Project 3 - MEAM 620 Advanced Robotics
Multi-Robot Path Planning

Ankit Vora - ankitvor@seas.upenn.edu ,
Anushree Singh - anusingh@seas.upenn.edu,
Vishnu Purushothaman Sreenivasan - visp@seas.upenn.edu

May 5, 2015

1 Phase 1

We used the Hungarian assignment algorithm described in this paper [2], and adapted the matlab implementation from [1]. For estimating the run-time of the robot teams we calculated the mean over 10 trails. Following are the plots of the computation time versus the number of quad-rotors flying as a team . We computed the run-time for 2 parts.

- For running the assignment algorithm (Hungarian).
- For the assignment problem plus computation of trajectories for the quads in 2D.

We observed the expected trend , i.e N^3 where N is the number of the quad-rotors.

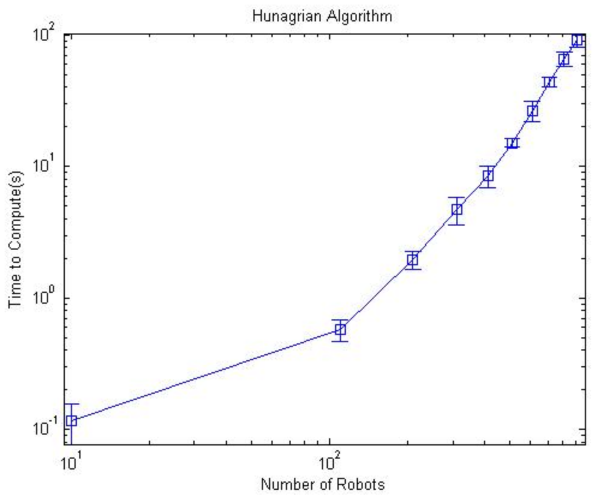


Figure 1: Plot for Run-time of Hungarian Algorithm versus number of robots

These figures resemble the results expressed in the paper [3] .
Further we implemented this algorithm on 2D robots in a 2 dimensional world. The step by step simulation snapshots are shown in figure 3. As expected the C-CAPT solutions provides with collision free paths.

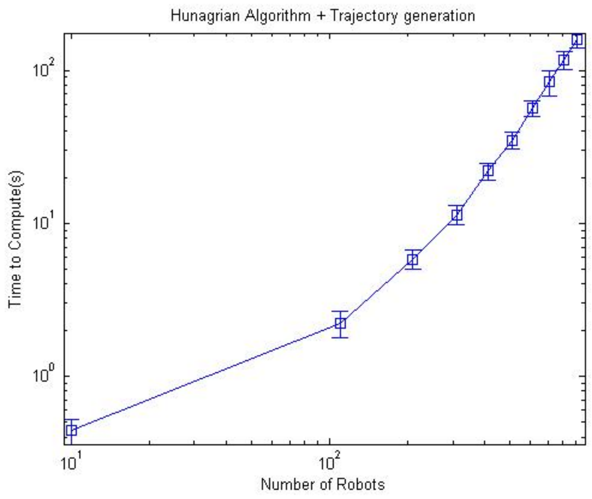


Figure 2: Plot for Goal assignment in C-CAPT via snapshots of the trajectory taken

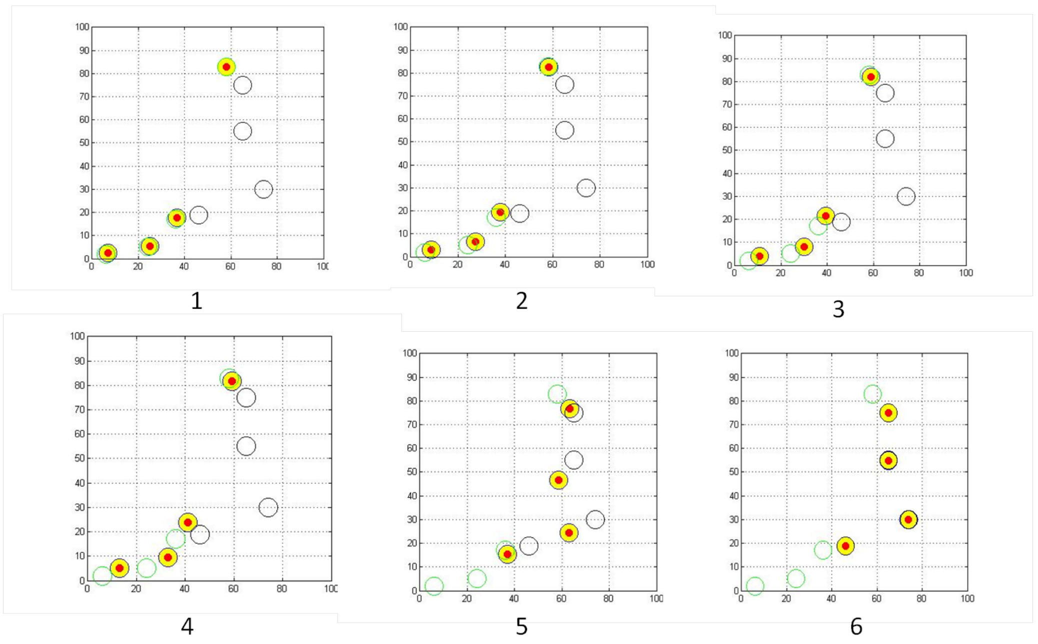


Figure 3: Plot for Goal assignment in D-CAPT via snapshots of the trajectory takens

2 Phase 2

We extended the 2D C-CAPT to 3D and ran the cases as below. The following images show the instantaneous states of the quadrotors in two different cases:

1. When the number of quadrotors are more than the number of goals.
2. And when the number of goals are twice the number of quadrotors.

a) For the case when the number of robots are more than the number of goals, we ran a simulation consisting of 15 robots and 10 goals. The algorithm minimized the cost function and assigned 10 robots

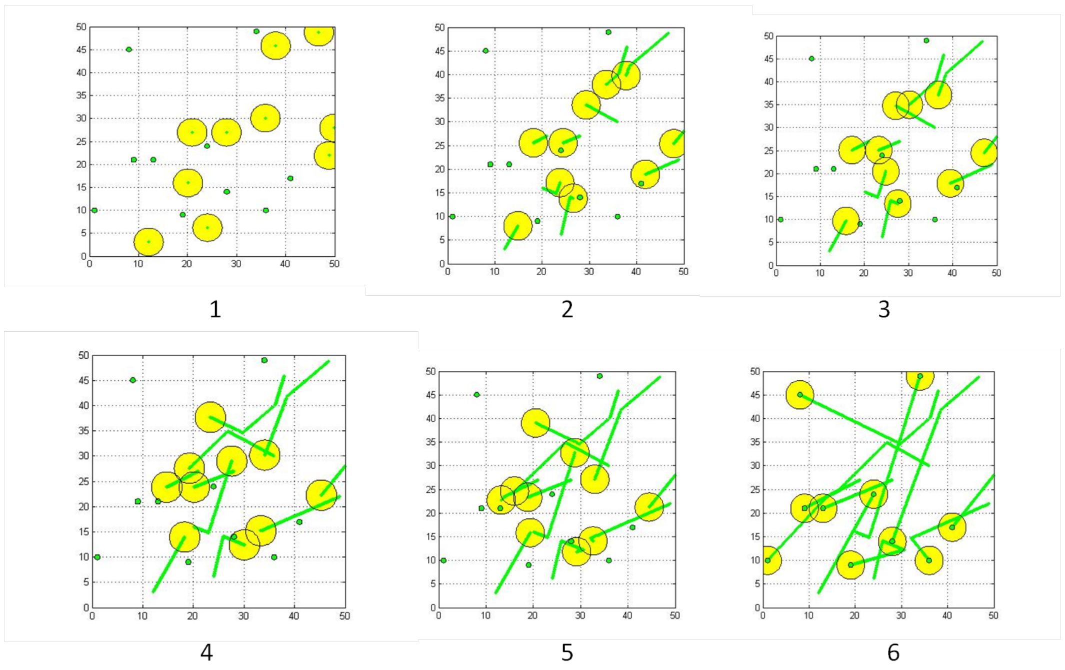


Figure 4: Plot for Run-time of Hungarian Algorithm and the generation of Trajectory co-efficients versus number of robots

to 10 goals in the most optimum way. The other 5 robots stay at their starting positions for the entire period.

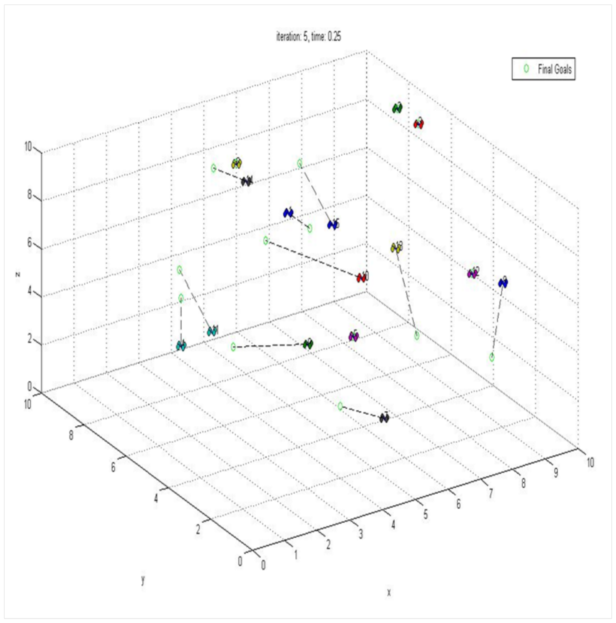
As we see in the figure 5(a), the starting position of the robots, goal position and robot-goal assignment is clearly depicted. Figure 5(b) shows the end of simulation where all the 10 robots have reached their respective goals and the remaining 5 robots stay in their initial position.

b) Another case that was studied during this project was the scenario where number of goals are more than the number of robots. In these type of cases our primary objective is to cover all the goals. The algorithm was implemented to ensure all the goals were visited with optimal trajectories. A simulation of 15 robots covering a total of 30 robots was conducted. As shown in the figure 6(a), the robots are assigned to most optimal 15 goals. Figure 6(b) shows the robots in their intermediate location with new goals assigned. Figure 6(c) shows the robots at their final goal positions after the completion of all goal positions.

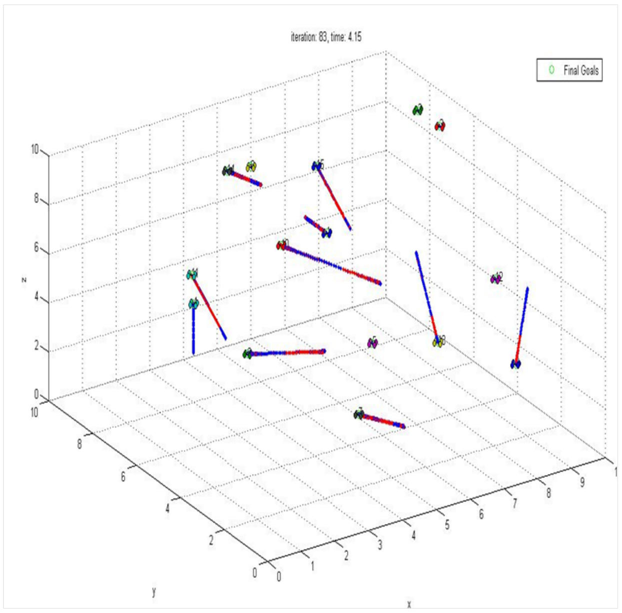
2.1 D-CAPT

We then extended the C-CAPT algorithm to the decentralized D-CAPT algorithm. We impose the same assumption of no obstacles in the convex hull covering the starting and the goal locations. In the decentralized version all the robots are assumed to be in sufficiently far away from each other so that robots can compute their trajectories without considering collisions and interactions with other robots. Initially all robots are randomly assigned a goal and further we make an assumption that the number of robots is the same as the number of goals. The robots are assumed to have a field of vision within which they can send and receive messages from other robots. So when two robots come within each other's field they locally optimize their trajectories with respect to the other robots in the field by changing their goal locations so that they locally minimize the squared distance loss function.

The series of snapshots from 2D simulation are presented in the figure 4. The yellow region represents



a



b

Figure 5: Plot for Run-time of Hungarian Algorithm and the generation of Trajectory co-efficients versus number of robots

range of communication of the robots and green dots represent the goal positions. As seen from the series of images, the robots move towards their randomly assigned goal positions (Trajectory depicted by the green line) and when other robots communicate that they are close enough, the interacting robots deviate from their paths reassigning their goal locations mutually.

As a final integration we incorporated the D-CAPT algorithm in a simulation where a team of quadrotors(nanoplus) are required to navigate to a set of goal locations. In order to comply with the minimum snap trajectories associated with quadrotor dynamics we utilize a septic 7th order time parametrization for the robot trajectories. We also respect the aerodynamic interactions by incorporating the ellipsoidal model of the quadrotor. As the paper suggests we pacify the effects of the downwash by maintaining the $8R$ separation in the z axis using a change of basis in the z dimension. We downscale the environment by 4 and obtain the solution but plot the results in the original space.

Advantages and Disadvantages of D-CAPT

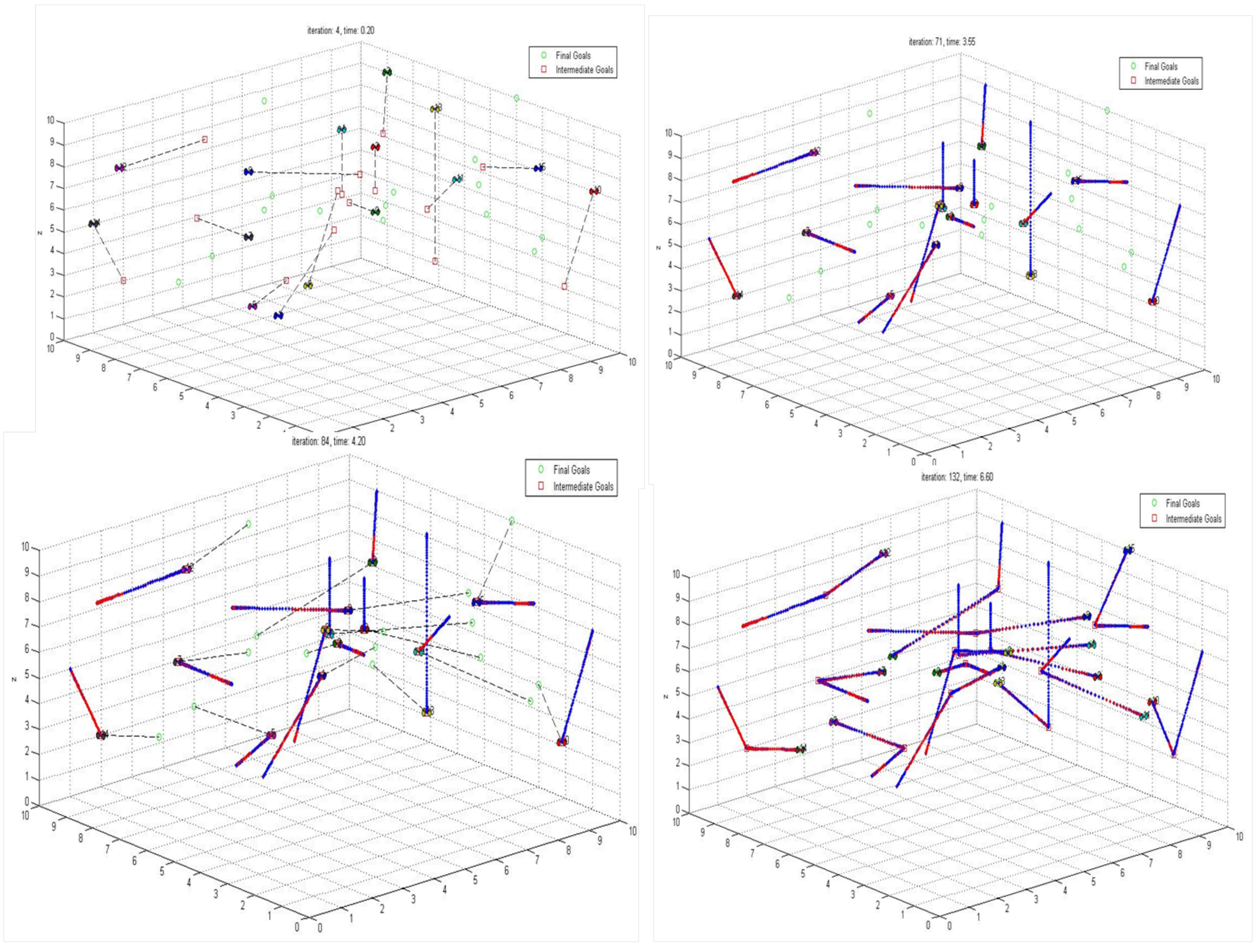


Figure 6: Plot for Run-time of Hungarian Algorithm and the generation of Trajectory co-efficients versus number of robots

Advantages	Disadvantages
Highly scalable, the computational power grows linearly with the number of robots, as each robot can locally optimize its trajectory in parallel.	It is suboptimal, C-CAPT definitely provides a much better optimal solution.
There is no need for a large centralized computational unit.	There is no easy way to extend the algorithm for a scenario with unequal number of goals and robots.
Irrespective of the drawbacks some of the D-CAPT is probably the most affordable and practical solution for real world problems	Since there is constant reassignment of goals, if the robots have complex dynamics the controller may not be able to track the trajectory generated by D-CAPT.

Test Cases

We consider three test cases as described below

1. **Robots on a collision path with a varying h**
   We placed two quadrotors in the same x and y coordinates but one vertically above the other 20m apart. We set their goals as each other's starting position. We find that even if the trajectory realizes

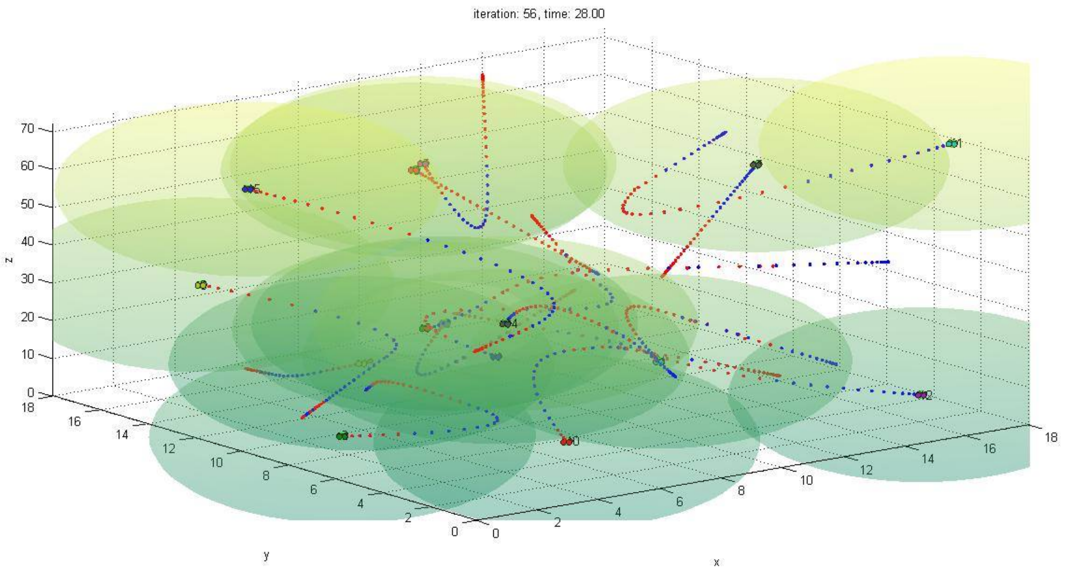


Figure 7: D-CAPT simulation for 15 robots in a (10, 10, 40)m space. The green circle markers indicate the goal locations. The red dotted line indicates the traversed trajectory and the blue dotted line indicates the trajectory generated D-CAPT. The green-yellow transparent ellipses indicate the communication range of each robot.

the suboptimality of the robots trajectories swaps their goal locations, the robots momentum is too large for them to stabilize and follow the trajectory issued by the generator. Increasing the h from 3m to 8m solved the issue because the controller had extra time to stabilise the quadotor and make it follow the new trajectory.

2. **Exposing suboptimal behaviour in the presence of dynamics**
   In our next test case, we try to elicit suboptimal behaviour from 2 quadrotors running the D-CAPT algorithm. Two robots are placed at $(0, 7, 0)$ and $(10, 3, 0)$ and are given the goals $(10, 7, 6)$ and $(0, 3, 6)$ respectively. When the robots come within the communication range they swap their goal positions. But due to the momentum gathered by the robots, the controllers were unable to immediately turn to the closest goals and ended up taking a long curve which resulted in a path much larger than it would have taken for the initial straight line paths. Though the algorithm is minimizing the cost function of the squared distance, the limitations of the controller make the quadrotor to produce worse results after the reassignment than the initial suboptimal assignment.

3. **Large number of robots**
   We found that in the case of large number of robots the random initialization initially play a major role. We found that in most of the simulations the robots were seen constantly changing their trajectories and producing suboptimal behaviour. An example of a simulated trajectory can be seen in figure 7. We posit that using greedy initial assignment of goals would produce better performance than complete random assignment. For example consider two robots each which is extremely close to two goal locations. A greedy assignment would have been optimal in this case whereas the random assignment imposes 50 percent chance of a suboptimal solution.

3 Conclusion

We have implemented the C-CAPT and D-CAPT algorithm as described in the paper [3]. We simulated the algorithms in 2D and 3D. We also simulated the algorithm for a team of KMel nanoplus quadrotor and described our findings. We also have enlisted the observed advantages and the disadvantages of the decentralized D-CAPT algorithm from our test cases. We suggested a plausible improvement to the algorithm using greedy assignment of goals instead of the random initial assignment as proposed by the paper.

References

[1] http://www.mathworks.com/matlabcentral/fileexchange/20652-hungarian-algorithm-for-linear-assignment-problems–v2-3-.

[2] James Munkres. Algorithms for the Assignment and Transportation Problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38, 1957.

[3] Matthew Turpin, Nathan Michael, and Vijay Kumar. Capt: Concurrent assignment and planning of trajectories for multiple robots. *Int. J. Rob. Res.*, 33(1):98–112, January 2014.