MAE247 – Cooperative Control of Multi-Agent System (Spring 2023)

Yang Jiao

Electrical and Computer Engineering, University of California San Diego

13 June 2023

Review on:

CAPT: Concurrent Assignment and Planning of Trajectories for Multiple Robots

Supplementary material (code and simulation videos): https://github.com/Joyyy821/CAPT

I. Introduction

This paper presents solutions to the problem of Concurrent Assignment and Planning of Trajectories (CAPT) for a system of multiple robots, which aims to solve robot task assignments and collision-free trajectory planning simultaneously. This problem has many application scenarios such as formation control for a group of unmanned aerial vehicles (UAVs) and deployment of sensors such as camera and microphones around urban neighborhoods.

The authors propose both centralized and decentralized solutions to the CAPT problem and have proved both of them to be collision-free. The centralized CAPT algorithm (C-CAPT) is also optimal in terms of producing the minimum sum of distance squared trajectories. The authors also conducted massive experiments on the CAPT algorithms. The C-CAPT algorithm was implemented in both simulation and hardware experiments (by a team of quadrotor), and the performance of the decentralized algorithm (D-CAPT) was compared in different simulation settings (e.g., different scale of number of robots, different sensing ranges, etc.). Here I also re-implemented both C-CAPT and D-CAPT algorithms at a small scale (about 10 robots). The details of my simulation results are presented in Section IV.

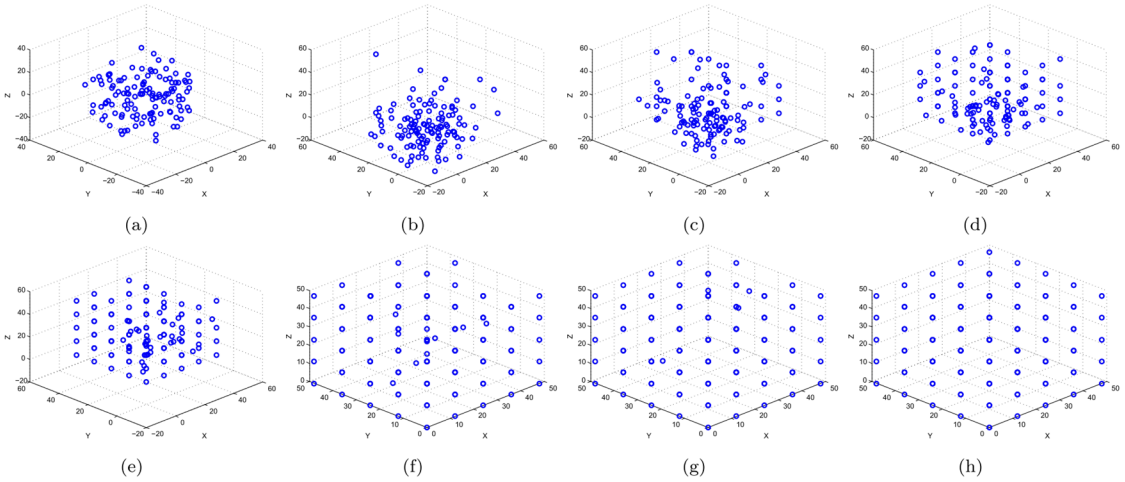
In the remaining of this report, we will use the term *robot* and *agent* interchangeably, as the paper considered CAPT problem for multiple robots while the context of robot is the same as the context of agent in our course.

II. Related Work

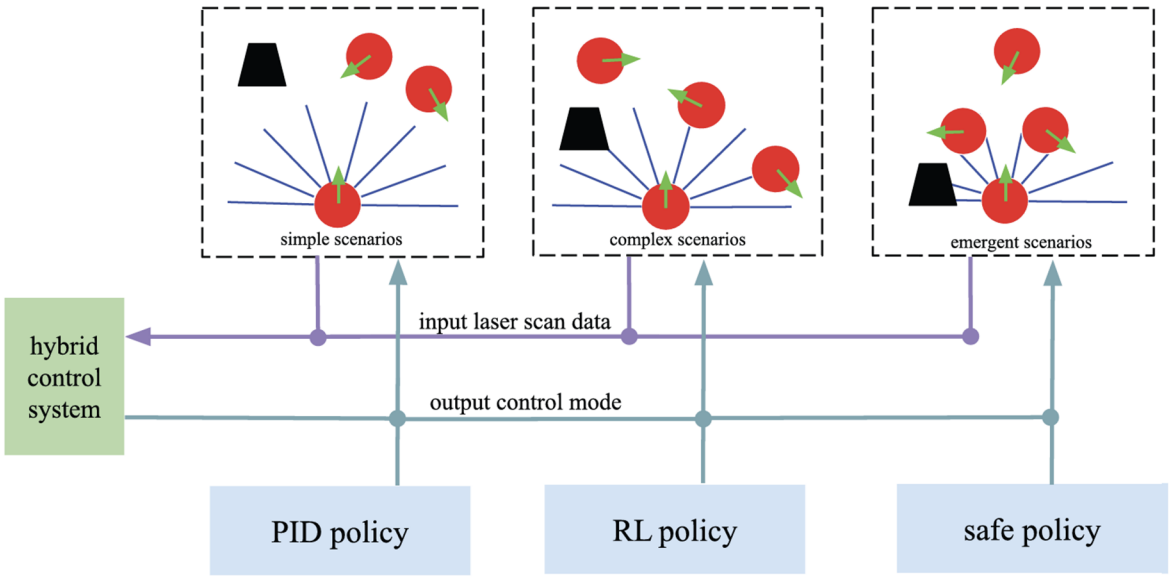
In this section, we first summarize some previous literature identified by the authors, and then introduce some state-of-the-art research works. Some of the previous research addressed either the assignment problem or the planning problem rather than solving them concurrently. Other existing algorithms that solved the full CAPT problem are mostly in centralized manner and are suboptimal, cannot guarantee convergence in polynomial time, or directly modeled the agents as mass points (which are problematic to apply on real robots with certain physical dimension).

Although the authors resolved most of issues from previous research, there are still some corner cases left to be pathological. Fig. 1 demonstrates two pieces of the most recent research on the CAPT algorithm. (J. Hu, et al) developed a decentralized algorithm based on the D-CAPT

algorithm with more detailed rules to coordinate the agents and worked out the corner cases (Fig. 1(a)). (T. Fan, et al) leverages the deep reinforcement learning for navigation. As demonstrated in Fig. 1(b), the hybrid design of control system applies different control policies based on the level of clearance of the robot neighborhood.



(a)



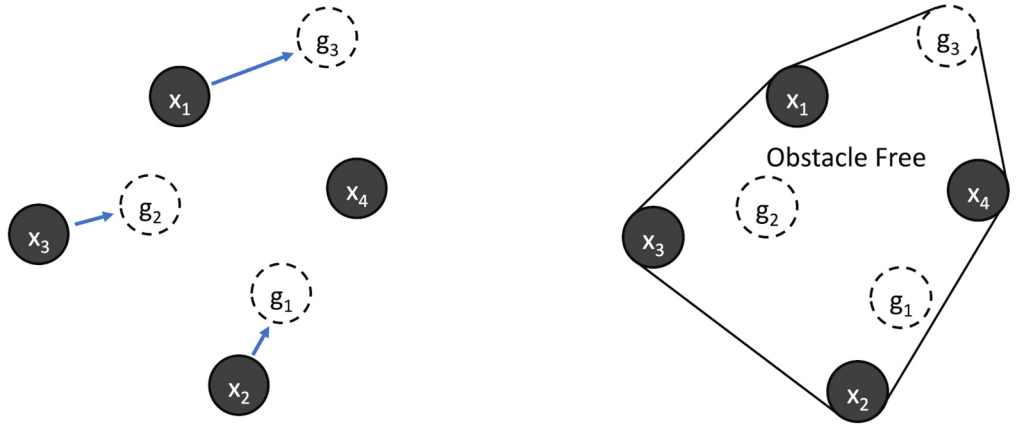
(b)

Fig. 1: Some state-of-the-art research on the CAPT problem

III. Algorithm and Proof Sketch

A. Problem Definition

We use N to denote number of robots and M to denote number of goals for the robots. n is the dimension of the workspace. Note that N and M do not have to be equal. For the case $N > M$, some of the robots will not have goal assigned to them and remain at their initial locations. For the case $N < M$, there will be some goals remain empty. E.g., in Fig. 2, $N = 4$, $M = 3$, and $n = 2$ (2-dimensional problem is considered).



(a) (b)

Fig. 2: (a) An example of the agents and goals in $\mathbb{R}^2$ and (b) the obstacle free workspace

The robots' locations are $x_i \in \mathbb{R}^n, \forall i = \{1, \dots, N\}$, and the goal locations are $g_i \in \mathbb{R}^n, \forall i = \{1, \dots, M\}$. The system vectors of robot states and goal states are the concatenated vectors:

$$\boldsymbol{X}(t) = [\boldsymbol{x}_1(t)^T, \boldsymbol{x}_2(t)^T, \dots, \boldsymbol{x}_N(t)^T]^T$$
$$\boldsymbol{G} = [\boldsymbol{g}_1^T, \boldsymbol{g}_2^T, \dots, \boldsymbol{g}_M^T]^T$$

As shown in Fig. 2(a), we need to assign the robots to the goals. This step is described by the assignment matrix $\phi \in \mathbb{R}^{N \times M}$ with each (i, j)-th entry to be:

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

In the paper, the authors defined the properties of the assignment matrix in detail. We can basically understand ϕ as a permutation matrix with zeros appended to fit in the correct shape $(N \times M)$. Some other notations we will introduce along with the general assumptions in the CAPT problem.

The general assumptions are as follows:

- Robots are homogeneous and interchangeable, confined in a ball with radius R (denoted as $\mathcal{B}_R$). Switching goal tasks is allowed for all of them;
- The workspace is an obstacle-free convex hull as presented in Fig. 2(b);
- Robots are controlled by a fully actuated first-order dynamics $\dot{\boldsymbol{x}}_i = \boldsymbol{u}_i$. No environmental disturbance is considered, and the robots always have perfect state knowledge.

B. C-CAPT Algorithm

To introduce the C-CAPT approach, the authors first considered the following cost function that gives minimum sum of distances trajectories:

$$\underset{\phi, \gamma(t)}{\text{minimize}} \sum_{i=1}^{N} \int_{t_0}^{t_f} \sqrt{\dot{\mathbf{x}}_i(t)^{\mathrm{T}} \dot{\mathbf{x}}_i(t)} dt$$

where $\gamma(t)$ denotes the trajectories of all agents. We can easily see that the optimal trajectories are intersection-free as illustrated in Fig. 3 and by triangular inequality:

$$\|\boldsymbol{r}_{2,1}\|_2 + \|\boldsymbol{r}_{1,2}\|_2 > \|\boldsymbol{r}_{1,1}\|_2 + \|\boldsymbol{r}_{2,2}\|_2$$

Hence, if there are two paths intersected, then we can always find another assignment by switching goals that leads to lower sum of distances.

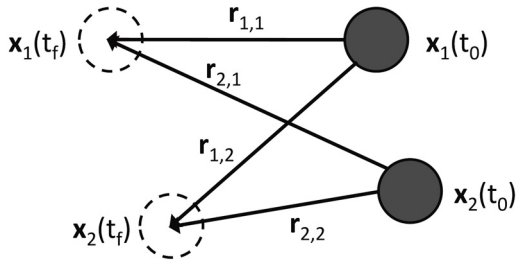


Fig. 3: Goal switching leading to non-intersecting paths and smaller sum of distances trajectories

However, intersection-free does not necessarily suggest collision-free. Fig. 3(a) depicts a simple example where two agents travelling in parallel and close to each other. The trajectories have no intersection but the two agents may encounter collision in the middle due to their physical dimension. On the contrary, switching goal assignments in Fig. 3(a) leads to intersected but collision-free trajectories as presented in Fig. 3(b). The agents will not experience collision if they are synchronized and x_2 leaves the area around g_2 while x_1 is approaching g_2 from the other side.

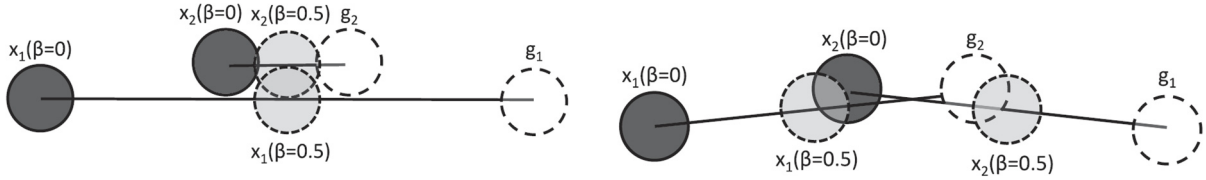


Fig 4: Examples for (a) intersection-free but not collision-free trajectories and (b) collision-free (if agents are synchronized) but intersected trajectories

This thus motivates the second method the authors proposed such that the sum of the integral of velocity squared (i.e., sum of distance squared) is minimized:

$$\underset{\phi,\gamma(t)}{\text{minimize}} \int_{t_0}^{t_f} \dot{X}(t)^{\mathrm{T}} \dot{X}(t) dt$$

The authors explained the intuition of designing such cost function using an analogy to a book-moving problem (Fig. 5). We consider to move the book denoted with the white star to the last position of the second row. The first method is to move this book directly, while the straight-line trajectory leads to collision with other books. However, we can also move this book to the first position of the second row, and shift all other books to the left with the width of one book simultaneously. This method gives us collision-free trajectories, and the sum of distance squared is smaller than the first method.

Therefore, C-CAPT algorithm uses this distance squared cost function, where this cost also has some nice properties as a strictly convex function. We can find the optimal distance squared assignment matrix ϕ^* first and derive the optimal trajectories $\gamma^*(t)$ (straight line segment) based on that, which leads us to the globally optimal solution. Here we summarize the C-CAPT solutions:

$$\phi^\star = \underset{\phi}{\operatorname{argmin}} \sum_{i=1}^{N} \sum_{j=1}^{M} \phi_{i,j} D_{i,j}$$

$$\gamma^\star(t) = (1-\beta(t))\mathbf{X}(t_0) + \beta(t)\Big(\Phi^\star \mathbf{G} + (I_{Nn} - \Phi^\star \Phi^{\star \mathrm{T}})\mathbf{X}(t_0)\Big)$$

where $\beta(t) = \alpha_0 + \alpha_1 t \in [0, 1\,]$, $\alpha_0 = \frac{-t_0}{t_f - t_0}$, $\alpha_1 = \frac{1}{t_f - t_0}$, and the termination time t_f is chosen by using maximum velocity v_{max}:

$$t_f = \underset{\mathrm{i}}{\operatorname{maximize}} \frac{\left\| \boldsymbol{x}_i(t_0) - \sum_{j=1}^{M} \phi_i^* \boldsymbol{g}_j \right\|_2}{v_{max}}$$

Note that $\phi^\star$ is the solution to a linear assignment problem, and can be solved by some classical algorithms such as Hungarian algorithm.

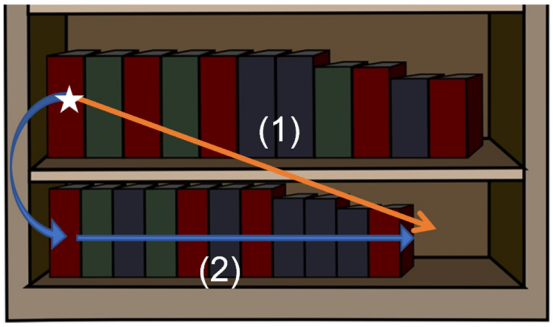


Fig 5: The moving-book example to explain the solution with minimum velocity squared trajectories

The authors proved that $\gamma^\star(t)$ is collision-free if the initial clearance Δ between robots satisfies $\Delta \geq 2\sqrt{2}R$. The ratio $2\sqrt{2}$ is determined given the condition that the centers of the nearest pair of robots should be separated by at least $2R$ during the travelling.

C. D-CAPT Algorithm

The decentralized version of the CAPT algorithm (D-CAPT) is proposed based on some theoretical results from C-CAPT. Before introducing the algorithm, here we impose two more constraints on D-CAPT problem:

- All of the agents have a certain sensing range h such that they can communicate with other agents when they are inside the sensing range of each other;
- $N = M$ is required. Each goal location is initially assigned to exactly one robot (e.g., if both robots and goals are labelled by 1 to N, then we can initially assign robot 1 to goal 1, robot 2 to goal 2, etc.

For every pair of agents, we then define the collision-free condition as below:

$$\mathbf{u}_{i,j}^T \mathbf{w}_{i,j} > 0 \qquad (*)$$

where $\mathbf{u}_{i,j} \equiv \mathbf{x}_j(t_c) - \mathbf{x}_i(t_c)$, $\mathbf{r}_{i,j} \equiv \mathbf{x}_j(t_f) - \mathbf{x}_i(t_c)$, and $\mathbf{w}_{ij} \equiv \mathbf{x}_j(t_f) - \mathbf{x}_i(t_f)$. By some equation simplification we can find that the collision-free condition is the same as:

$$\|\mathbf{r}_{i,i}\|_2^2 + \|\mathbf{r}_{j,j}\|_2^2 < \|\mathbf{r}_{i,j}\|_2^2 + \|\mathbf{r}_{j,i}\|_2^2$$

which is the minimum sum of distance squared as in C-CAPT.

Therefore, whenever we have a pair of agents (inside the sensing range of each other) that violates the collision-free condition, the two agents can switch the goal of each other. This locally minimizes the distance squared cost and resolves the potential local collision between the pair of the robots. Fig. 6 provides a high-level illustration of the D-CAPT algorithm.

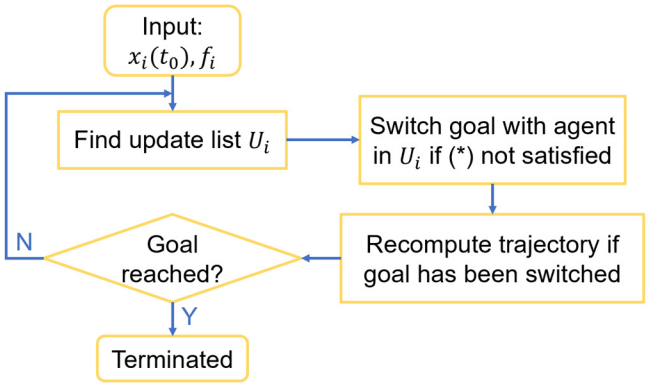


Fig. 6: The pipeline of the D-CAPT algorithm. f_i denotes the current goal assigned to robot i.

IV. Implementation

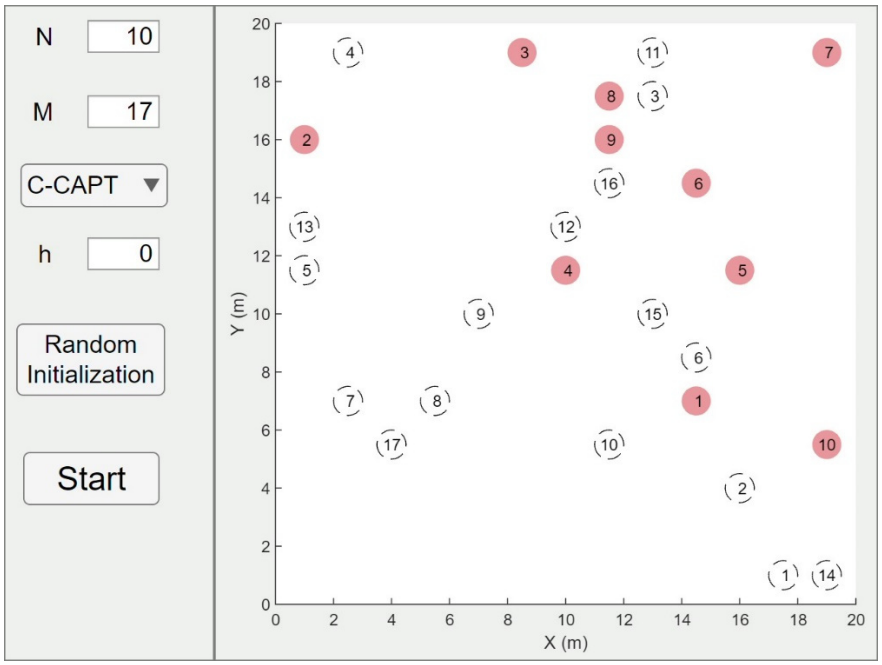


Fig. 7: The graphical user interface for implementation of the C-CAPT and D-CAPT simulations

I re-implemented the algorithms at a scale about 10 robots. The code and simulation videos can be accessed from the link at the beginning of this report. I set up the simulation with a 20×20m workspace (Fig. 7). Each robot has radius $R = 0.5$m. Fig. 8 presents the simulation results of C-CAPT. Fig. 9-11 show the results of D-CAPT with different setting of sensing range (which yields different level of decentralization).

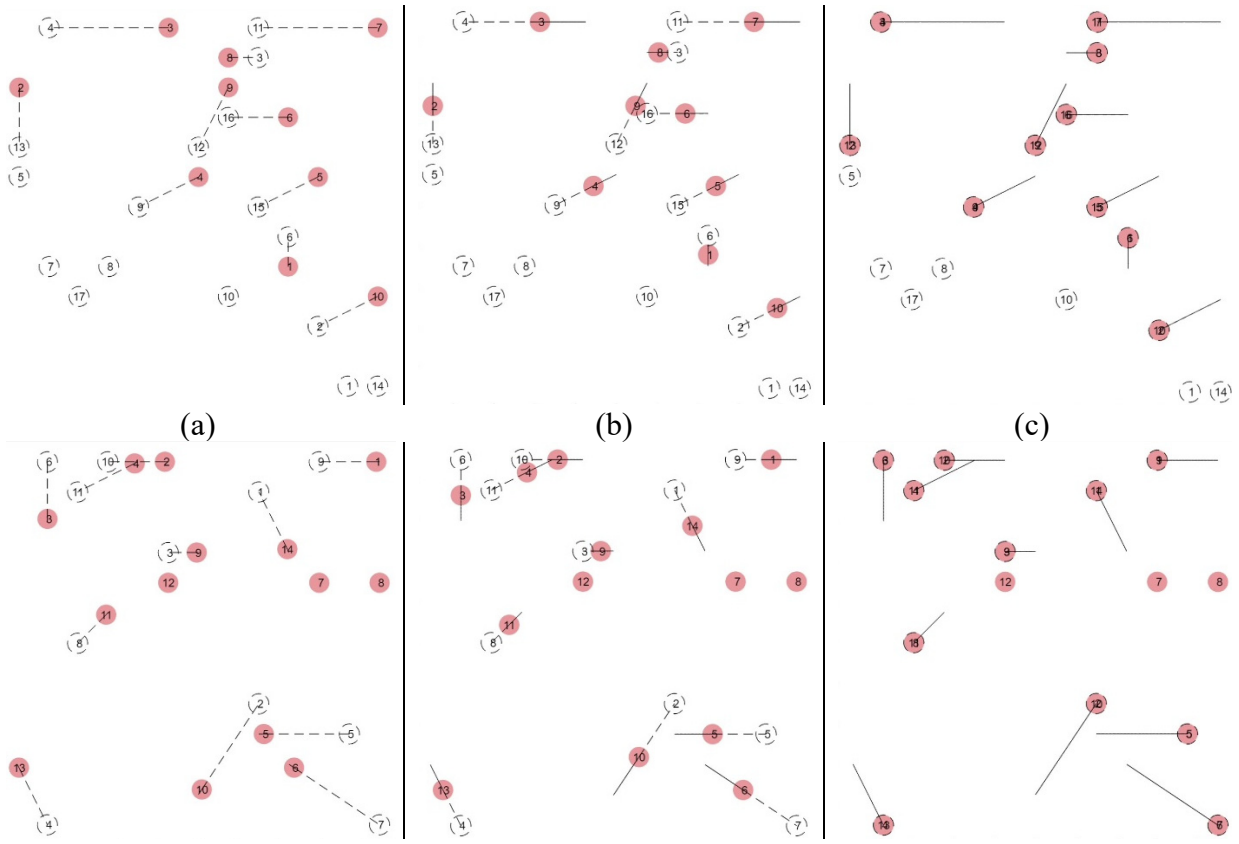


Fig. 8: The simulation procedure of the C-CAPT algorithm, where the optimal trajectories are collision-free straight-line segments. (a-c) C-CAPT with $N = 10$ and $M = 17$ (e-f) C-CAPT with $N = 12$ and $M = 11$.

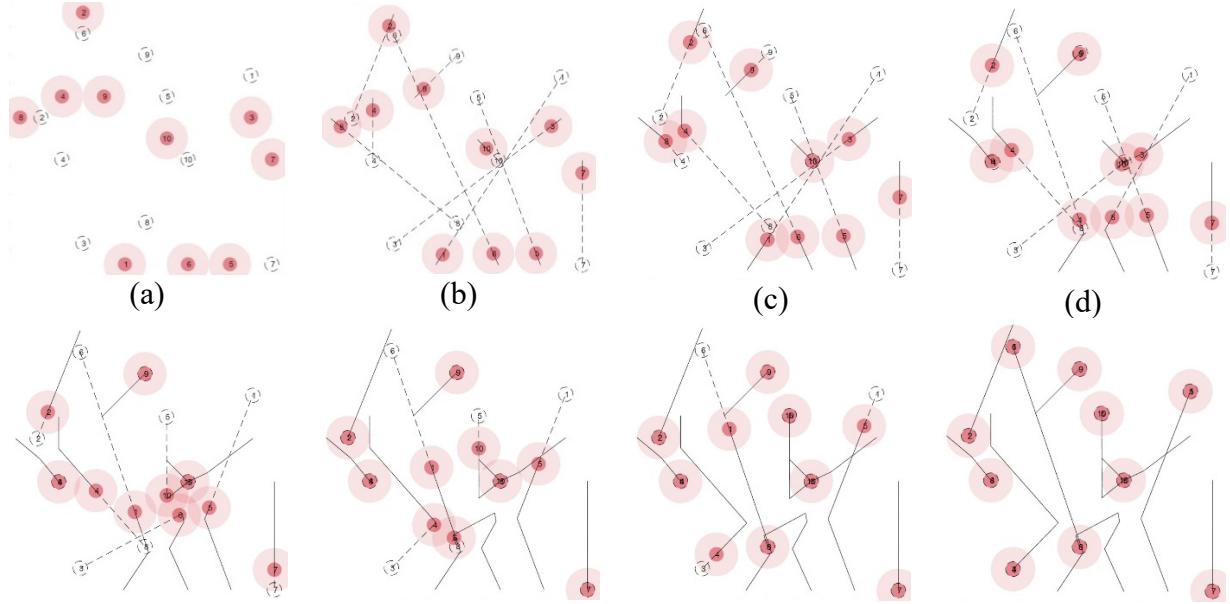


Fig. 9: The simulation procedure of the D-CAPT algorithm with $h = 1.5$m

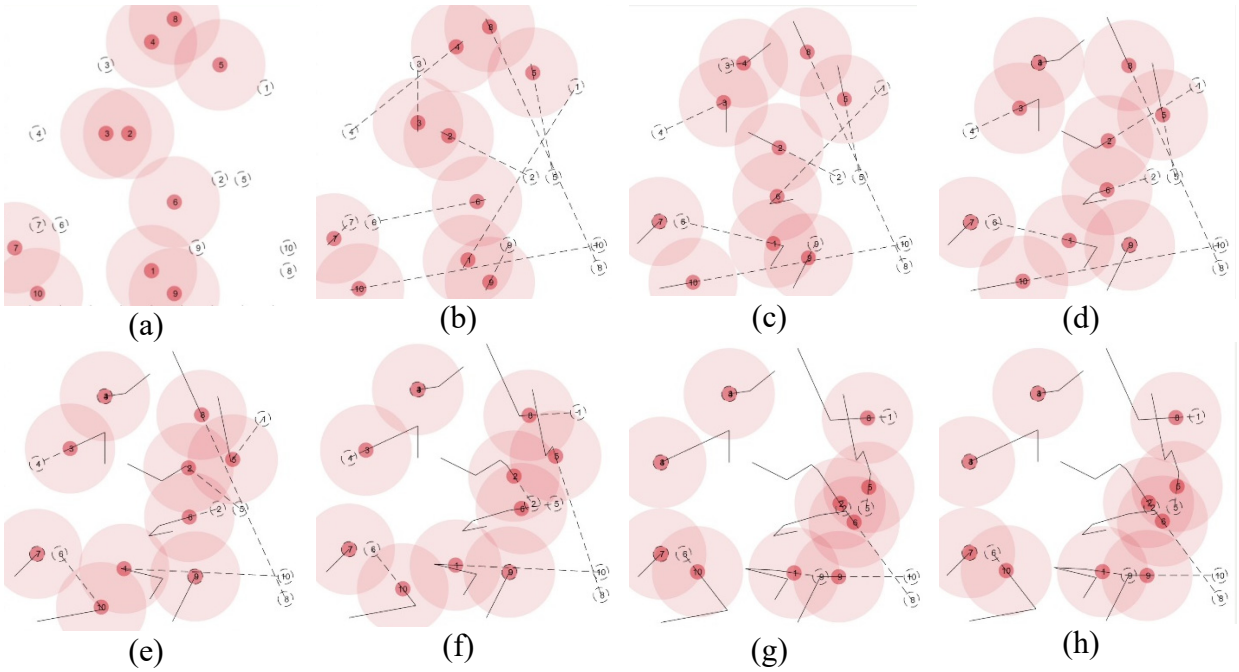


Fig. 10: The simulation procedure of the D-CAPT algorithm with $h = 3\text{m}$

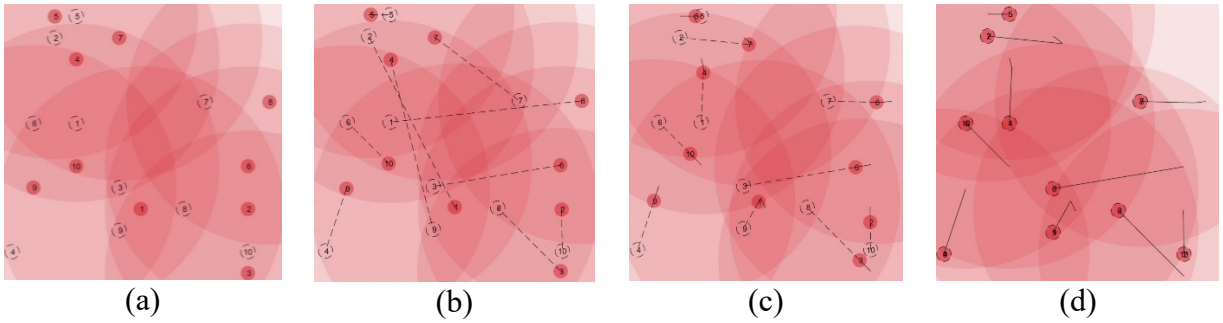


Fig. 11: The simulation procedure of the D-CAPT algorithm with $h = 10\text{m}$

In Fig. 9, we set $h = 1.5\text{m}$ where $\Delta = 2\sqrt{2}R = \sqrt{2}\text{m} \approx 1.414\text{m}$. With h slightly larger than the clearance distance Δ, the D-CAPT algorithm can still produces a collision-free solution. We can model the communication graph between the agents as a h-disk proximity graph. Increasing h yields more connectivity between the agents. The larger the h, the more communication burden the network has (and thus hurting the level of decentralization), but it also gives resulted trajectories closer to the optimal ones from C-CAPT. As we can see from Fig. 11, for $h = 10\text{m}$, almost all of the agents finished goal switching at the beginning part of their trajectories, and the results are very close to the optimal straight-line trajectories.

V. Conclusions and Findings

In this paper, the authors discussed two solutions to the CAPT problem. The centralized solution C-CAPT has been proved to be optimal and safe (collision-free), while the decentralized solution D-CAPT is collision-free and suboptimal. From the simulation results we can see a trade-off between the level of decentralization and optimality of the solution.

However, there are also some minor problems on the proposed algorithms. In the D-CAPT algorithm, since the h-disk proximity graph is not guaranteed to be connected when the agents reaching their goals, the agents may not be able to pass messages to each other and a single agent cannot have the knowledge about the termination time for the entire group. The termination criterion is actually essential for real-world applications as the robots may be assigned consecutive tasks after reaching their goal locations. Another issue for the C-CAPT algorithm is that the C-CAPT solution is only optimal with respect to the distance squared cost function. The cost function cannot easily migrate to other aspects (e.g., minimizing the consumed energy or minimizing the time for travelling) since collision-free cannot be guaranteed once we modify the objective function. The authors also assume the perfect first-order dynamics for the agents, i.e., any valid velocity command can be executed without time delay. However, the agent's velocity is always changing continuously in real-world, which means the authors ignored the acceleration and de-acceleration process that can essentially consume a lot of energy.

References

M. Turpin, N. Michael, V. Kumar, Capt: Concurrent assignment and planning of trajectories for multiple robots. *The International Journal of Robotics Research*. 2014;33(1):98-112. doi:10.1177/0278364913515307

J. Hu, H. Zhang, L. Liu, X. Zhu, C. Zhao and Q. Pan, Convergent Multiagent Formation Control With Collision Avoidance. *IEEE Transactions on Robotics*. 2020:36(6):1805-1818. doi:10.1109/TRO.2020.2998766

T. Fan, P. Long, W. Liu, J. Pan, Distributed multi-robot collision avoidance via deep reinforcement learning for navigation in complex scenarios. *The International Journal of Robotics Research*. 2020;39(7):856-892. doi:10.1177/0278364920916531