot i replaces $y_{i,j}$ with the lowest value between itself and its neighborhood. Similarly, the assigned robot list z_i is also updated. Then, if $z_{i,J_i} \neq i$, it means robot i loses its assignment($x_{i,J_i} = 0$). If ties occur in determining $z_{i,j}$, each robot selects one of them based on the order of the robot index.

Algorithm 11: Consensus Process for robot i at iteration t

Input : $\mathcal{N}_i, x_i(t), y_i(t), z_i(t), J_i$
Output : $x_i(t), y_i(t), z_i(t)$

Send y_i and z_i to the neighborhood $\mathcal{N}_i$
Receive y_k and z_k from the neighborhood $\mathcal{N}_i$ for $k \in \mathcal{N}_i$
$y_{i,j}(t) = \min_{k \in \mathcal{N}_i \bigcup i} y_{k,j}(t) \qquad \forall j$
$z_{i,j} = \text{argmin}_{k \in \mathcal{N}_i \bigcup i} y_{k,j}(t) \qquad \forall j$
if $z_{i,J_i} \neq i$ **then**
    $x_{i,J_i} = 0$
end

Bibliography

[1] Michael Corkery and David Gelles. *Robots Welcome to Take Over, as Pandemic Accelerates Automation*. The New York Times. https://www.nytimes.com/2020/04/10/business/coronavirus-workplace-automation.html.

[2] Spyros G Tzafestas. *Introduction to mobile robot control*. Elsevier, 2013.

[3] Flavio Scalzo. *Tommy the robot nurse helps keep Italy doctors safe from coronavirus*. Reuters. https://www.reuters.com/article/us-health-coronavirus-italy-robots/tommy-the-robot-nurse-helps-keep-italy-doctors-safe-from-coronavirus-idUSKBN21J67Y.

[4] Matt Simon. *This Incredible Hospital Robot Is Saving Lives. Also, I Hate It*. Wired. https://www.wired.com/2015/02/incredible-hospital-robot-saving-lives-also-hate/.

[5] Robotics Business Review. *Robots Will Be Working in 50,000 Warehouses by 2025, Report Says*. https://www.roboticsbusinessreview.com/supply-chain/robots-will-be-working-in-50000-warehouses-by-2025-report-says/.

[6] Eugene Kagan, Nir Shvalb, and Irad Ben-Gal. *Autonomous Mobile Robots and Multi-Robot Systems: Motion-Planning, Communication, and Swarming*. John Wiley & Sons, 2019.

[7] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 2019.

[8] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*, 33(1):98–112, 2014.

[9] Dimitra Panagou, Matthew Turpin, and Vijay Kumar. Decentralized goal assignment and safe trajectory generation in multi-robot networks via multiple lyapunov functions. *IEEE Transactions on Automatic Control*, 2019.

[10] Lorenzo Sabattini, Valerio Digani, Cristian Secchi, and Cesare Fantuzzi. Optimized simultaneous conflict-free task assignment and path planning for multi-agv systems. *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1083–1088, 2017.

[11] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[12] Markus Jager and Bernhard Nebel. Decentralized collision avoidance, deadlock detection, and deadlock resolution for multiple mobile robots. *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, 3:1213–1219, 2001.

[13] Lucia Pallottino, Vincenzo G Scordio, Antonio Bicchi, and Emilio Frazzoli. Decentralized cooperative policy for conflict resolution in multivehicle systems. *IEEE Transactions on Robotics*, 23(6):1170–1183, 2007.

[14] Emilio Frazzoli, Lucia Pallottino, Vincenzo Scordio, and Antonio Bicchi. Decentralized cooperative conflict resolution for multiple nonholonomic vehicles. *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6048, 2005.

[15] Benjamin Gravell and Tyler Summers. Concurrent goal assignment and collision-free trajectory generation for multiple aerial robots. *IFAC-PapersOnLine*, 51(12):75–81, 2018.

[16] Stefano Giordani, Marin Lujak, and Francesco Martinelli. A distributed algorithm for the multi-robot task allocation problem. *International conference on industrial, engineering and other applications of applied intelligent systems*, pages 721–730, 2010.

[17] Matteo Golfarelli, Dario Maio, and Stefano Rizzi. Multi-agent path planning based on task-swap negotiation. *Proc. 16th UK Planning and Scheduling SIG Workshop*, pages 69–82, 1997.

[18] Luc Brunet, Han-Lim Choi, and Jonathan How. Consensus-based auction approaches for decentralized task assignment. *AIAA guidance, navigation and control conference and exhibit*, page 6839, 2008.

[19] Nathan Michael, Michael M Zavlanos, Vijay Kumar, and George J Pappas. Distributed multi-robot task assignment and formation control. *2008 IEEE International Conference on Robotics and Automation*, pages 128–133, 2008.

[20] Béla Bollobás. *Modern Graph Theory*. Graduate texts in mathematics. Springer, Heidelberg, corrected edition, 1998.

[21] Reinhard Diestel. *Graph Theory (Graduate Texts in Mathematics)*. Springer, August 2005.

[22] Francesco Bullo, Jorge Cortés, Florian Dörfler, and Sonia Martínez. *Lectures on Network Systems*. CreateSpace, 1 edition, 2018.

[23] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[24] Oliver Purwin, Raffaello D'Andrea, and Jin-Woo Lee. Theory and implementation of path planning by negotiation for decentralized agents. *Robotics and Autonomous Systems*, 56(5):422–436, 2008.

[25] Francesco Bullo, Jorge Cortés, and Sonia Martínez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Princeton University Press, USA, 2009.

[26] R. Bellman. On a routing problem. *Quarterly of Applied Mathematics*, pages 87–90, 1958.

[27] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.