# Planning and Navigation for a Game Bot

*by Giorgio Gori*

Computational Robotics [CPSC 515] Course Project
University of British Columbia
10 December 2015

# Outline

- Game, interface and environment description
- Problem statement
- State of the art
- Related work
- Proposed method (Work in progress)
- Evaluation (Work in progress)

# GUILD WARS

- Multiplayer online role-playing game
- Instanced areas
- Released in 2005

# User Interface

# User Interface

# Program Interface

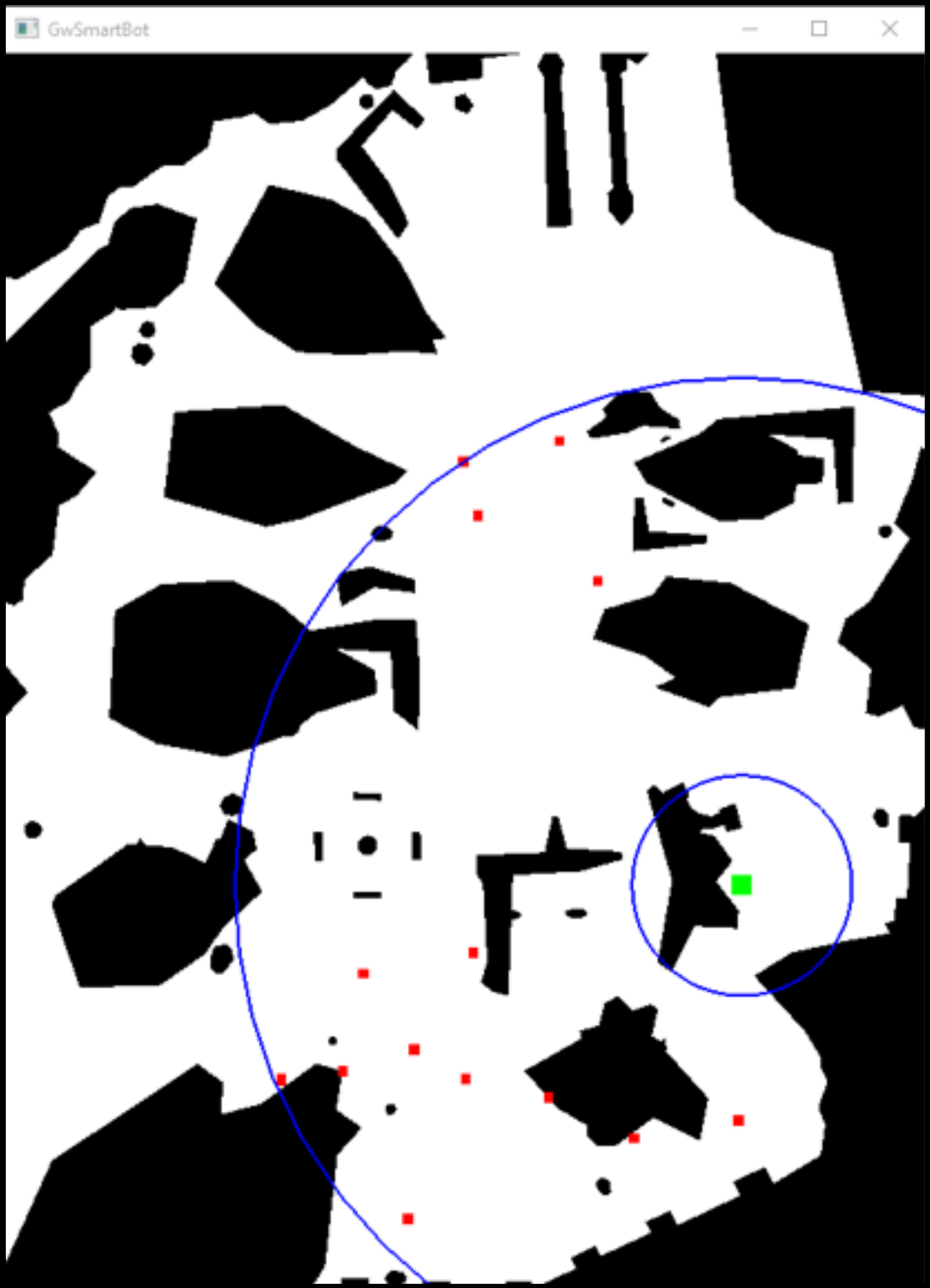
```
struct Agent {
    long id;
    float x;
    float y;
    ....
}
```

```
struct MapTrapezoid {
    long id;
    float XTL, XTR, YT;
    float XBL, XBR, YB;
}
```

```
// Sensors
Agent* GetPlayer();
vector<Agent*> GetAgents();
vector<MapTrapezoid*> GetMap();

// Actuators
void Move(float x, float y);
void UseSkill(int skill, long target_id);
```

# Vision and Behavior




- Player can see agents if inside the outer circle
- Agents can see the player if inside the inner circle
- When agents see the player, they will chase him
- Groups of 4 nearby agents share vision

# Problem Statement: Informal

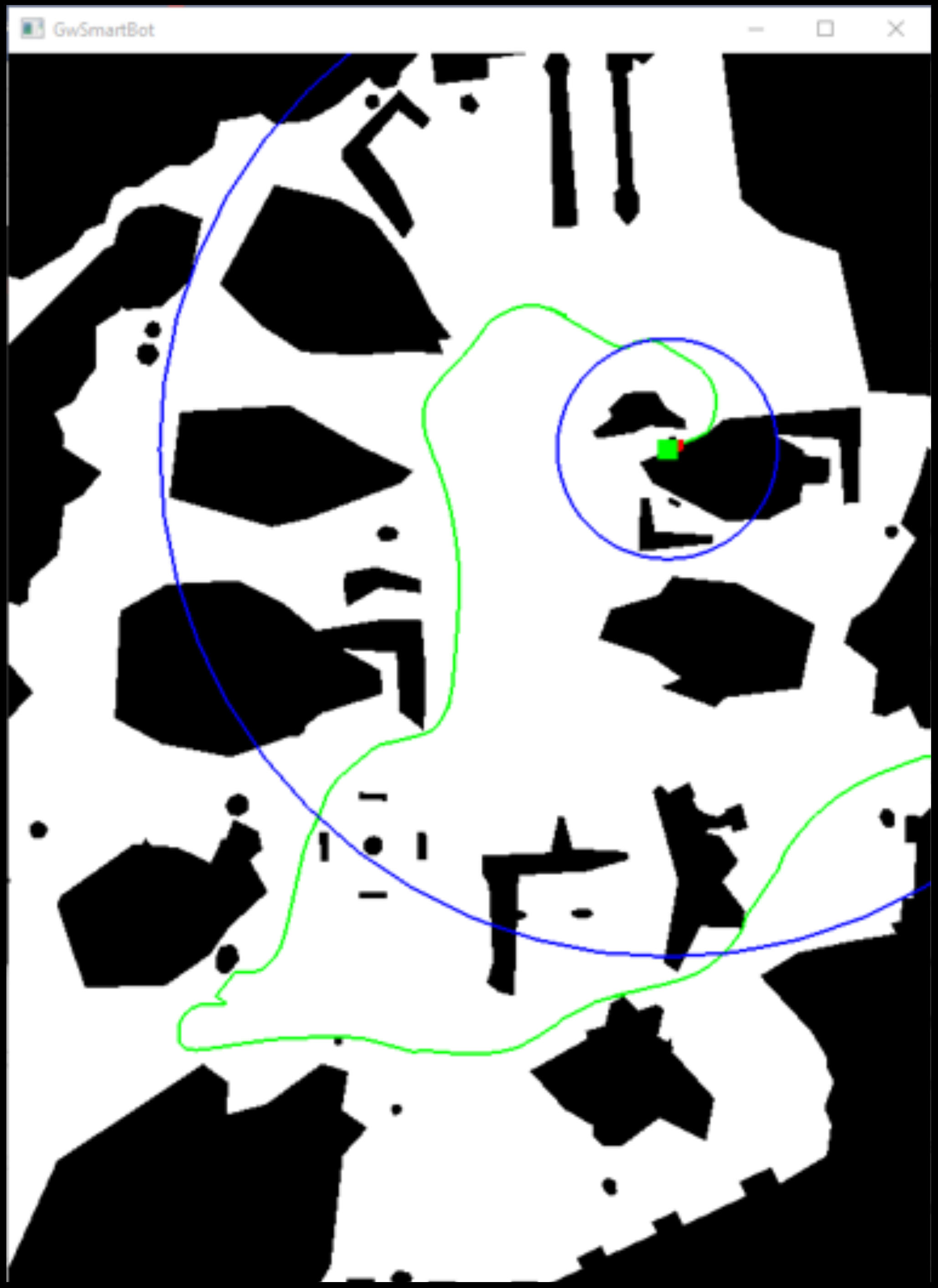
Group all agents in the area

*or*

Move in such a way that all agents can see the player

# Problem Statement: Example


# Problem Statement: Example




# Problem Statement: Formal

**Input:** Player position *{Px, Py}*, an end position *{Ex, Ey}* and a set *Agents* = $\{A_1, \ldots, A_n\}$ where each Agent $A_i = \{id_i, x_i, y_i\}$

**Output:** The shortest path $P = (p_1, p_2, \ldots, p_{m-1}, p_m)$ such that $p_1 = \{Px, Py\}$, $p_m = \{Ex, Ey\}$ and for at least one agent in every group, *distance(Agent,* $p_j$*)* $< k$ for some $p_j$ in $P$

At each time step *t* find the shortest path *P* and move towards the next step $p_2$

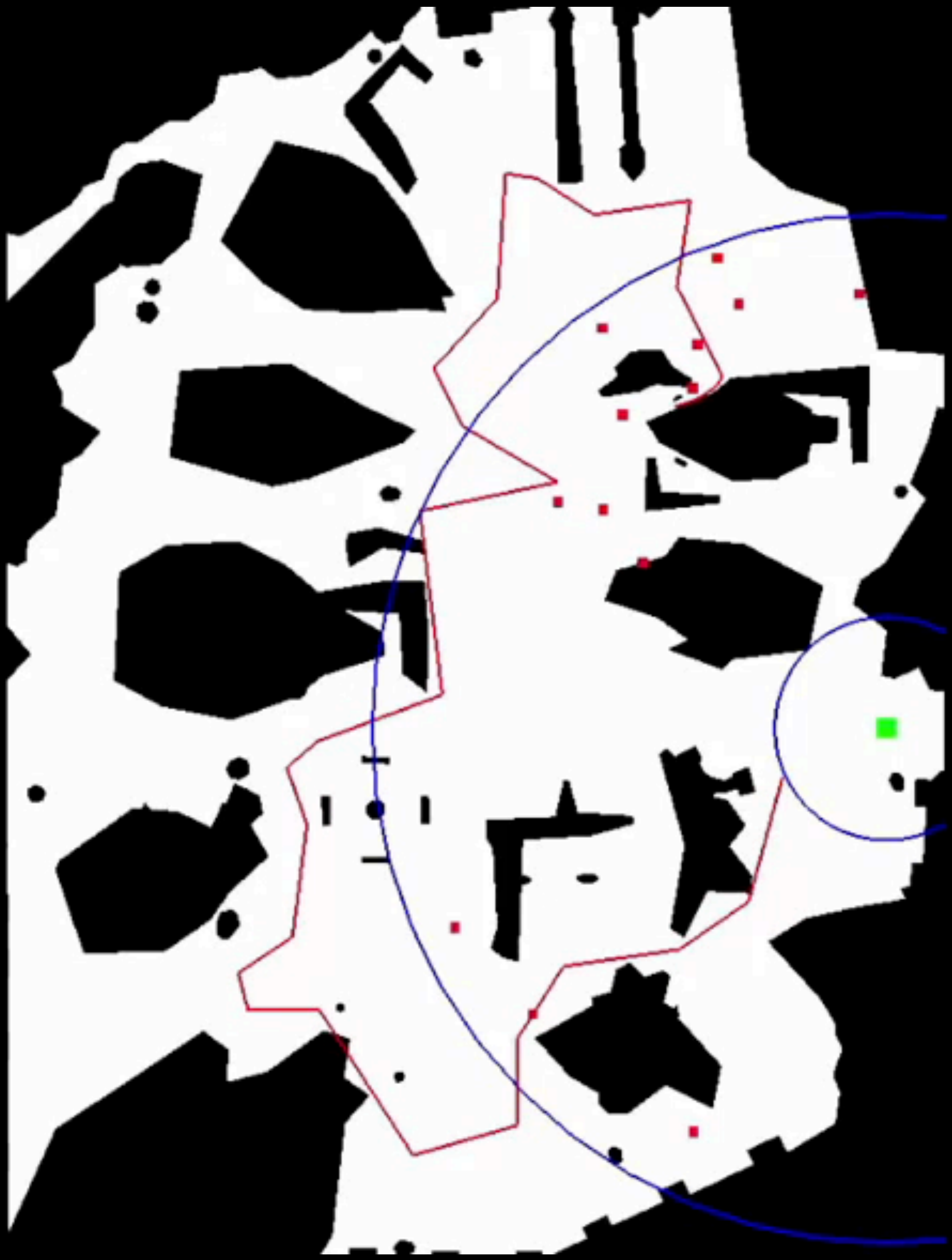
# State of the Art: Fixed Path

**Input:** Player position $\{Px, Py\}$, a path $P = (p_1, ..., p_m)$

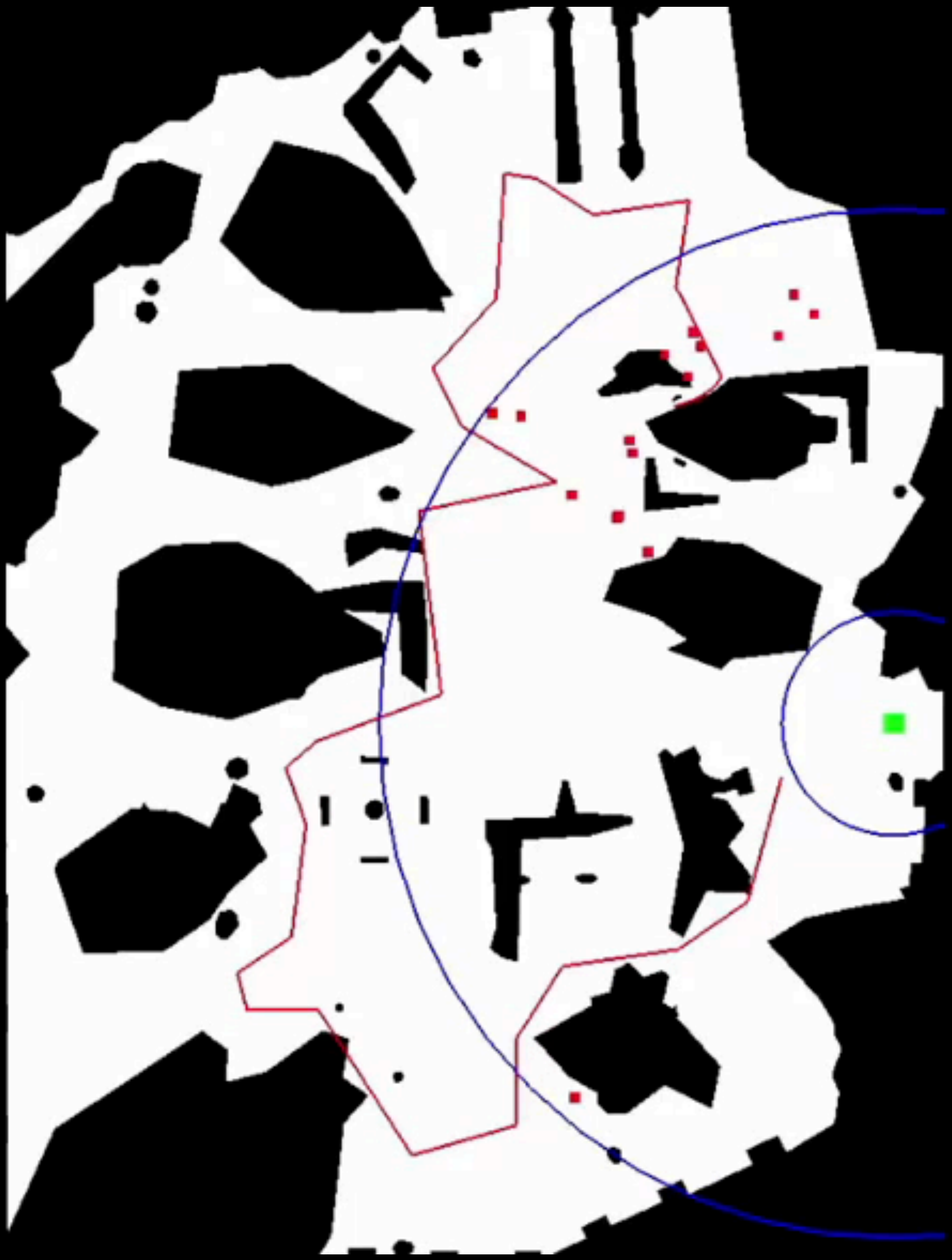
**Output:** The next destination $p_i$ in the path

At each time step $t$ move towards the next destination $p_i$

## Human



## Fixed Path



3x speed

# Related Work

- **K-Means clustering**
  James MacQueen. "Some methods for classification and analysis of multivariate observations." (1967).

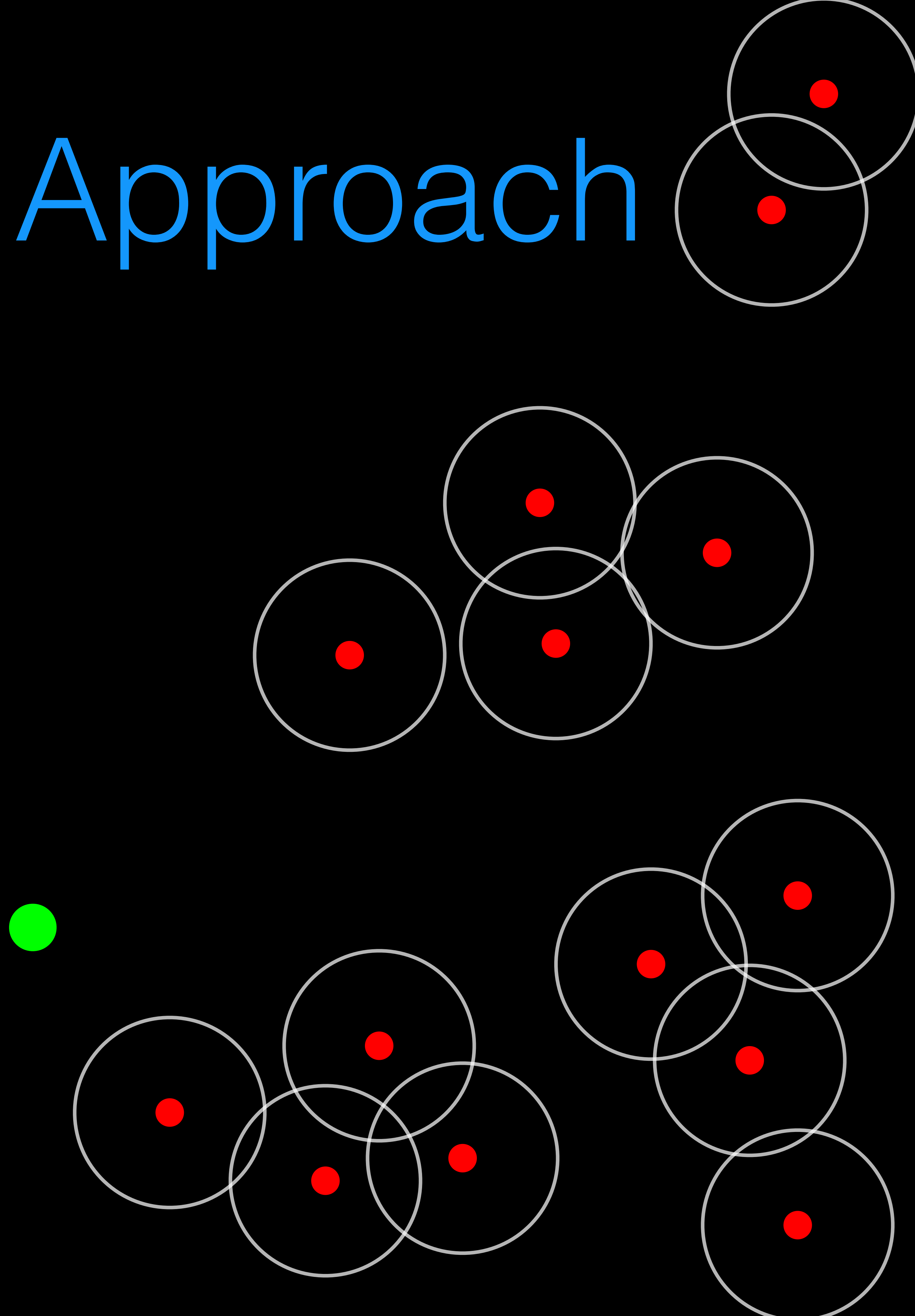
- **Travelling Salesman Problem (TSP)**
  Dorigo, Marco, and Luca Maria Gambardella. "Ant colonies for the travelling salesman problem." (1997).

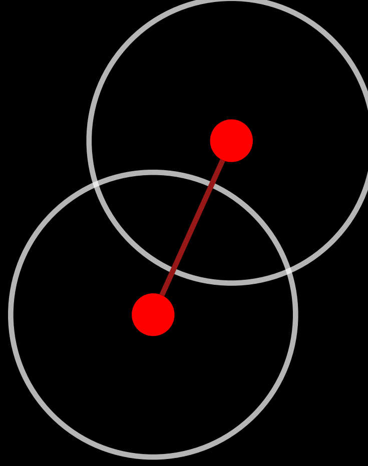
- **Generalized TSP (GTSP)**
  Laporte, Gilbert, Ardavan Asef-Vaziri, and Chelliah Sriskandarajah. "Some applications of the generalized travelling salesman problem." (1996).

- **Close-Enough TSP (CETSP)**
  Mennell, William Kenneth. "Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem." (2009).

- **Touring a sequence of circles**
  Chou, Chang-Chien. "On the Shortest Path Touring n Circles." (2012).

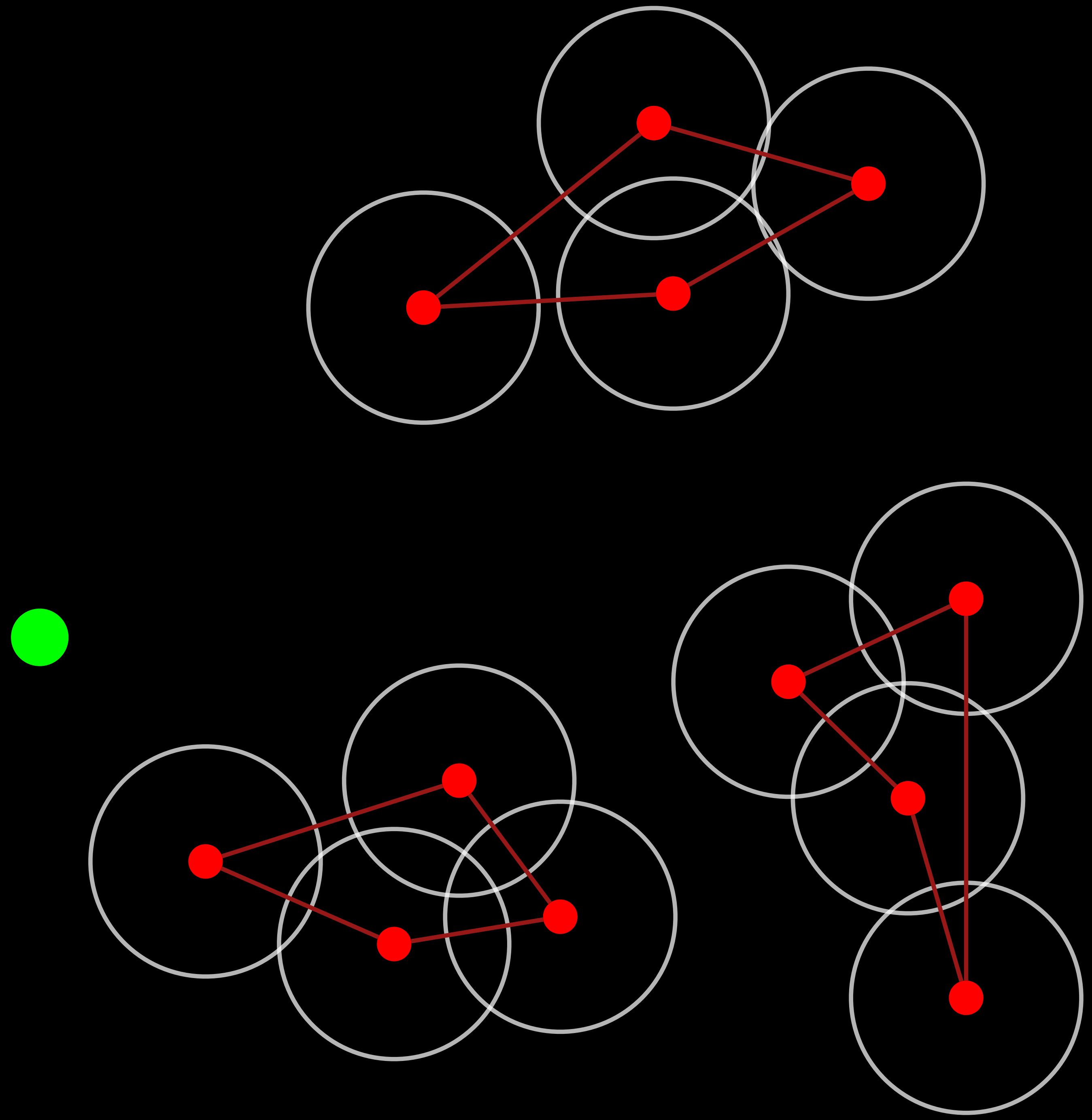
# Proposed Approach





# Proposed Approach

- Compute clusters of agents (same-sized k-means)

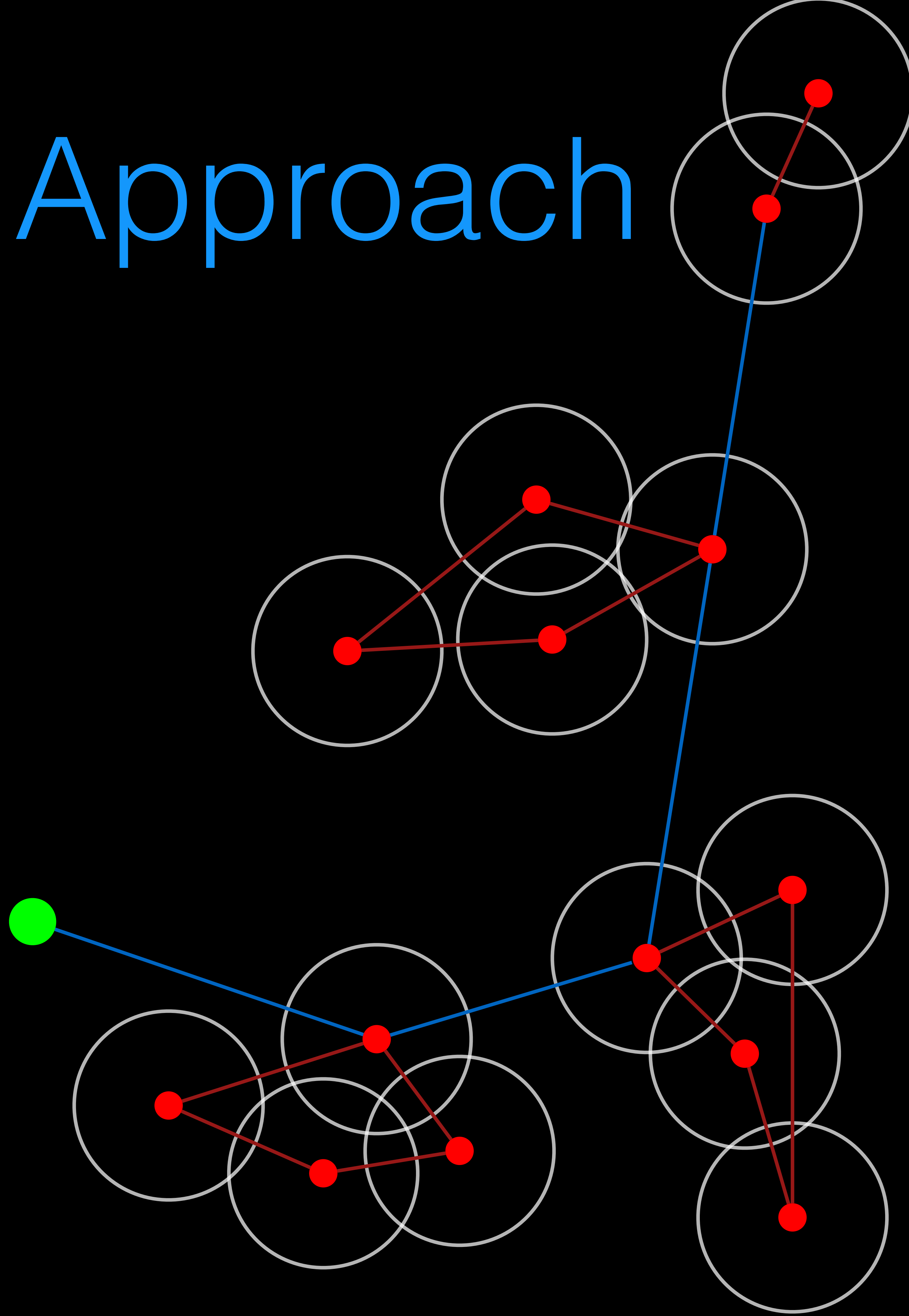


# Proposed Approach

- Compute clusters of agents (same-sized k-means)

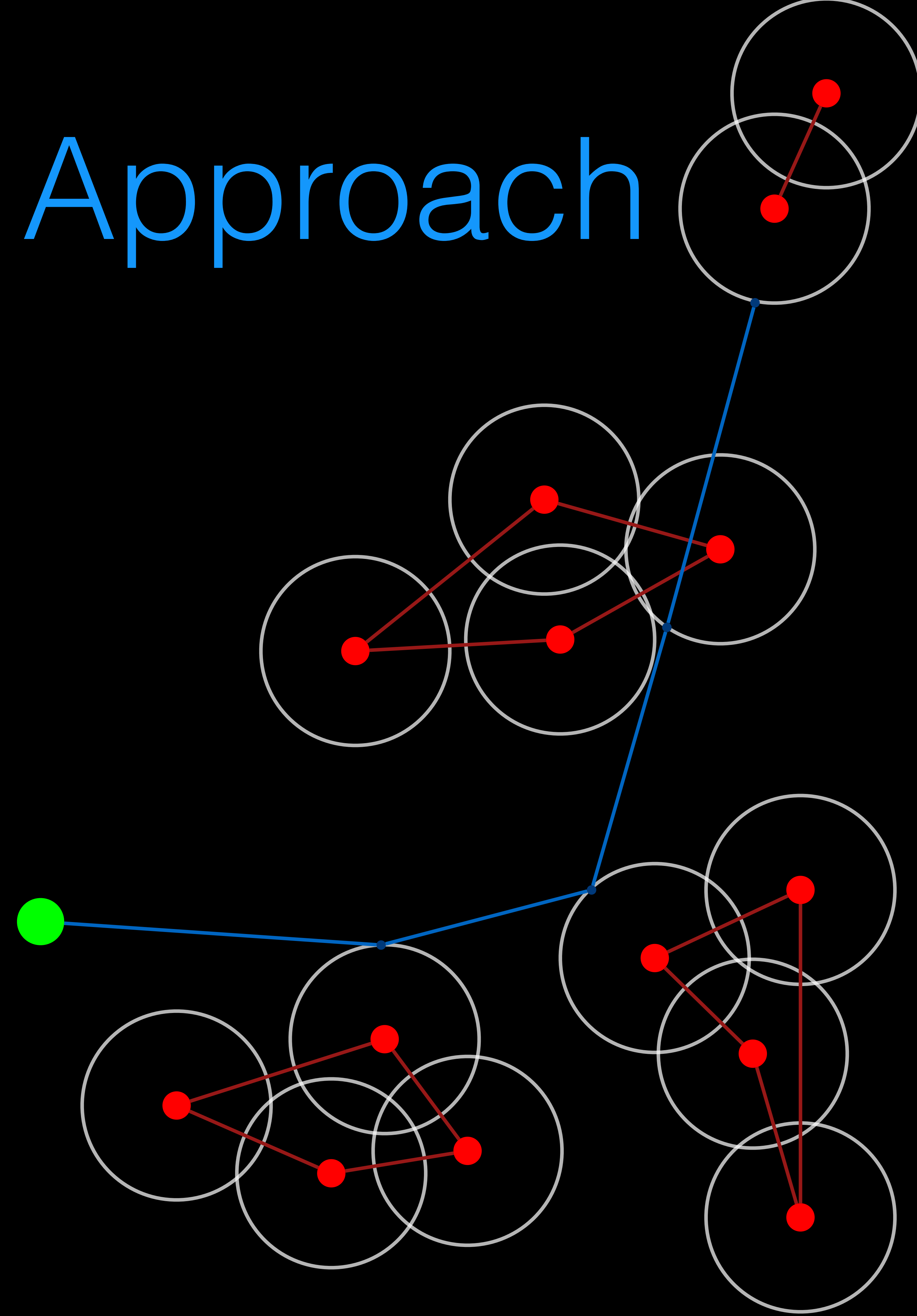
- Compute the path using GTSP on agent positions.

  Work in progress.
  Currently considering:
  - Clustered Nearest Neighbor
  - Brute Force
  - Ant Colony Optimization (ACO)



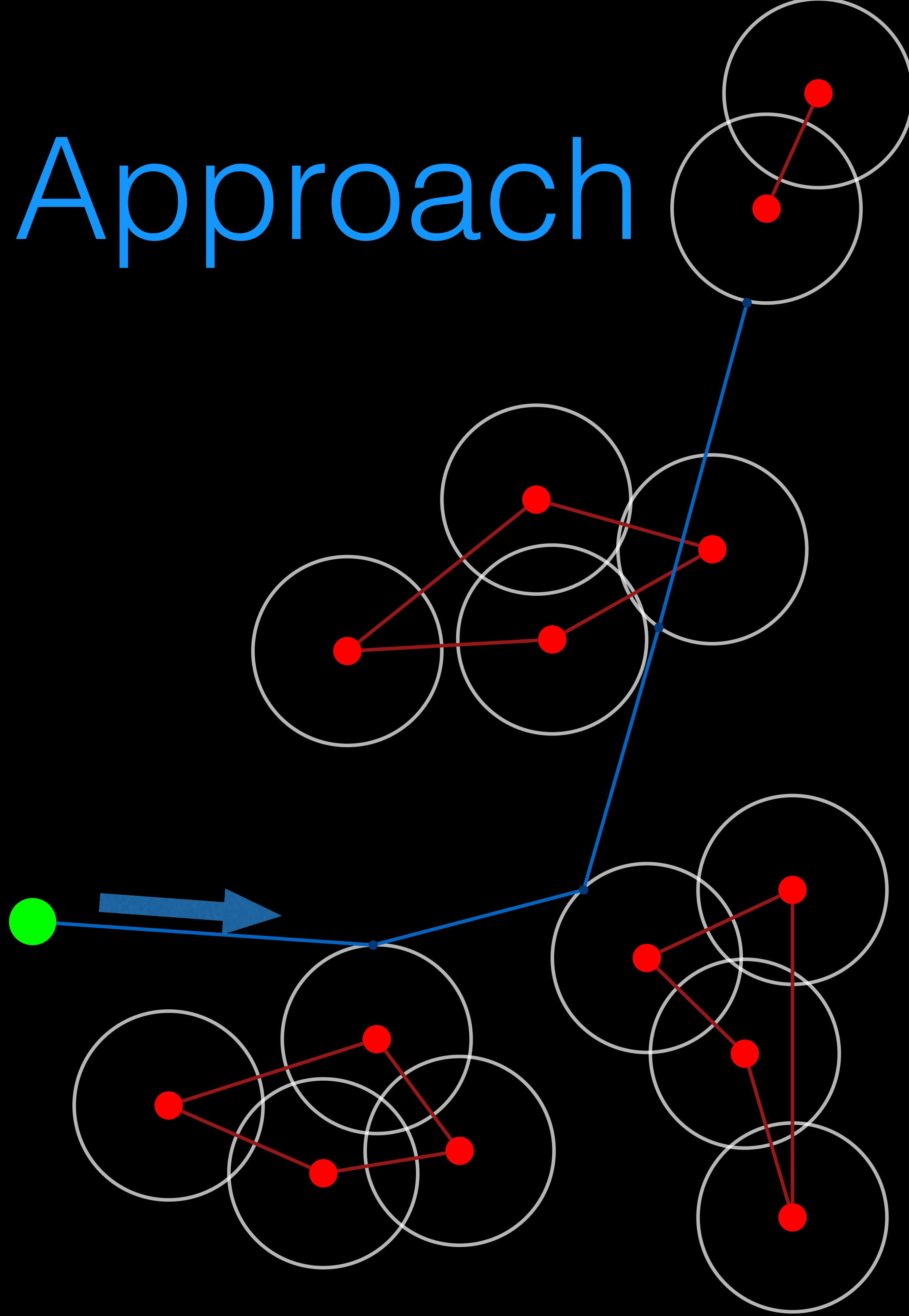
# Proposed Approach

- Compute clusters of agents (same-sized k-means)
- Compute the path using GTSP on agent positions.
- Find the optimal points in the circles (Touring Circles)



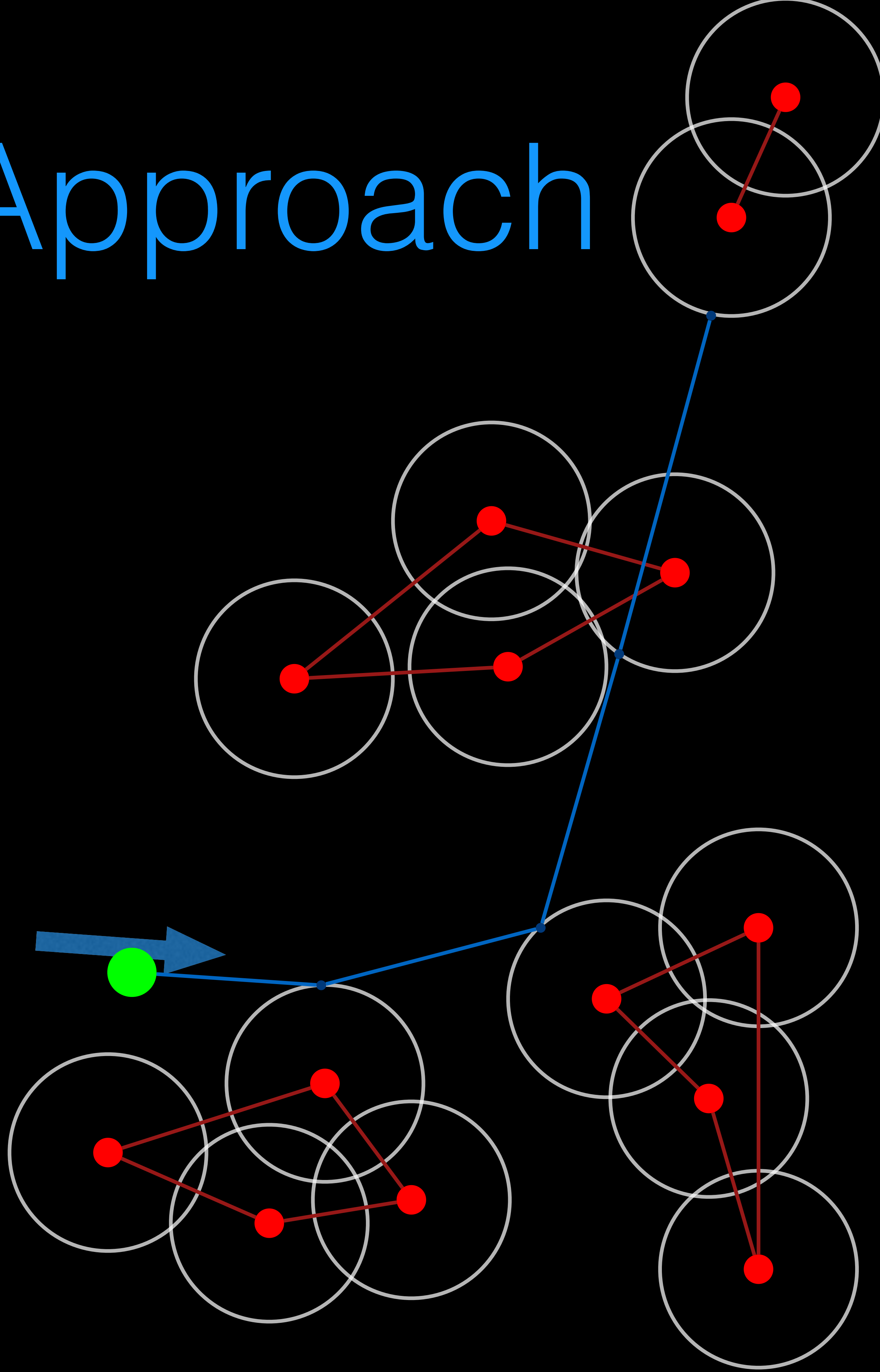
# Proposed Approach

- Compute clusters of agents (same-sized k-means)
- Compute the path using GTSP on agent positions.
- Find the optimal points in the circles (Touring Circles)
- Move to the next point



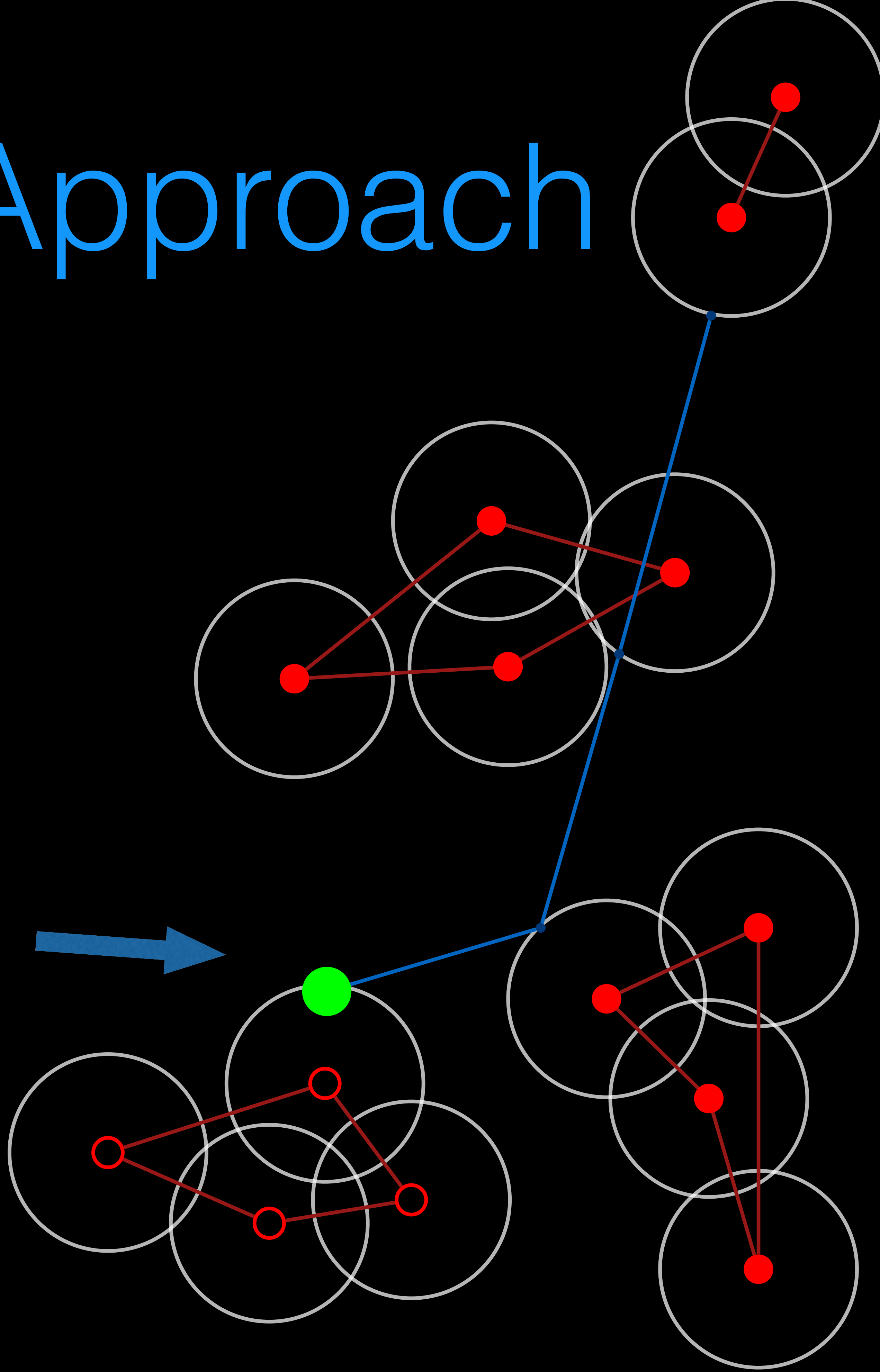
# Proposed Approach

- Compute clusters of agents (same-sized k-means)
- Compute the path using GTSP on agent positions.
- Find the optimal points in the circles (Touring Circles)
- Move to the next point
- Repeat at every iteration, using the previous clusters and path as seed.



# Proposed Approach

- Compute clusters of agents (same-sized k-means)
- Compute the path using GTSP on agent positions.
- Find the optimal points in the circles (Touring Circles)
- Move to the next point
- Repeat at every iteration, using the previous clusters and path as seed.
- After a cluster has been visited, remove it from the input set



# Evaluation

Impossible to run different methods on the same input, need to compare average across multiple trials, considering those results:

- **Speed**
  How long did it take to complete the task?
  (Average / Best / Worst)

- **Average Miss Rate**
  How many enemies did it miss in average?

- **Fail Rate**
  How many runs failed? (e.g. stuck)

# Evaluation

| | Human | Fixed path | Proposed Method |
|---|---|---|---|
| **Average Time** (min:sec) | | | |
| **Best Time** (min:sec) | | | |
| **Worst Time** (min:sec) | | | |
| **Average Missed Agents** (quantity) | | | |
| **Fail Rate** (percentage) | | | |

# Conclusion

**(Hopefully) Improvement on state of the art:**

- Up to 30% faster
- No more missing agents

**Future Work beyond CPSC 515:**

- Use map data to compute the real shortest path between agents
- Consider map and agents for obstacle avoidance / crowd navigation.