Multi agent Path Planning for Proximal Coverage of Agricultural farms

Final Presentation

Suhrudh. S

16-12-2022

Friday

Structure

1. Introduction
2. Multiple Cluster Coverage
3. Multi-agent Multiple Cluster Coverage
4. Hardware Experiments
5. Conclusion and Future Work

Introduction

- Motivation
- Proximal Points
- Contribution

Motivation

* Agriculture needs automation and technology.
* Use of advanced sensing in UAVs and UGVs allows us to perform perception tasks efficiently.
* Solve the bottle neck of approaching the biomass "proximally".
* Given such proximal points, exploit the environmental conditions to perform effective path planning

Proximal Points

- Locations that an UAV can reach around a biomass safely without collision.
- Each cluster has a set of Proximal Points.

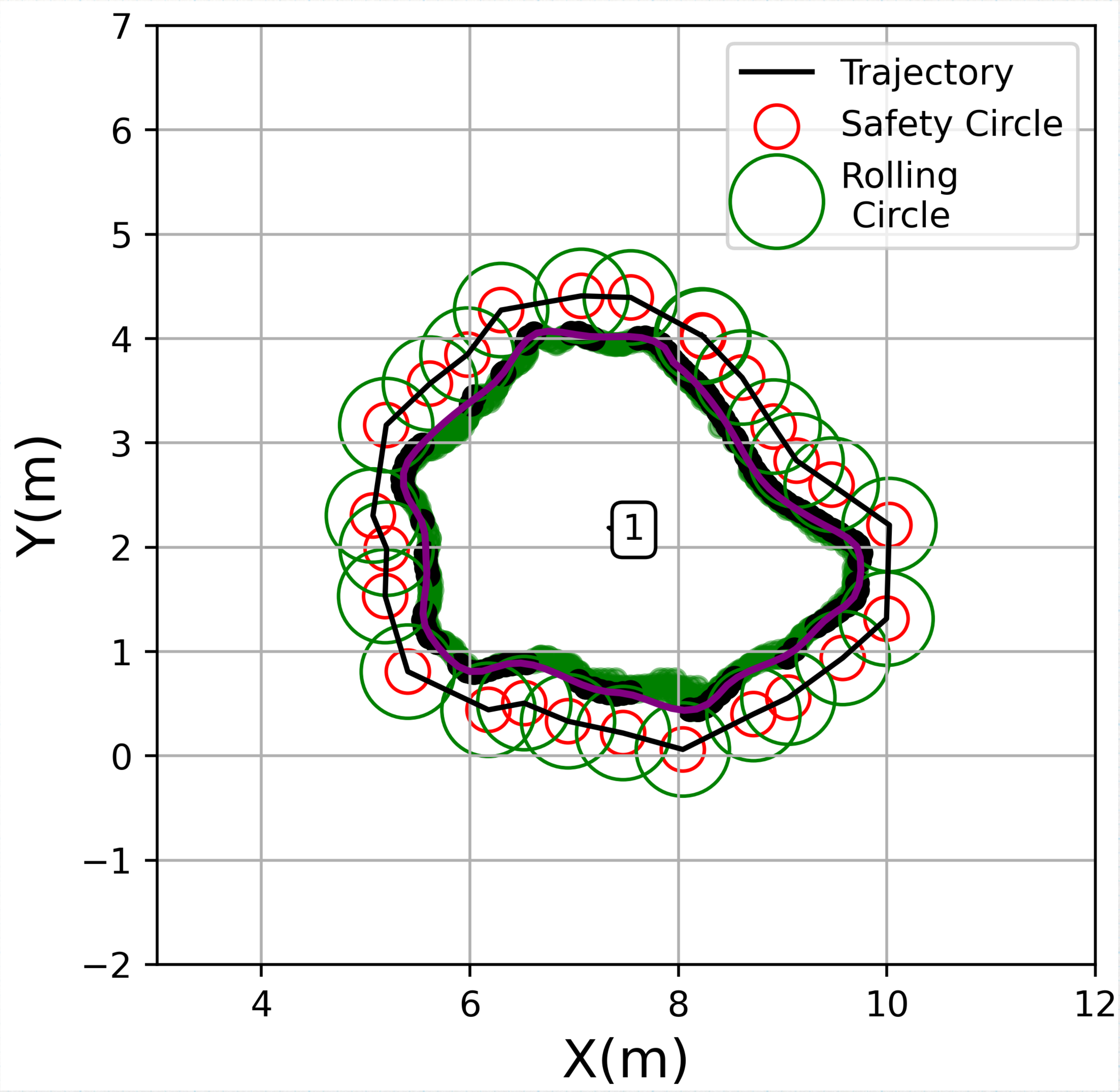


Fig 1: Proximal points for a single cluster

Contribution

* Given multiple clusters, computing the shortest path for all the Proximal points is analogous to TSP and is NP-Hard.
* Standard TSP solvers do not exploit geometric arrangement of the grid arrangement in farms to compute.
* We impose certain constraints and obtain near optimal paths with less time and compute.
* We then extend our method to deploy Multiple agents using existing solutions to obtain near optimal solutions.

Multiple Cluster Coverage

- Problem Setup
- Similarity to TSP
- Method
- Proposed Approach
- Results and Discussions

Problem Setup

- Given K clusters, arranged in a $M \times N$ Grid, the set of Proximal points $\omega \in \Omega$, a valid path σ comprises all the waypoints traversed at least once.
- Given a cost function c and set of all feasible paths Σ, the optimal path $\sigma*$ is defined as

$$\sigma * = \arg\min_{\sigma}\{c(\sigma) \,|\, \forall \omega \in \Omega\}$$

Similarity to TSP

* Given nodes i, j and the cost of travelling from i to j be c_{ij}, we can formulate the TSP problem as

$$\min \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} c_{ij}x_{ij}$$

$$\text{S.T,} \sum_{i=1, i\neq j}^{n} x_{ij} = 1$$

$$\sum_{j=1, j\neq i}^{n} x_{ij} = 1$$

* Where x_{ij} is the variable that decides whether an edge connects node i and j.

$$x_{ij} = \begin{cases} 1 & \text{if edge connects i and j} \\ 0 & else \end{cases}$$

Method

Problem Formulation: Preliminary

- Let us denote the Proximal points belonging to a cluster to be P_i^K.
  - The cost to cover one cluster K can be written as
    $$C^k = \sum_{i=0}^{n-1} ||P_{i+1}^k - P_i^k||$$
  - The cost to "switch" from one cluster to another can be written as $C_{ij}^{kl} = ||P_i^k - P_j^l||$.

Method

Problem Formulation: Cost

* The cost of a path σ can be given as a sum of costs required to cover one cluster and the cost to switch from one cluster to another.

* Hence, we can formulate our problem as

$$\min \sum_{i=0}^{K} C^k + \sum_{i} \sum_{j} C_{ij}^{kl} x_{ij}$$

$$\text{S.T,} \sum_{i=1, i \neq j}^{n} x_{ij} = 1$$

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1$$

* Similar to TSP, x_{ij} holds the power of calculating the solution

Proposed Approach

Graph Construction and Graph Traversal

* We convert the problem into a Graph traversal problem.
* The proposed solution can be split into two parts
  * Graph Construction
  * Graph Traversal

Proposed Approach

Constraints

* To achieve near optimal solutions, we impose the following constraints
    1. Any Proximal point can be traversed more than once.
    2. Any Proximal point can only be connected to two neighbours.
    3. Two clusters can be connected only by a predefined switching edge.
    4. The agent is constrained to always switch at the switching edge.

Proposed Approach

Graph Construction: Within Cluster

* Given the Proximal points P^n for the n-th cluster, we construct a graph $G_\phi(V_\phi, E_\phi)$.

* For vertices $v_i^n \in P^n$

$$
\begin{aligned}
v_{i-}^n &= P^n_{(i-1) \mod S} \\
v_{i+}^n &= P^n_{(i+1) \mod S} \\
\text{where, } S &= |P^n| \\
\phi^n_{i,i-} &= (v_i^n, v_{i-}^n) \\
\phi^n_{i,i+} &= (v_i^n, v_{i+}^n)
\end{aligned}
$$

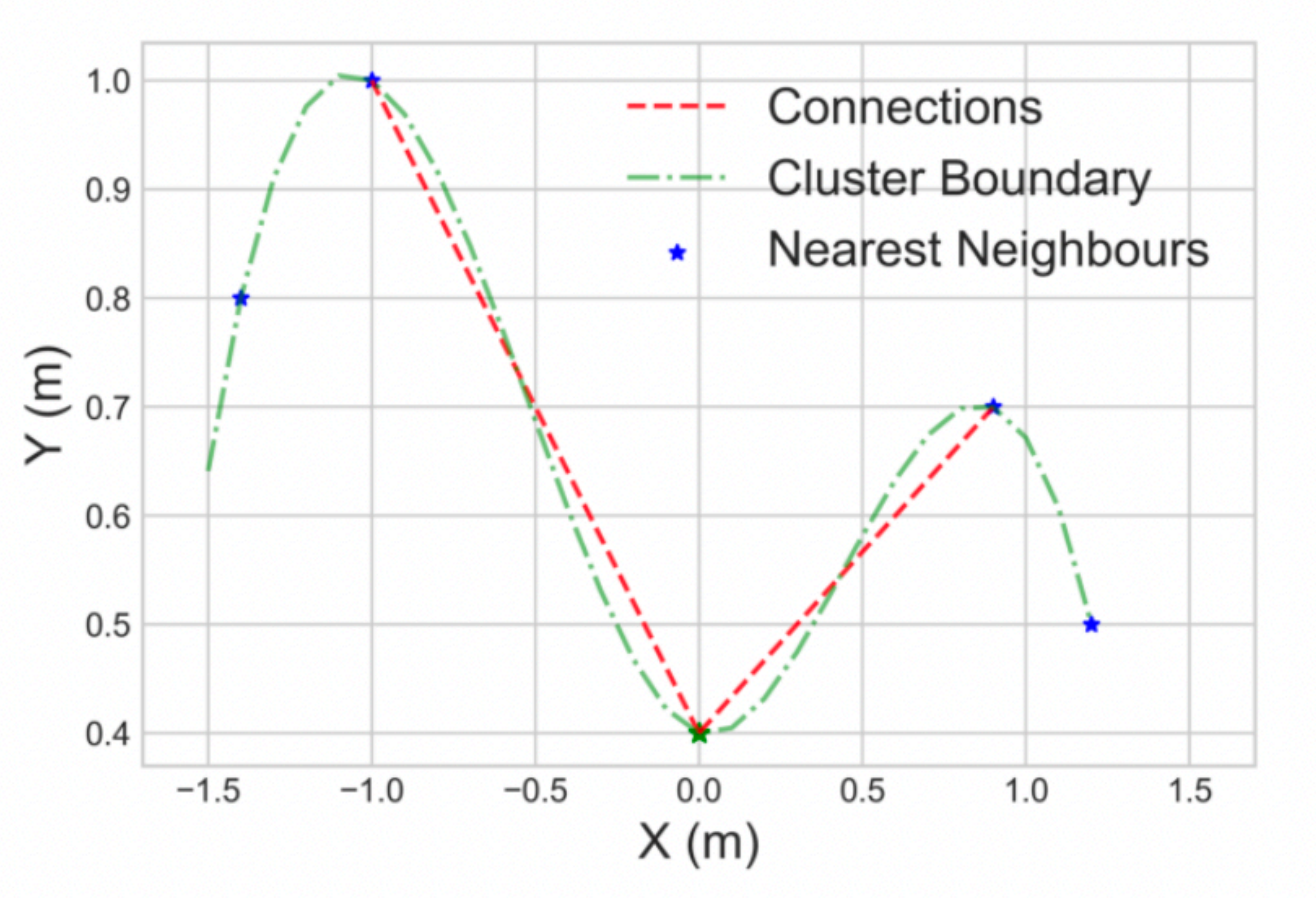


Fig 2: Nearest Neighbours for an example set

Proposed Approach

Graph Construction: Within Cluster

* For all the points P^n, we can define the set of all edges that completely cover the the cluster forming a cyclic graph.

$$\Phi^n = \bigcup_{i=1}^{S} \{\phi^n_{i,i-}, \phi^n_{i,i+}\}$$

* Finally, the set V_ϕ and E_ϕ can be defined as

$$V_\phi = \bigcup_{n=1}^{N} P^n$$

$$E_\phi = \bigcup_{n=1}^{N} \Phi^n$$

Proposed Approach

Graph Construction: Switching Edges

* As defined previously, a switching edge is defined as

$$\psi^{n,m} = (P_i^n, P_j^m)$$

* For any cluster, the center and radius of the cluster can be defined as

$$\mu^n = \frac{\Sigma P_i^n}{|P^n|}$$
$$r^n = \max_{p \in P^n} ||\mu^n - p||$$

* Given the depot D, we construct another Graph $G_\psi(V_\psi, E_\psi)$ using the centers of the clusters

$$V_\psi = \{\bigcup_{n=0}^{K} \mu^n\} \cup \{D\}$$
$$E_\psi = \{(v^i, v^j) \mid v^i, v^j \in V_\psi \text{ and } v^i \neq v^j\}$$

Proposed Approach

Graph Construction: Switching Edges

* Given the cost function,

$$C = \sum_{i=0}^{K} C^k + \sum_i \sum_j C_{ij}^{kl} x_{ij}$$

* Following our constraints, the cost function becomes

$$C = \sum_{i=0}^{K} C^k + 2 \sum_{(i,j) \in E_\psi} C_{ij}$$

* To minimise the cost function is to minimise

$$\min C \implies \min \sum_{(i,j) \in E_\psi} C_{ij}$$

Proposed Approach

Graph Construction: Switching Edges

* Hence, we construct a Minimum Spanning Tree of the graph G_ψ to obtain the minimum weighted edges E^*_ψ

* Using the Edges in E^*_ψ we find the nearest points in the cluster and connect them as switching edges

* Given the new edges, all the edges connecting alternate clusters are $\Psi^{ij} = \{\hat{e} \mid \forall e \in E^*_\psi\}$

* Finally,

$$
\begin{gathered}
e = (v^n, v^m) \\
\mu^n, r^n = v^n \\
\mu^m, r^m = v^m \\
\hat{\mathbf{x}}_{mn} = (v^m - v^n)/||v^m - v^n|| \\
x^n = v^n + r^n \cdot \hat{\mathbf{x}}_m n \\
x^m = v^m - r^m \cdot \hat{\mathbf{x}}_m n \\
\hat{v}^n = \underset{v \in V\phi}{||x^n - v||} \\
\hat{v}^m = \underset{v \in V\phi}{||x^m - v||} \\
\hat{e} = (\hat{v}^n, \hat{v}^m)
\end{gathered}
$$

$$
\begin{gathered}
V = V_\phi \\
E = E_\phi \;\cup \Psi^{ij}
\end{gathered}
$$

Proposed Approach

Graph Traversal: Keys

* For a vertex v_i the edge e_i* to be taken can be obtained from

$$e_i = \{(v_i, v_j) \mid (v_i, v_j) \in E\}$$
$$e_i^* = \underset{e \in e_i}{\operatorname{argmin}}\ k(e)$$

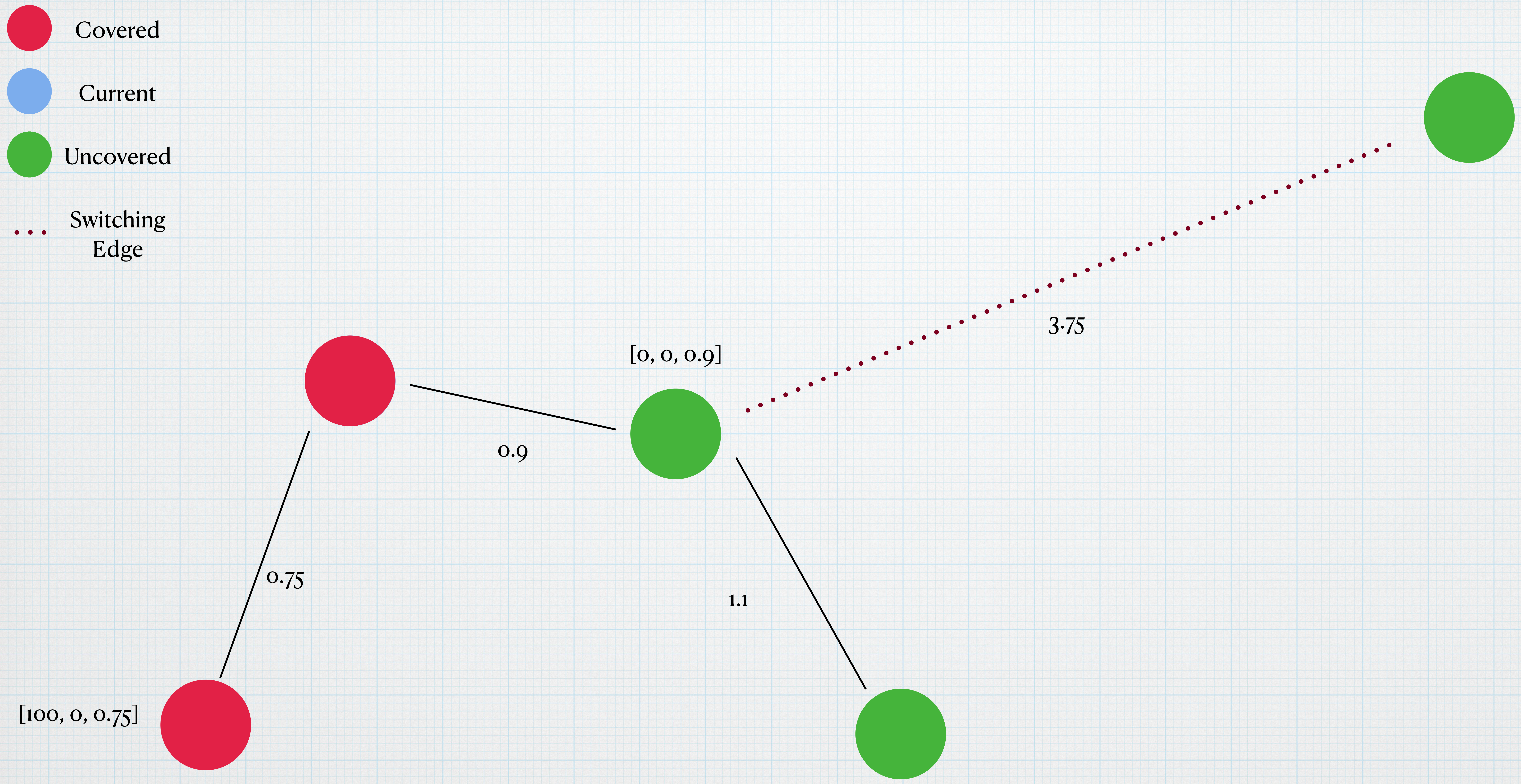
* Where, the function $k(e)$ is defined as a priority key with components:

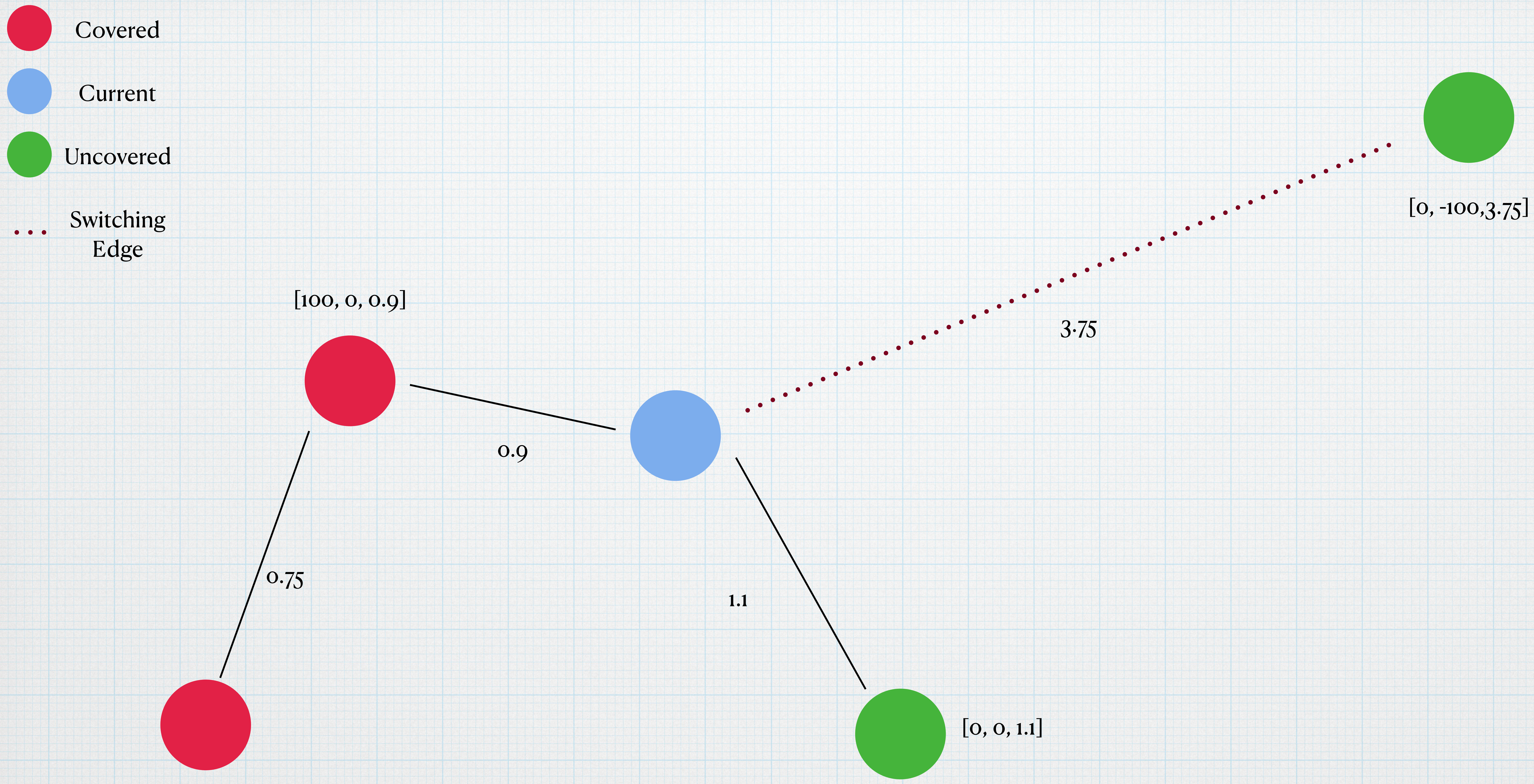
  1. Traversal cost
  2. Switching cost
  3. Distance cost

$$k(e) = [\ c_{traversed}, c_{switching}, ||e||\ ]$$

* Finally, the path can be defined as

$$\sigma^* = (e_1^*, e_2^*, \dots e_n^*)\ \forall e_i^* \in E$$


Covered
Current
Uncovered
Switching Edge
[0, 0, 0.9]
3.75
0.9
0.75
1.1
[100, 0, 0.75]


Covered
Current
Uncovered
Switching Edge
[100, 0, 0.9]
0.9
0.75
1.1
3.75
[0, -100,3.75]
[0, 0, 1.1]

Results and Discussions

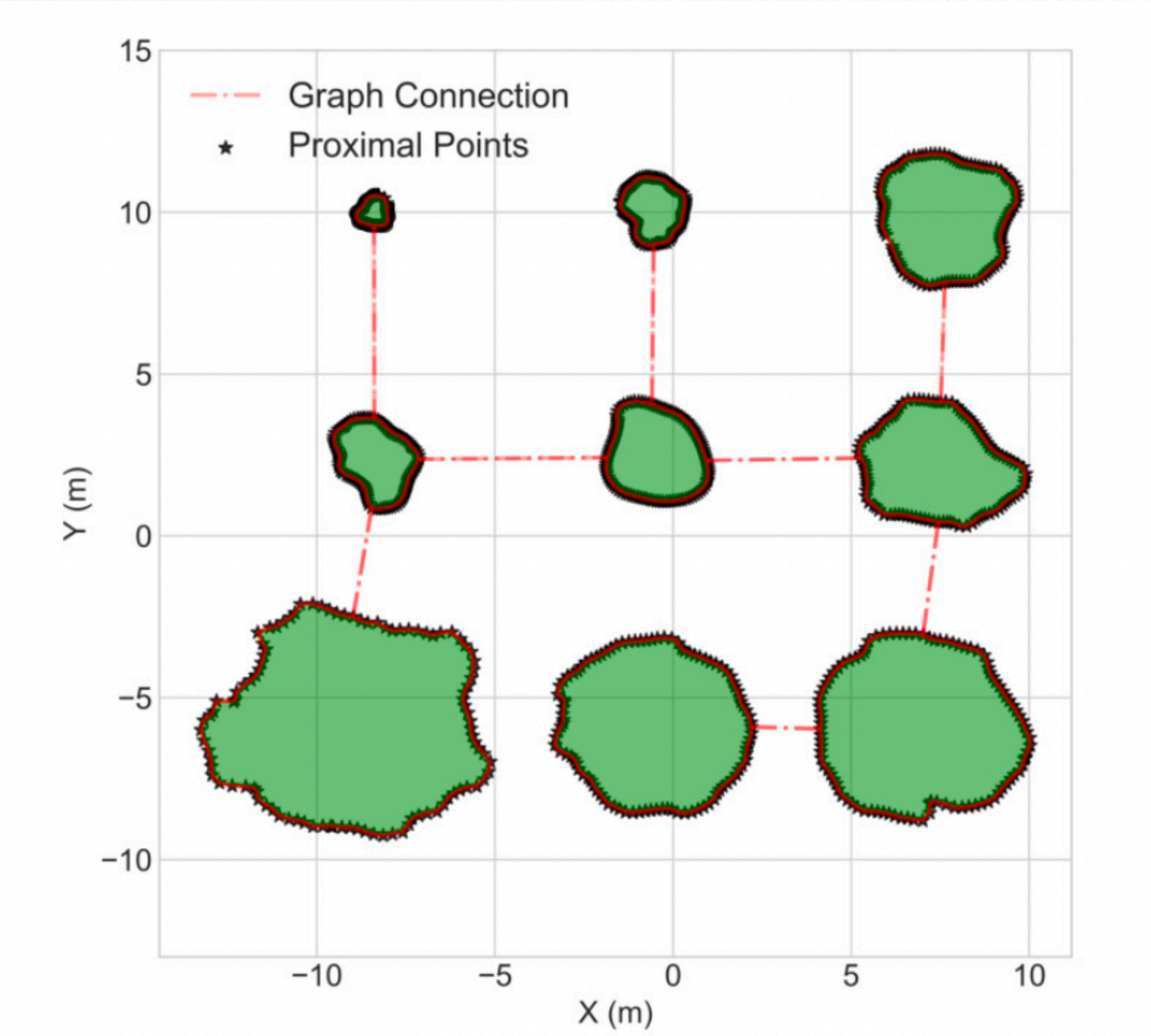


Fig 3: Complete Graph constructed

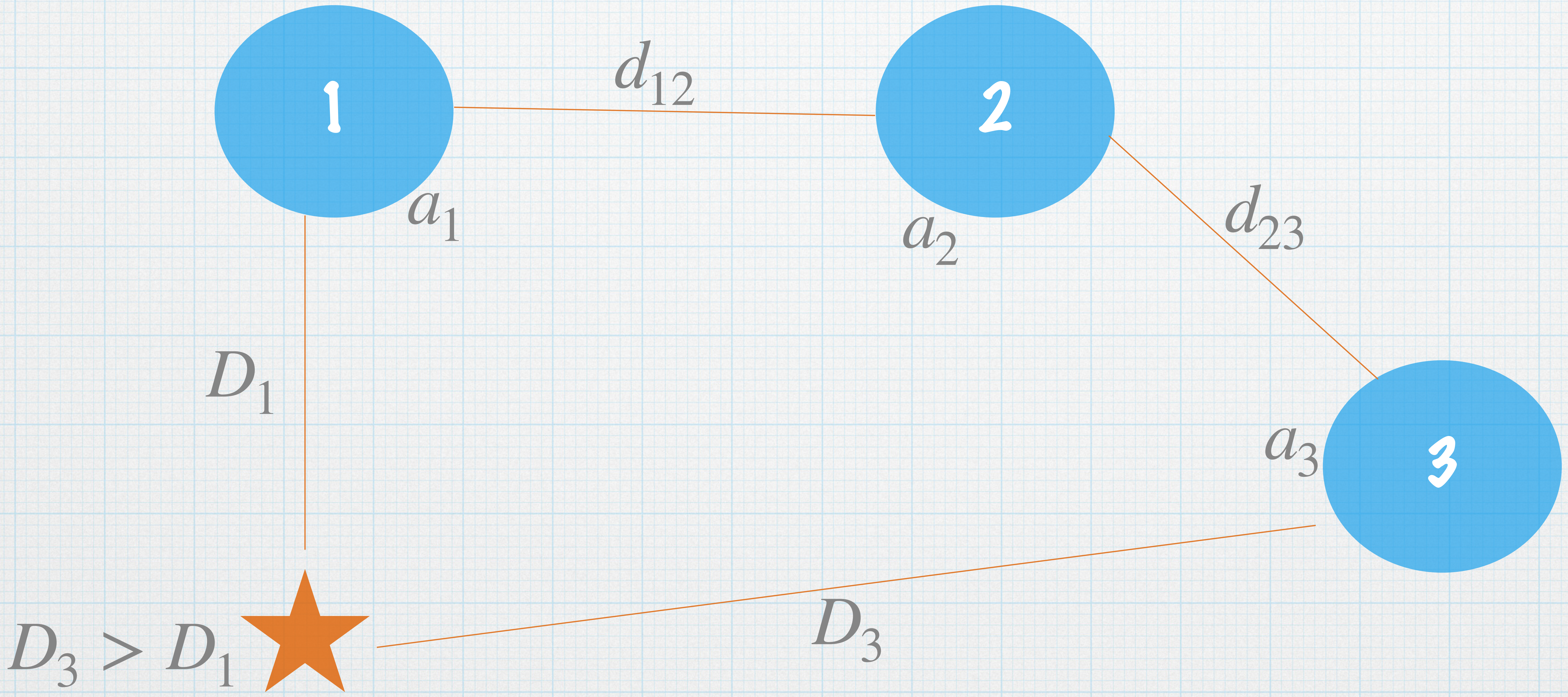
Environment	$L_{proposed}$	L_{TSP}
Grid-(2, 1)	$40.78 \pm 0.11m$	$39.14 \pm 0.01m$
Grid-(4, 1)	$83.10 \pm 1.12m$	$76.81 \pm 0.02m$
Grid-(5, 3)	$357.57 \pm 9.25m$	$290.73 \pm 0.42m$
Grid-(8, 8)	$1321.01 \pm 6.14m$	$1186 \pm 1.83m$
Environment	$t_{proposed}$	t_{TSP}
Grid-(2, 1)	$0.01 \pm 3 \times 10^{-5}s$	$0.03 \pm 12 \times 10^{-5}s$
Grid-(4, 1)	$0.04 \pm 17 \times 10^{-5}s$	$0.13 \pm 1.6^{-3}s$
Grid-(5, 3)	$0.57 \pm 50 \times 10^{-5}s$	$3.25 \pm 0.06s$
Grid-(8, 8)	$26.70 \pm 3.40s$	$98.64 \pm 4.58s$

Tab 1: Comparison for path length L and time take t

Is the presented solution always optimal?

NO

Analytic condition for optimality



Optimal: $d_{12} + d_{23} < a_1 + a_2 + (D_3 - D_1)$

Multi Agent Multiple Cluster Coverage

Multi-agent Multiple Cluster Coverage

- Problem Setup
- Proposed Method
- Existing Solutions
- Results and Discussions

Problem Setup

* Given K clusters, arranged in a $M \times N$ Grid, the set of Proximal points $\omega \in \Omega$, a valid path σ comprises all the waypoints traversed at least once.
* Given P - heterogenous agents with their capacity C_p, the optimal solution partitions the clusters such that the demand of each cluster is met.

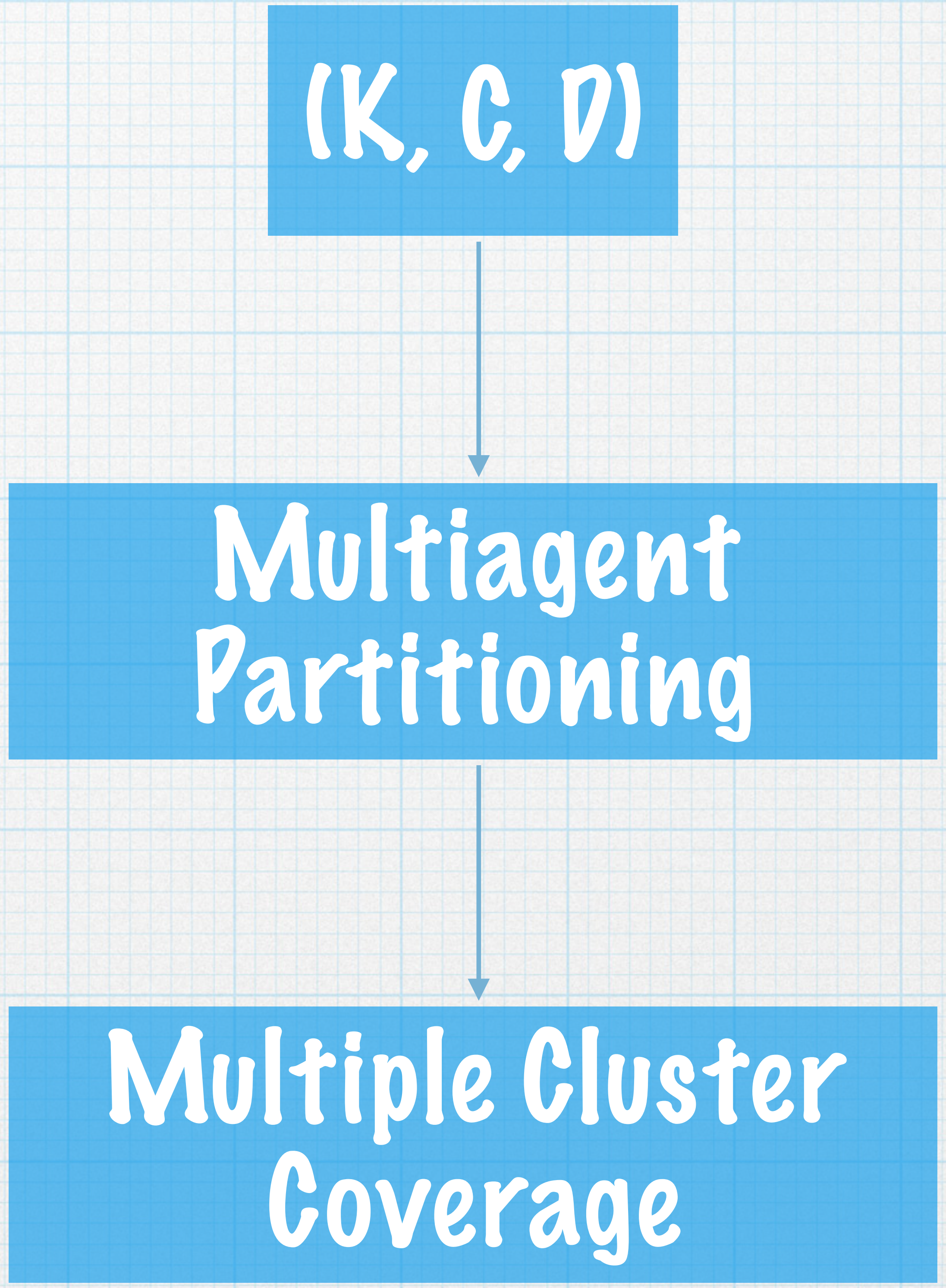
Demand of a Cluster

* It is important to define the Demand of a cluster.
* Given a cluster radius r and a height H, we can formulate two kinds of demands.

1. Volumetric Demand: $\pi R^2 H$
2. Time Demand: $\dfrac{2\pi RH}{v_p}$

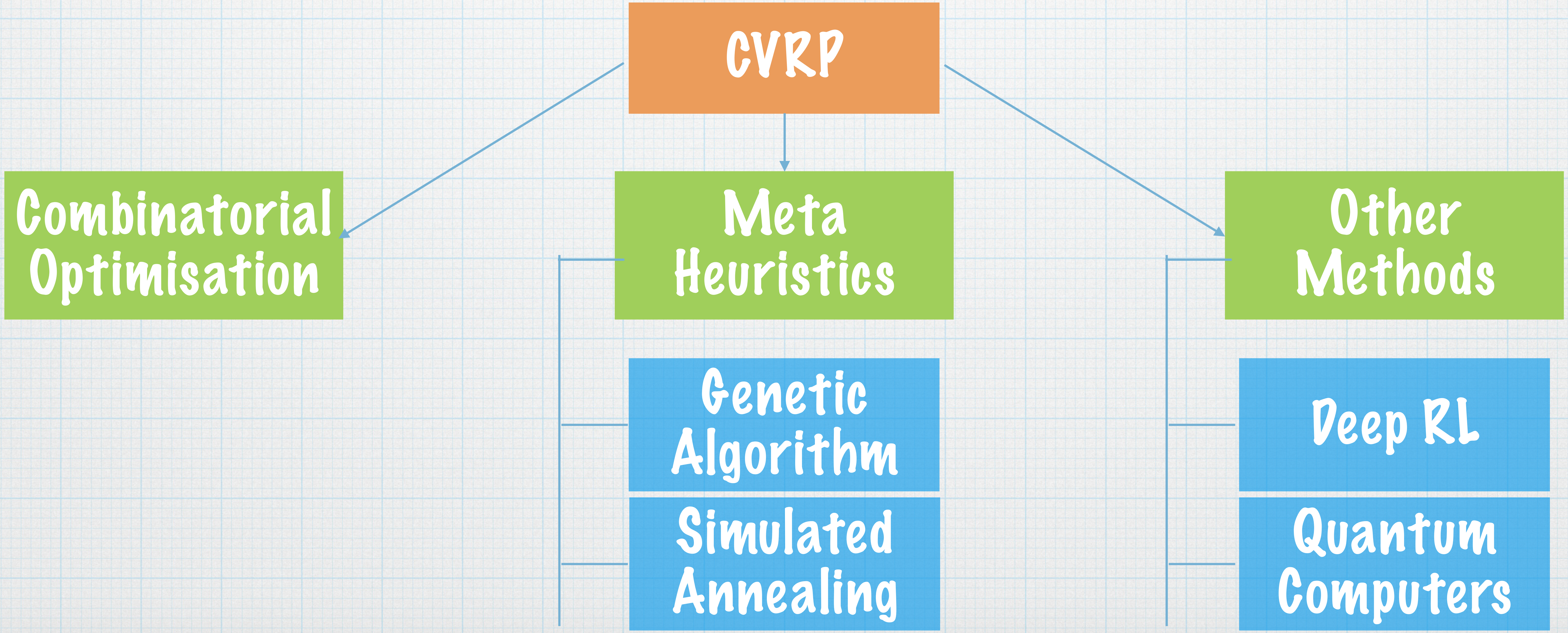
Proposed Method

* Given the clusters K
* Given a cluster radius r and a height H, we can formulate two kinds of demands.
  1. Volumetric Demand: $\pi R^2 H$
  2. Time Demand: $\frac{2\pi RH}{v_p}$



Existing Solutions

* Multi agent partitioning problem is analogous to CVRP problem. There has been extensive research going on to solve the CVRP



Method of Application

* We convert the K-cluster environment to a Topological Graph.
* A Topological Graph $G_\tau(V_\tau, E_\tau)$ contains the cluster centers as vertices with some demand D_v. The edges connect alternate clusters. The agents each have capacity C_p.

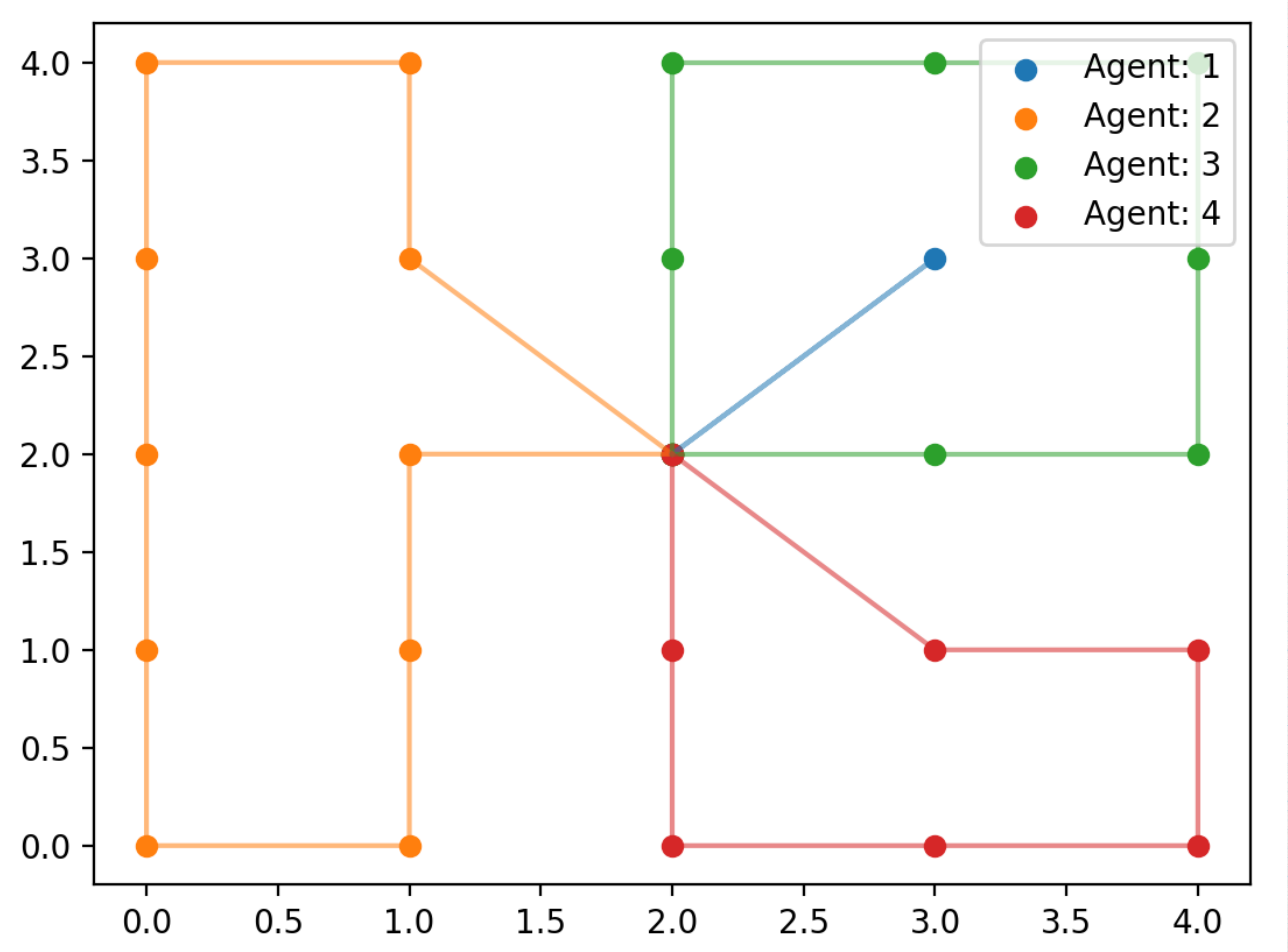


Fig 4: Graph Partitioning using Combinatorial Optimisation for CVRP

Existing Solutions

Combinatorial Optimisation: Voronoi Partitioning

Algorithm 1 Voronoi Partitioning of Graph (Modified)

$G = (V, E)$
$P = \{(p_i, L_i)\ \forall i \in R\}$ ▷ All robot positions and capacities
$g_i \leftarrow \emptyset\ \forall r_i \in R$
for $v \in V$ **do**
  if $g_i = \emptyset$ **then**
    $c_i, r_i = Dijkstra(p_i, v)$
  else
    $c_i, r_i = Dijkstra(g_i[-1], v)$ ▷ Compute path from latest location
  end if
  while $\{c_i, r_i\} \neq \emptyset$ **do**
    $c_i^* = \arg\min(\{c_i, r_i\})$
    if $c_i^* < L_i$ **then**
      $g_i \cup \{v\}$
      $L_i \leftarrow L_i - c_i$
      break
    else
      $Pop(c_i, r_i)$ ▷ Remove the cost and robot tuple
    end if
  end while
end for

Alg 1: Voronoi Partitioning Algorithm

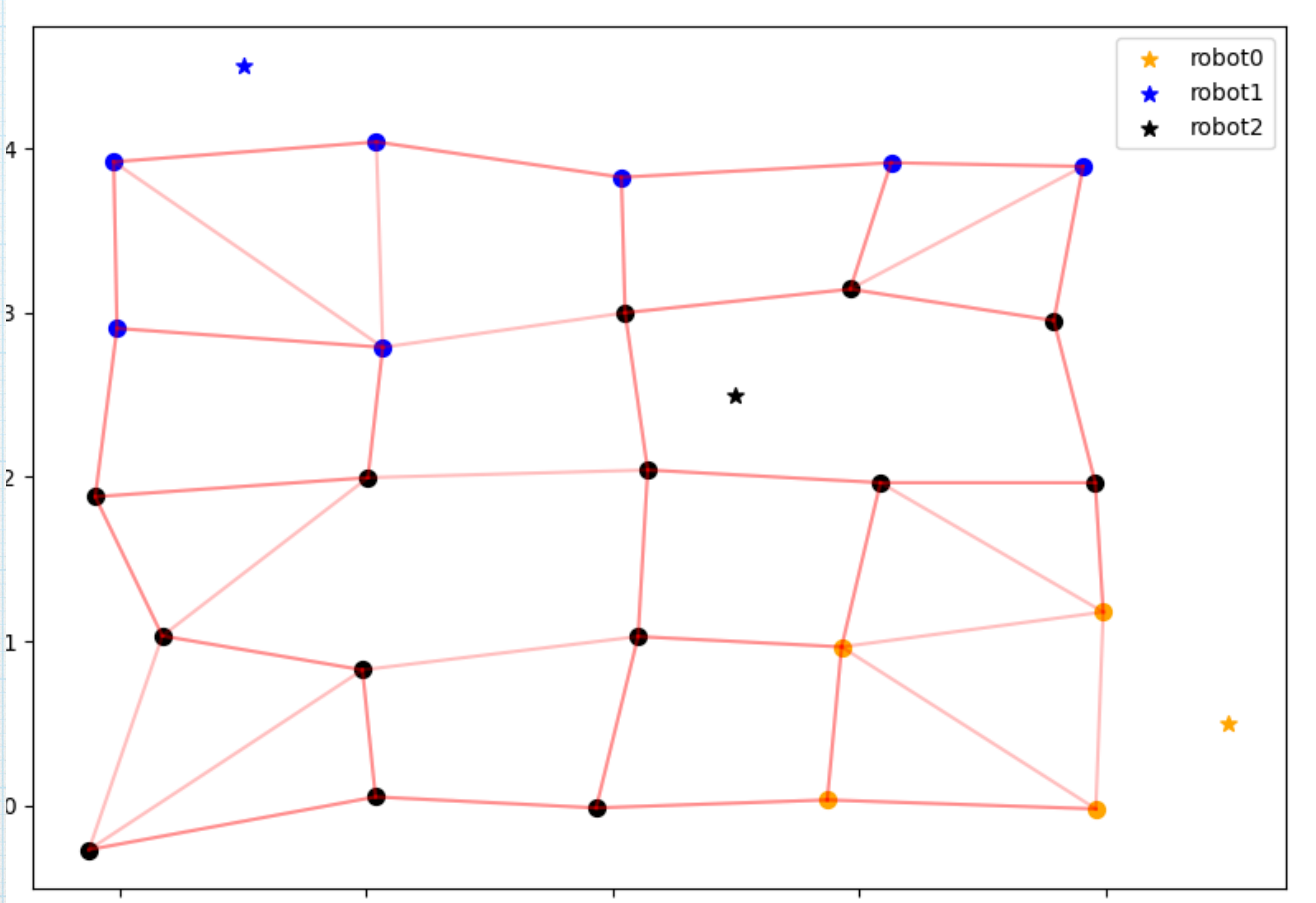


Fig 5: Graph Partitioning using using Voronoi Partitioning

Results and Discussions

Algorithm	$T(s)$	Success rate (%)
CO - ORTools	$0.234 \pm 0.011s$	99.78 ± 0.02
CO - Voronoi	$0.531 \pm 0.12s$	76.81 ± 5.12
GA	$4.573 \pm 1.25s$	40.73 ± 9.87

* The Combinatorial Optimisation method from Google OR Tools is the fastest and is the most reliable. This is also corroborated across literature.

* The success rate of Genetic Algorithm heavily depends on parameters and the choice of Crossover and Mutation functions.

Hardware Experiments

- Hardware setup
- Local Planning using Error Prediction
- Local Planning using Graph based Planner

Hardware Setup

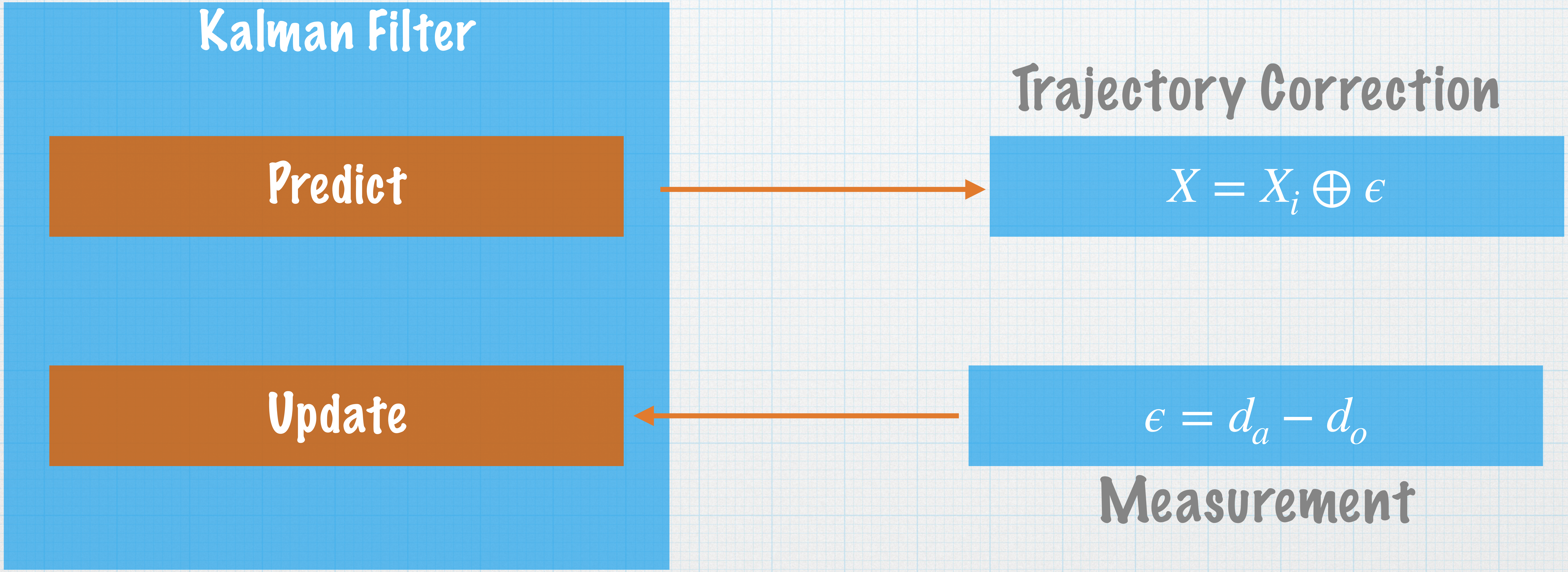
* DJI Matrice M600 Pro as testbed.
* Equipped with d435i RGB-D camera.
* The interface is written using DJI ROS SDK

Fig 6: DJI Matrice M600 Pro

Local Planning using Error prediction

* Sometimes the UAV goes really close to the canopy. This is dangerous without any onboard correction.
* At every time step, we measure Depth error and predict the Depth error for the next time step and correct the control command accordingly

Local Planning using Error prediction



Local Planning using Error prediction

* There were some issues due to the use of raw depth image.
* Use averaging kernel as a solution

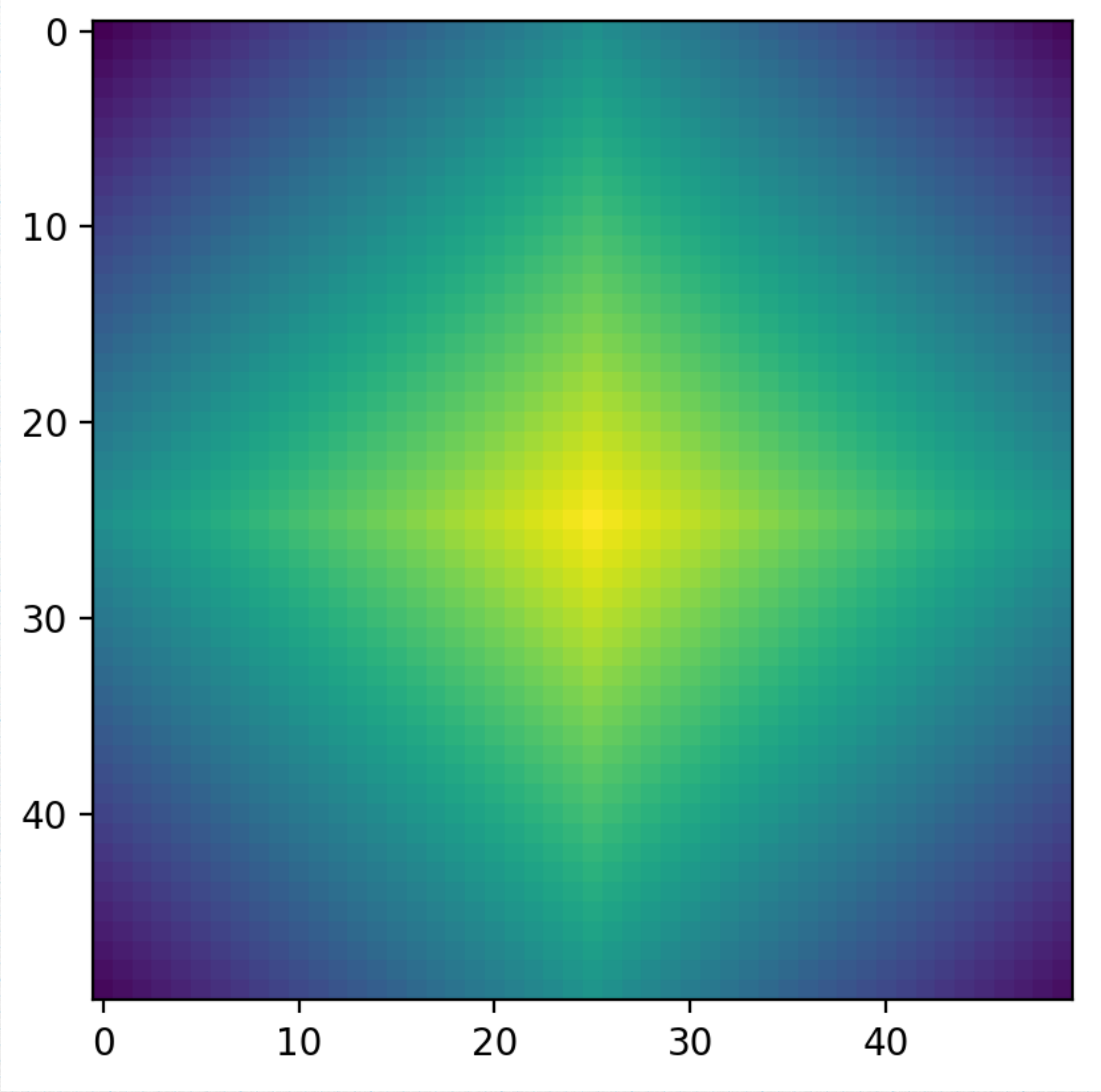


Fig 7: Averaging Kernel

Local Planning using Graph Based Planner

* The above approach is reactive in nature. This may lead to unpredictable behaviour.
* Hence, we try to use a Graph based local planner.
* We perform Scan registration and then use the SDF generated to sample points and perform Graph Search.

Local Planning using Graph Based Planner

Scan Registration

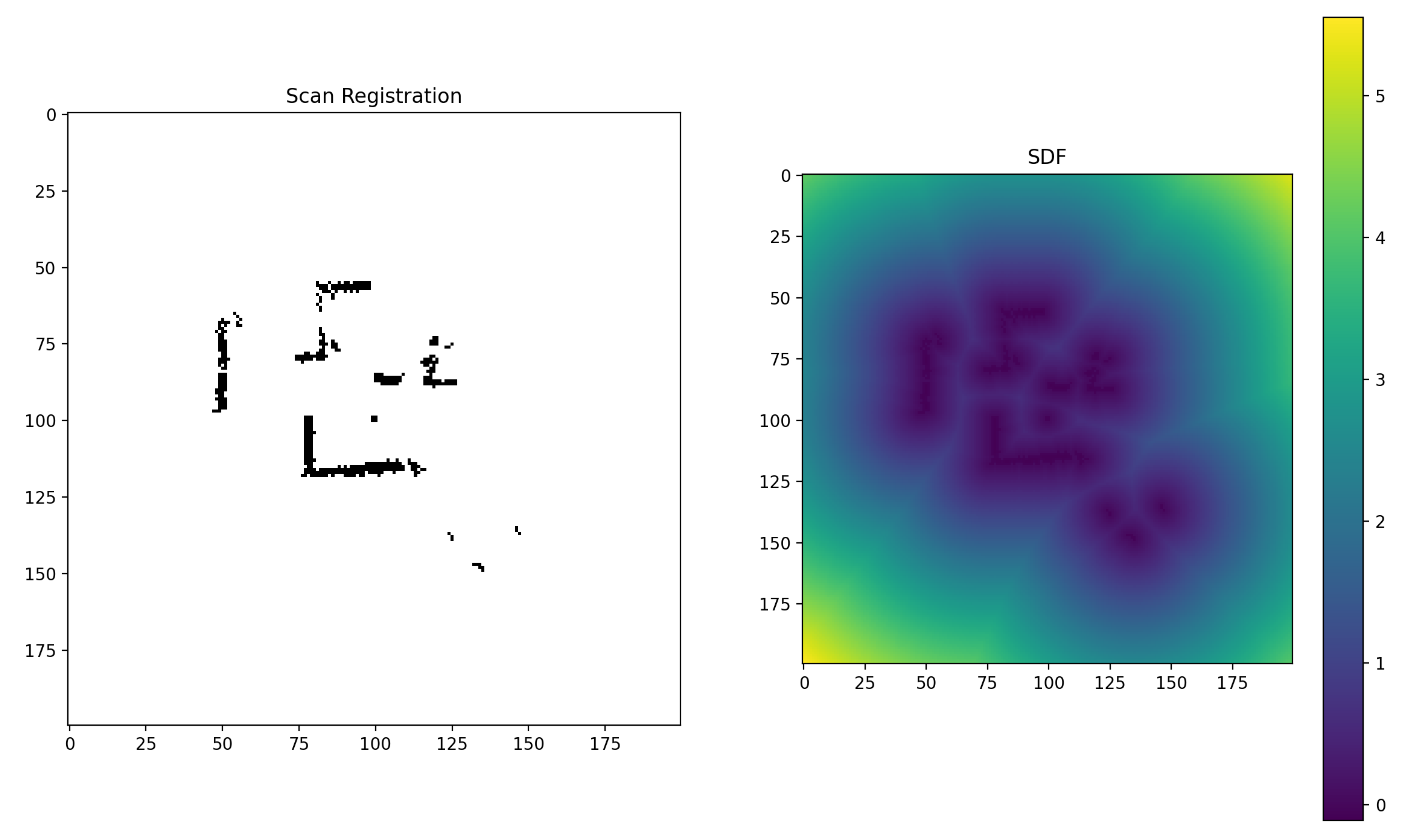


Fig 8: Scan Registration and SDF

Local Planning using Graph Based Planner

Ellipsoid Rejection Sampling

* Given the Start and Goal locations, we randomly sample states in the ellipse joining the start and goal.

* We then reject the samples that are too close to the obstacles using the SDF.

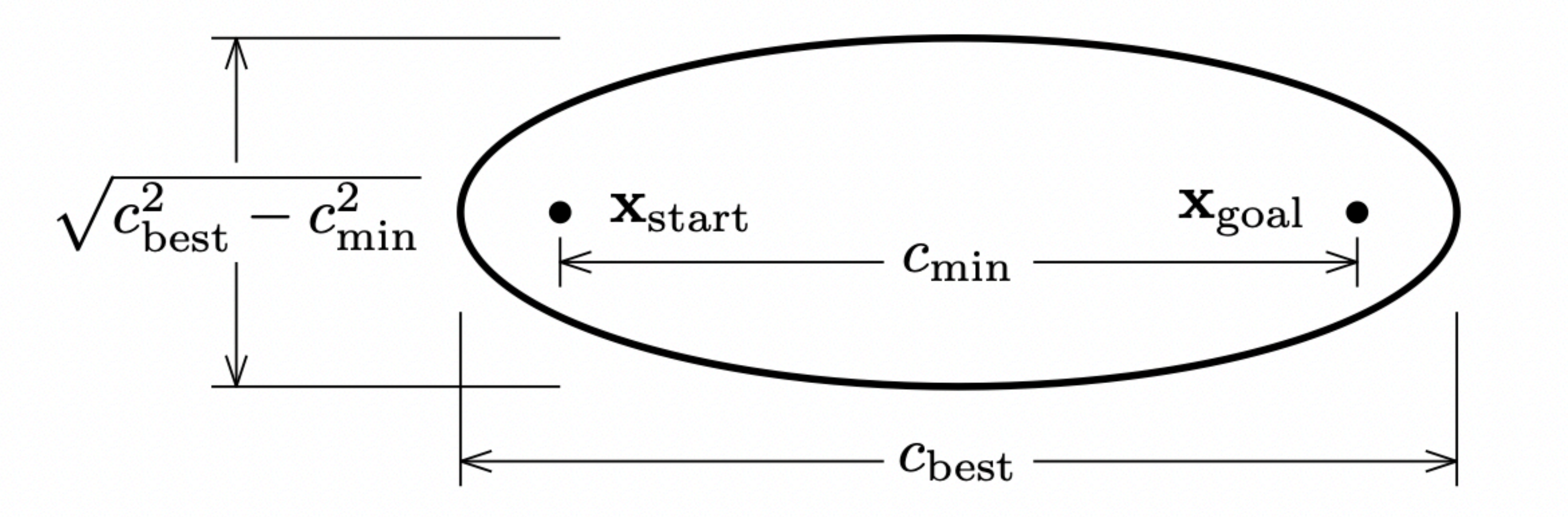


Fig 9: Randomly sample states within an ellipse whose foci are Start and Goal and the eccentricity of the ellipse being $\frac{c_{min}}{c_{best}}$. Here c_{best} can be considered to be $\alpha \cdot c_{min}$ where α is a hyper parameter

Local Planning using Graph Based Planner

Search Algorithm

* Given the sampled points, a Graph is constructed with the sampled points as vertices.

* The vertices that are close enough and are away from collision or safety threshold are connected as edges.

* Then we use a Graph search algorithm (A-Star, Dijkstra) to find a path from start to goal.

* The algorithm is Asymptotically optimal. The optimal solution is found as the number of samples increases.

Local Planning using Graph Based Planner

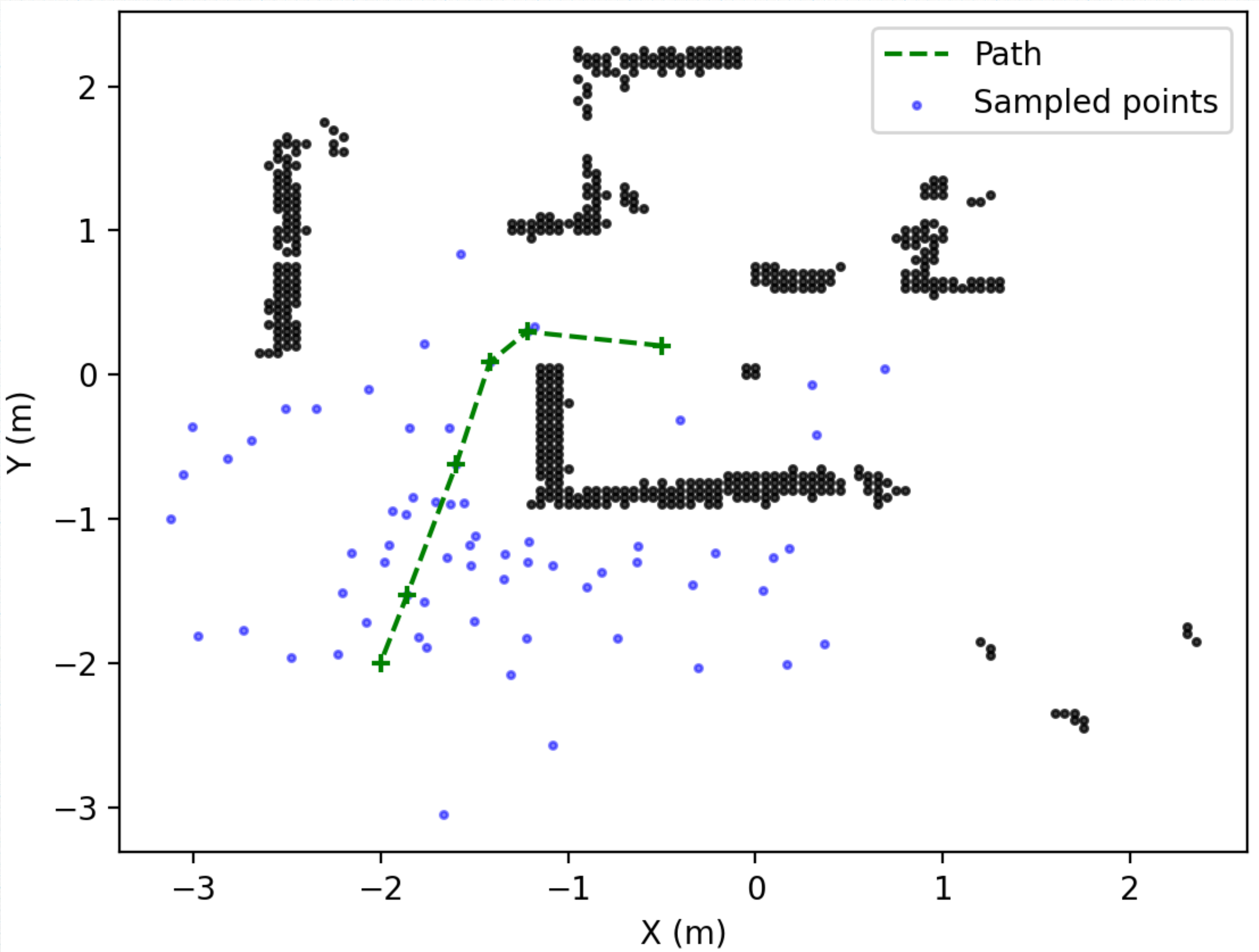


Fig 10: Graph based planning using rejection sampling

Conclusion

* We presented a near optimal solution to a NP Hard problem of Multiple cluster coverage.
* We extended the Multiple cluster coverage algorithm to use Multi agent systems.
* We developed local planners to ensure safety during traversal of global waypoints.

Thank You