Near-Optimal Lifelong Multi-Agent Path Finding Distributed System

Dongming Shen, Ricardo Xu, Tigo Jiang

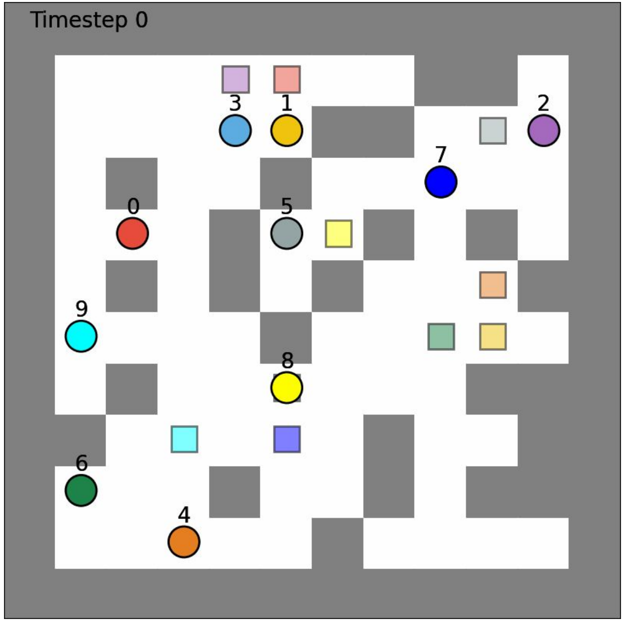
Stakeholder: Christopher Leet

Contents

- **Introduction, Impacts, and Goals**
- Problem Formulation
- Social and Ethical Impact
- System Architecture
- Implementation
- Result Demonstration

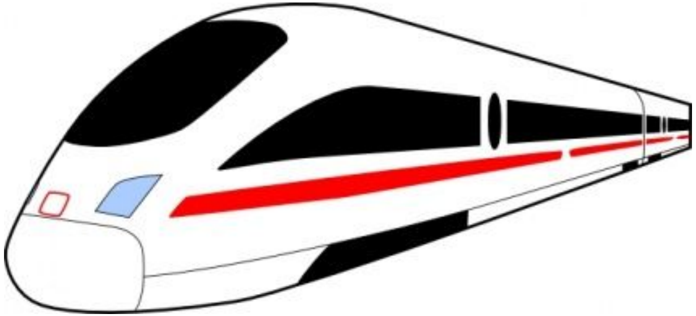
What is Multi-Agent Path Finding (MAPF)?

“Finding a set of paths which move a set of agents through a workspace to their goal location.”



Local and Global Impact of MAPF

Who solves the MAPF problem at scale?

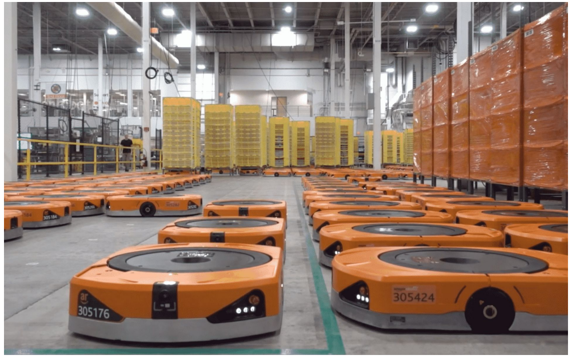


Train Timetable Scheduling

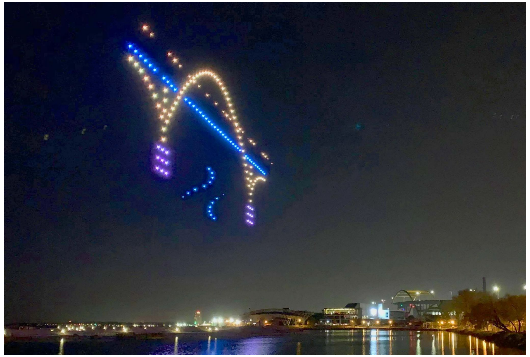
Airport Luggage Logistics

Local and Global Impact of MAPF

Who solves the MAPF problem at scale?



Automated Warehouse Logistics



Drone Swarm Operators

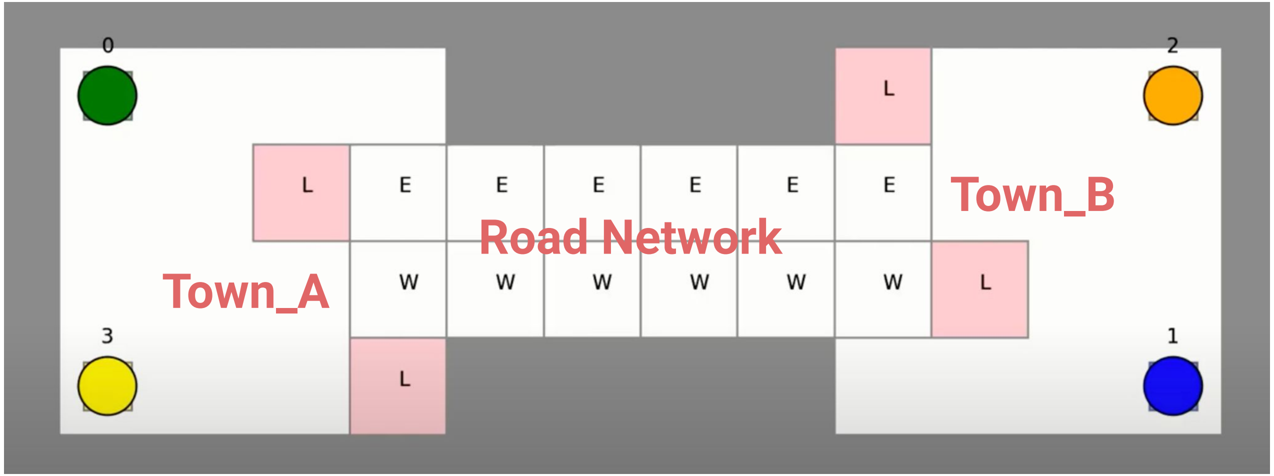
Challenge of MAPF

MAPF is NP-Hard

But **large MAPF instances must be solved in real time** by multi-robot systems!

Our Approach: Town and Road System

"Divide the workspace into a series of geographic regions called **towns** linked by a **road network**."



Our Project Goals

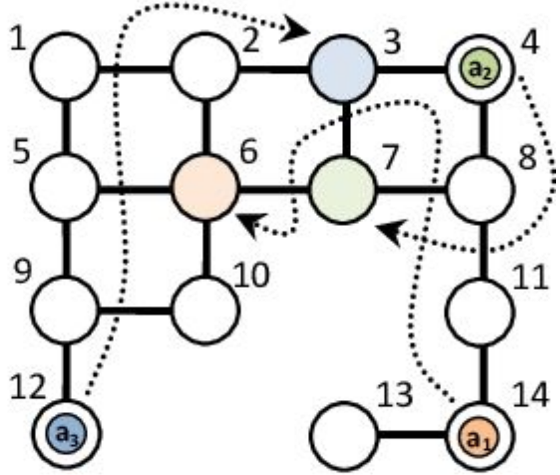
1. Design an **architecture** for a town and road system.
2. **Implement** this architecture.
3. **Evaluate** our implementation.

Contents

- Introduction, Impacts, and Goals
- **Problem Formulation**
- Social and Ethical Impact
- System Architecture
- Implementation
- Result Demonstration

Problem Formulation

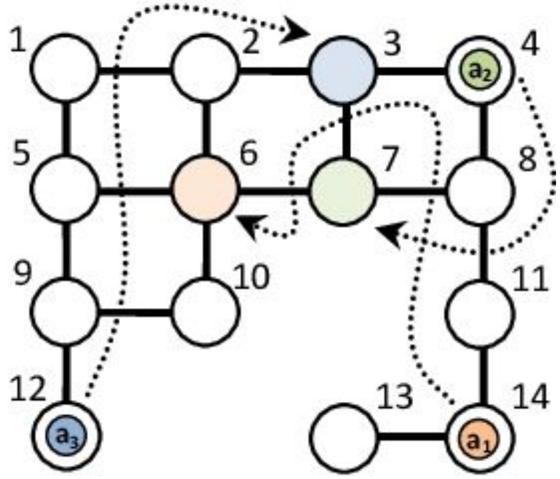
- Time is discretized.
- Each timestep, an agent can:
  - move to an adjacent vertex
  - remain at its current vertex
- **Goal.** Find a plan for each agent which moves it to its goal without colliding with another agent.



Problem Formulation

Formally, a MAPF instance consists of:

- A graph (V, E).
- A set of agents $\{a_1, a_2, \ldots\}$ positioned at the start vertices $\{s_1, s_2, \ldots\}$
- A set of goal vertices $\{g_1, g_2, \ldots\}$

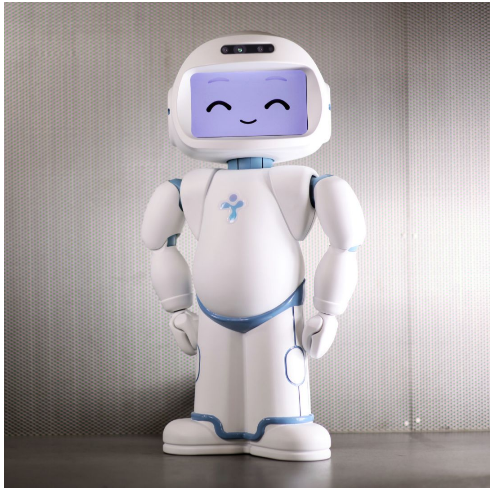


Contents

- Introduction, Impacts, and Goals
- Problem Formulation
- **Social and Ethical Impact**
- System Architecture
- Implementation
- Result Demonstration

Social and Ethical Impact of MAPF

How will robotic automation affect people's livelihoods?

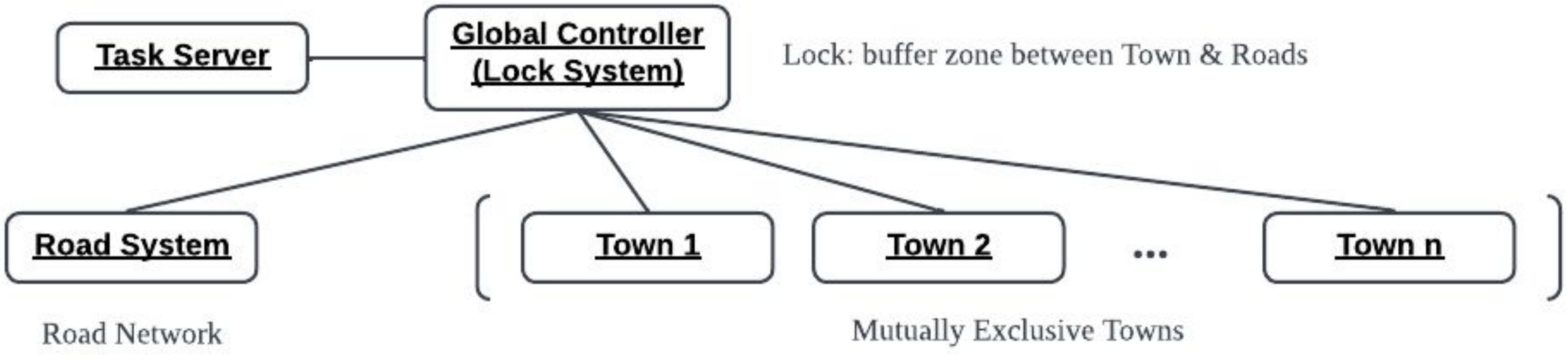


Is using AI to control large systems of robots ethical and safe?

Contents

- Introduction, Impacts, and Goals
- Problem Formulation
- Social and Ethical Impact
- **System Architecture**
- Implementation
- Result Demonstration

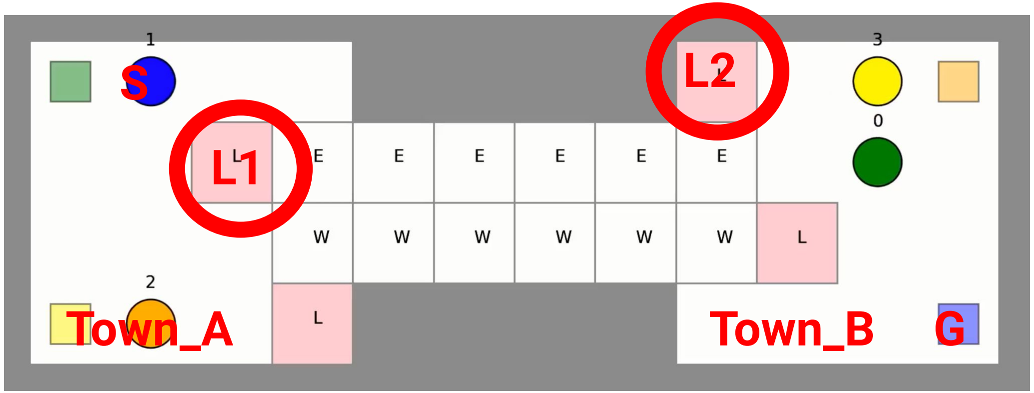
High Level System Architecture



System Architecture

Lifecycle of one robot from S (Town_A) to G (Town_B)

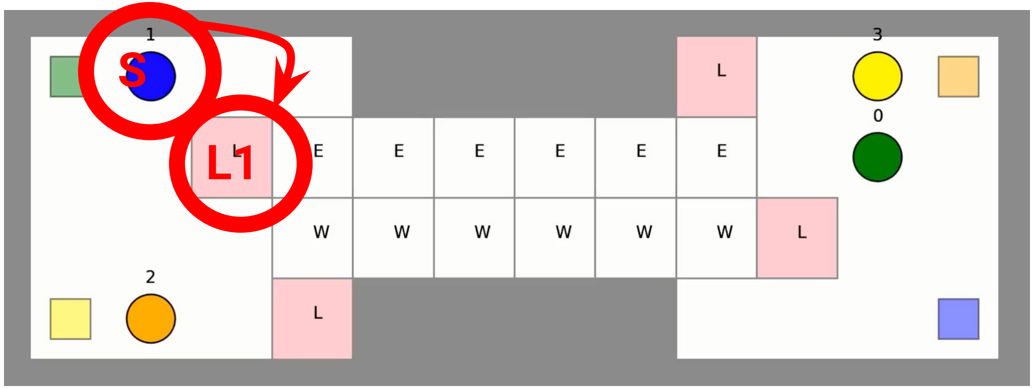
1. **START: Global assign Locks pair (L1, L2)**
2. Town_A route robot from S to L1 - Global
3. Road route robot to L1 to L2 - Global
4. Town_B route robot from L2 to G - FINISH!



System Architecture

Lifecycle of one robot from S (Town_A) to G (Town_B)

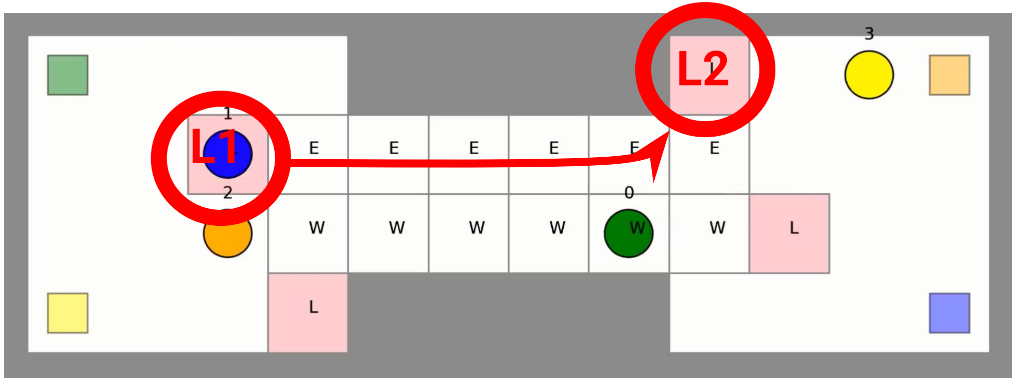
1. START: Global assign Locks pair (L1, L2)
2. **Town_A route robot from S to L1 - notify Global**
3. Road route robot to L1 to L2 - Global
4. Town_B route robot from L2 to G - FINISH!



System Architecture

Lifecycle of one robot from S (Town_A) to G (Town_B)

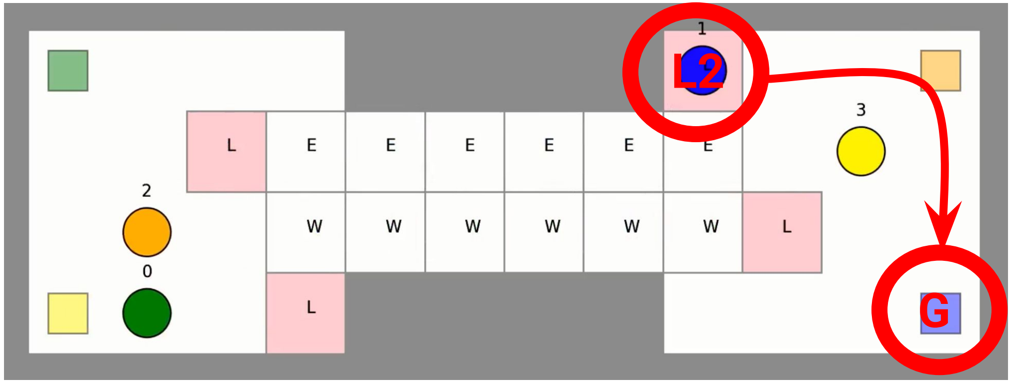
1. START: Global assign Locks pair (L1, L2)
2. Town_A route robot from S to L1 - Global
3. **Road route robot to L1 to L2 - notify Global**
4. Town_B route robot from L2 to G - FINISH!



System Architecture

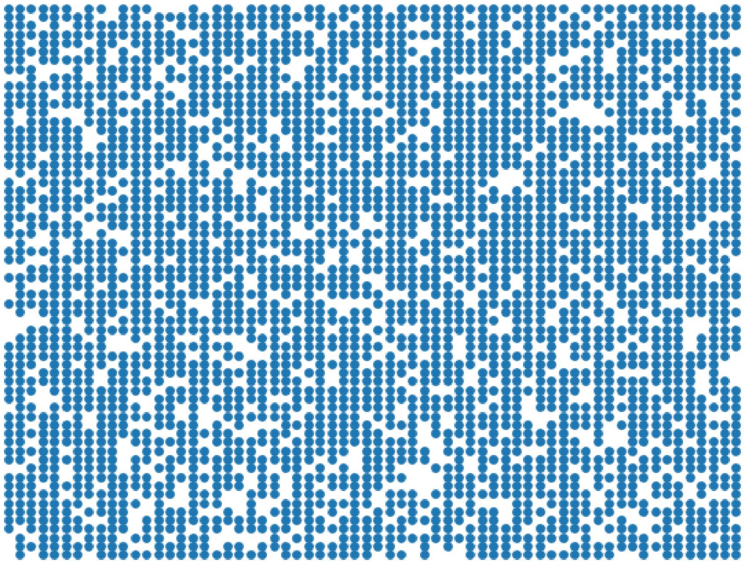
Lifecycle of one robot from S (Town_A) to G (Town_B)

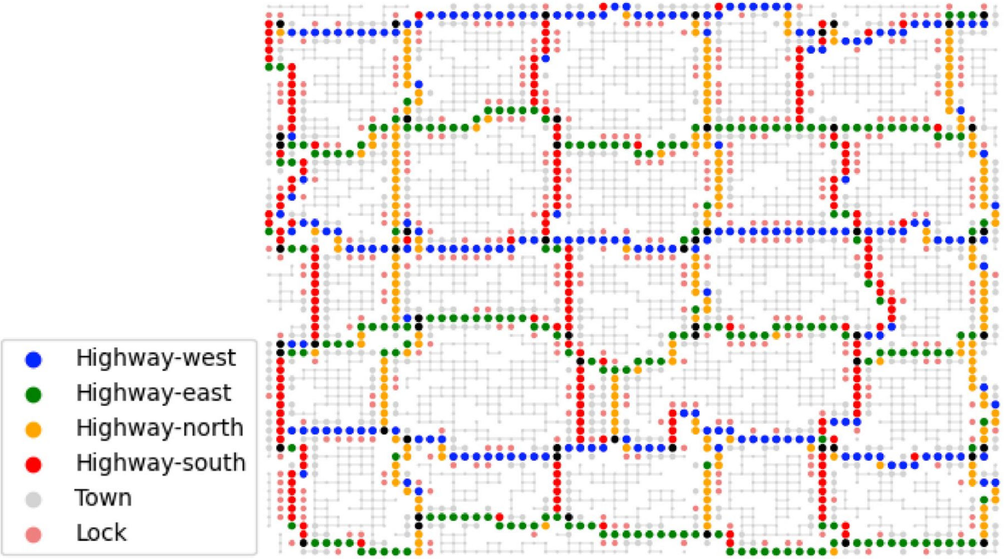
1. START: Global assign Locks pair (L1, L2)
2. Town_A route robot from S to L1 - Global
3. Road route robot to L1 to L2 - Global
4. **Town_B route robot from L2 to G - FINISH!**



System Architecture

Original Map → Road & Town Map





Contents

- Introduction, Impacts, and Goals
- Problem Formulation
- Social and Ethical Impact
- System Architecture
- **Implementation**
- Result Demonstration

Software Development Skills and Tools

C++ used for scalability.

GitHub used for version control and backup.

Software Development Skills and Tools

VSCode IDE used for programming.

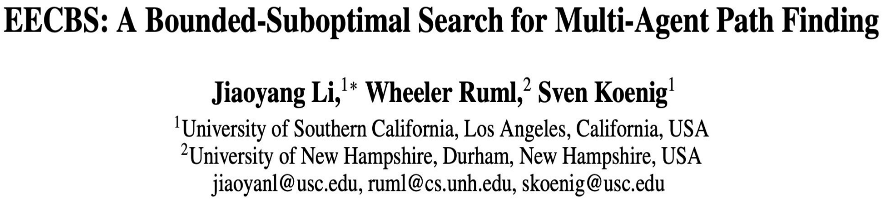
AWS used for scalability and future deployment.

Software Development Skills and Tools

Lucidchart used for diagram graphing.

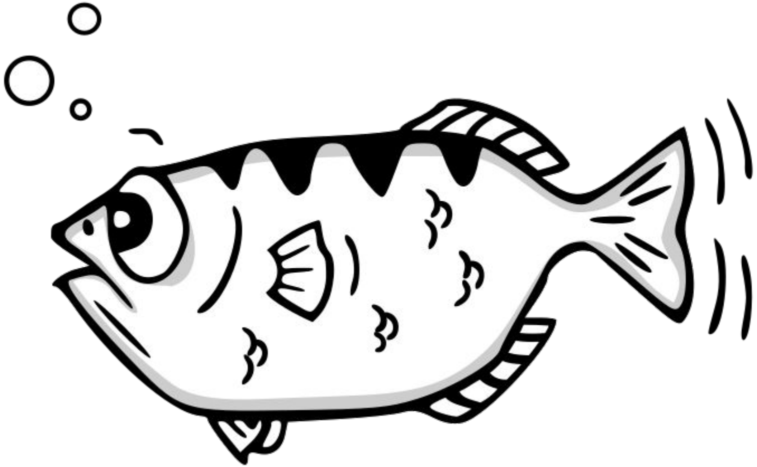
Pyplot used for result visualization.

Software Development Skills and Tools



EECBS: A Bounded-Suboptimal Search for Multi-Agent Path Finding

Jiaoyang Li,[1*] Wheeler Ruml,[2] Sven Koenig[1]
[1]University of Southern California, Los Angeles, California, USA
[2]University of New Hampshire, Durham, New Hampshire, USA
jiaoyanl@usc.edu, ruml@cs.unh.edu, skoenig@usc.edu

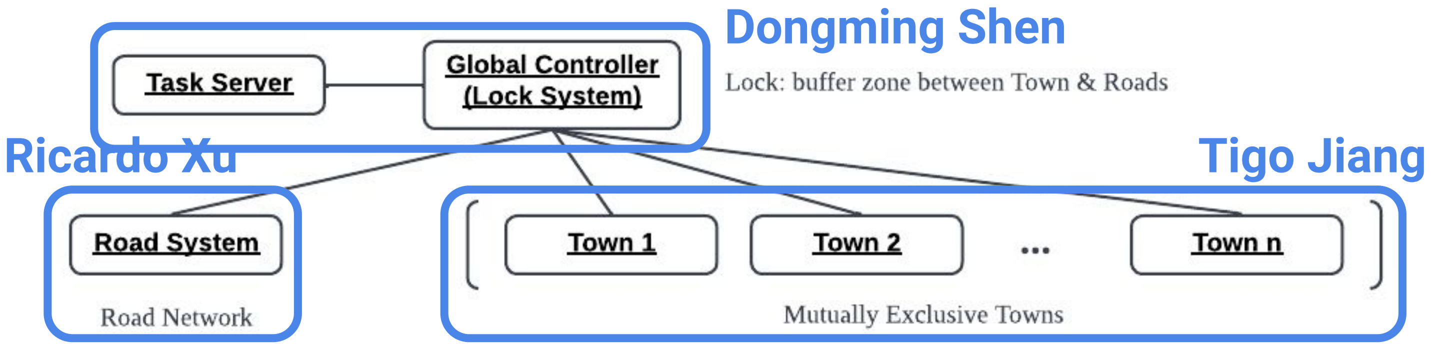


EECBS used as the SOTA town MAPF solver.

GNU Debugger (GDB) used for debugging.

Development Methodology

Teamwork and Responsibility

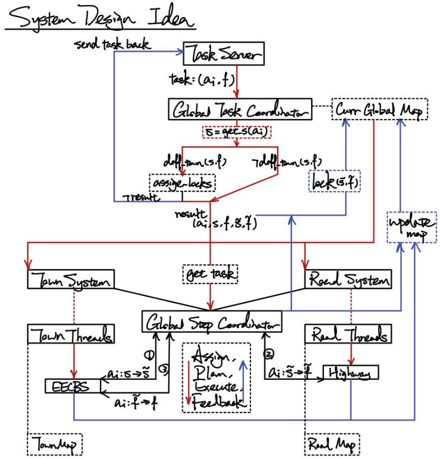


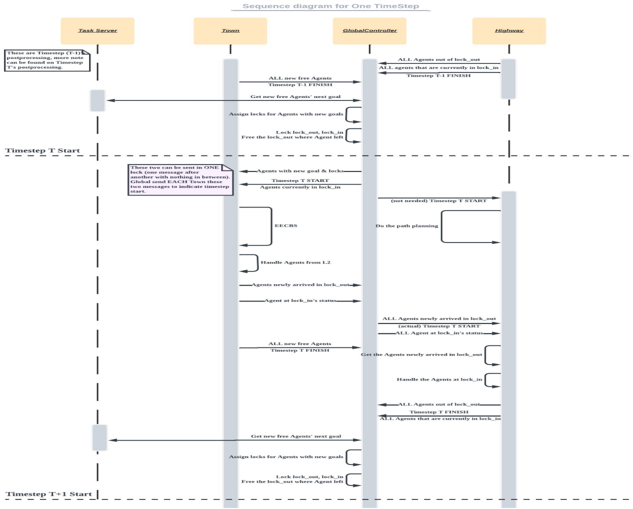
- Divide three main system components (Global, Town, Road) evenly.
- Other minor components (main.cpp, Agent, Grid, etc.) are implemented together.
- **Pros**: Allowed parallel development of main components.
- **Cons**: Debugging and Communication became difficult.

Development Methodology

Difficulties and Solutions

1. System Architecture: Pseudo Code v.s. Formal System Diagrams




Sequence diagram for One TimeStep
Task Server
Town
GlobalController
Highway
These are Timestep (T-1)'s postprocessing, more note can be found on Timestep T's postprocessing.
ALL Agents out of lock_out
ALL agents that are currently in lock_in
Timestep T-1 FINISH
ALL new free Agents
Timestep T-1 FINISH
Get new free Agents' next goal
Assign locks for Agents with new goals
Lock lock_out, lock_in
Free the lock_out where Agent left
Timestep T Start
These two can be sent in ONE lock (one message after another with nothing in between). Global send EACH Town these two messages to indicate timestep start.
Agents with new goal & locks
Timestep T START
Agents currently in lock_in
(not needed) Timestep T START
EECBS
Do the path planning
Handle Agents from L2
Agents newly arrived in lock_out
Agent at lock_in's status
ALL Agents newly arrived in lock_out
(actual) Timestep T START
ALL Agent at lock_in's status
ALL new free Agents
Timestep T FINISH
Get the Agents newly arrived in lock_out
Handle the Agents at lock_in
ALL Agents out of lock_out
Timestep T FINISH
ALL Agents that are currently in lock_in
Get new free Agents' next goal
Assign locks for Agents with new goals
Lock lock_out, lock_in
Free the lock_out where Agent left
Timestep T+1 Start

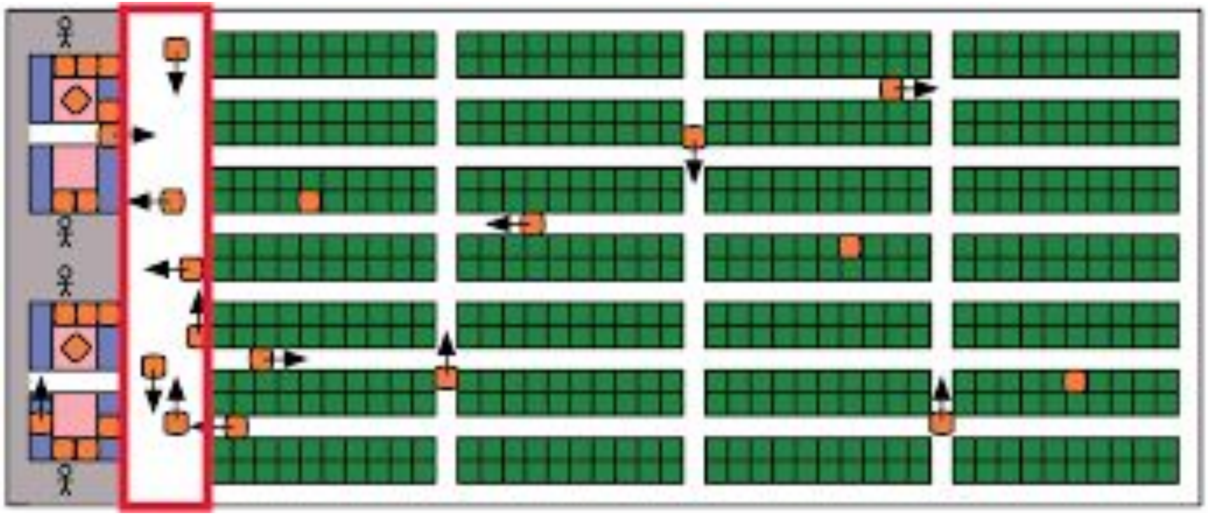
Development Methodology

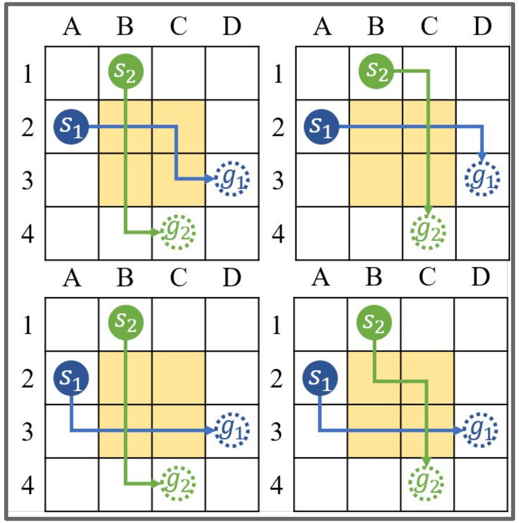
Difficulties and Solutions

2. Debugging Separately v.s. Localized Collaboration

Continuing Professional Development

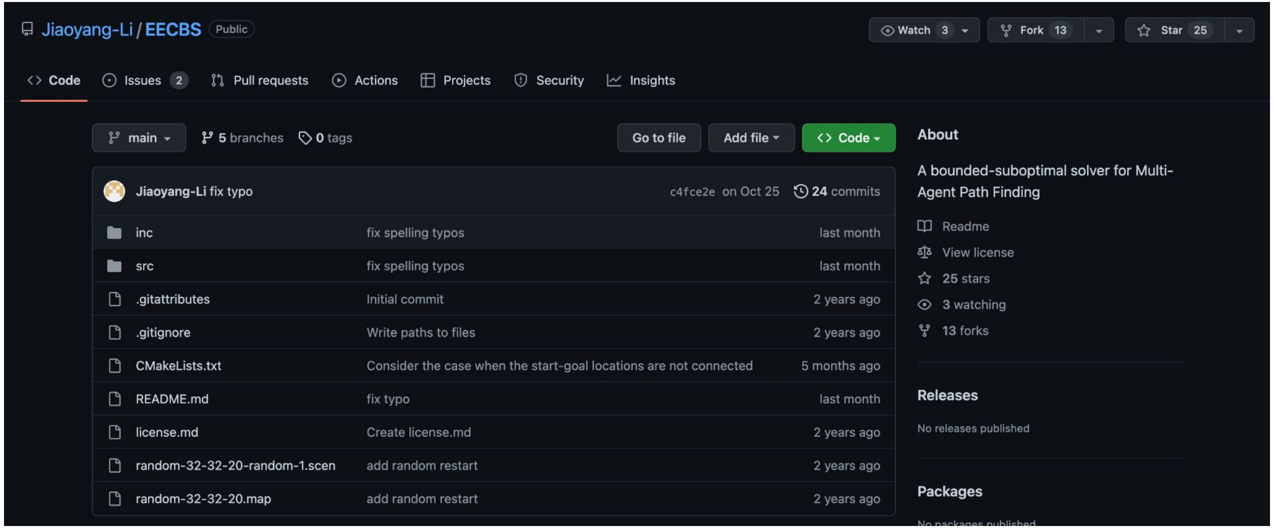
1. MAPF and multi-robot systems





Continuing Professional Development

2. Explicit Estimation Conflict-Based Search (EECBS): Town



Jiaoyang-Li fix typo c4fce2e on Oct 25 24 commits

inc	fix spelling typos	last month
src	fix spelling typos	last month
.gitattributes	Initial commit	2 years ago
.gitignore	Write paths to files	2 years ago
CMakeLists.txt	Consider the case when the start-goal locations are not connected	5 months ago
README.md	fix typo	last month
license.md	Create license.md	2 years ago
random-32-32-20-random-1.scen	add random restart	2 years ago
random-32-32-20.map	add random restart	2 years ago

About

A bounded-suboptimal solver for Multi-Agent Path Finding

Readme

View license

25 stars

3 watching

13 forks

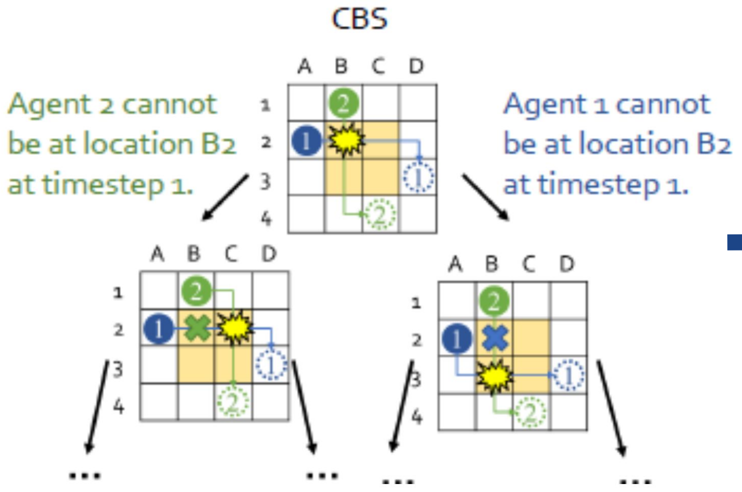
Releases

No releases published

Packages

No packages published

3. Rule-Based Search: Road

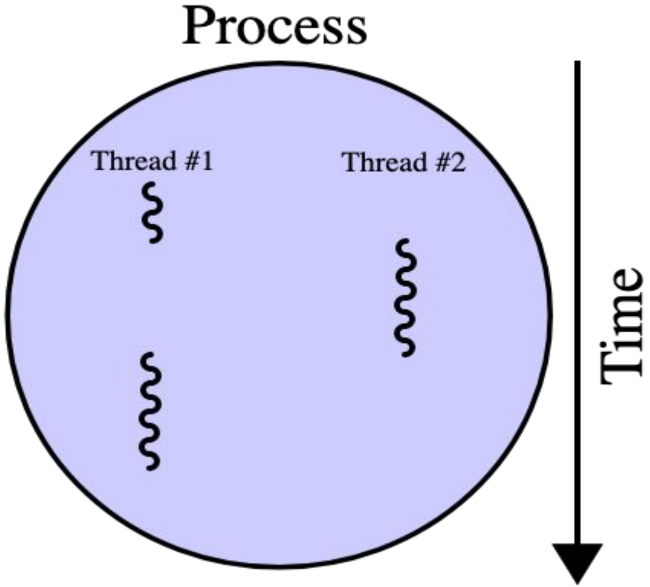


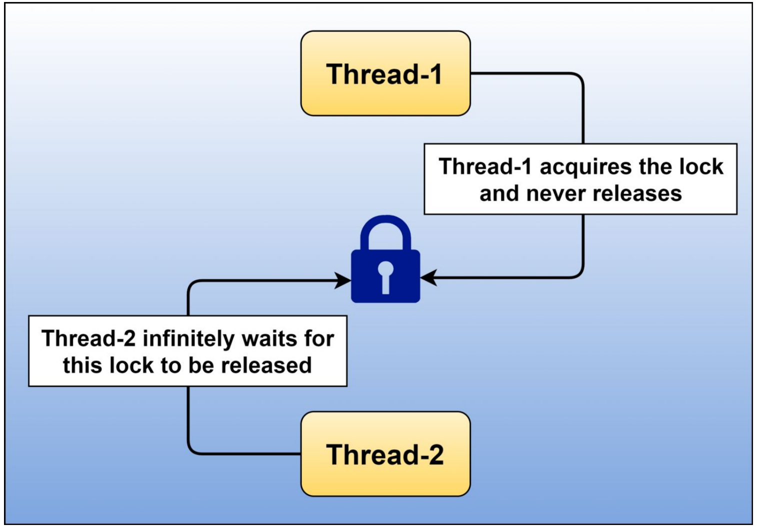
Agent 2 cannot be at location B2 at timestep 1.

Agent 1 cannot be at location B2 at timestep 1.

Continuing Professional Development

4. Multithreading and Lock Mechanism





Thread-1

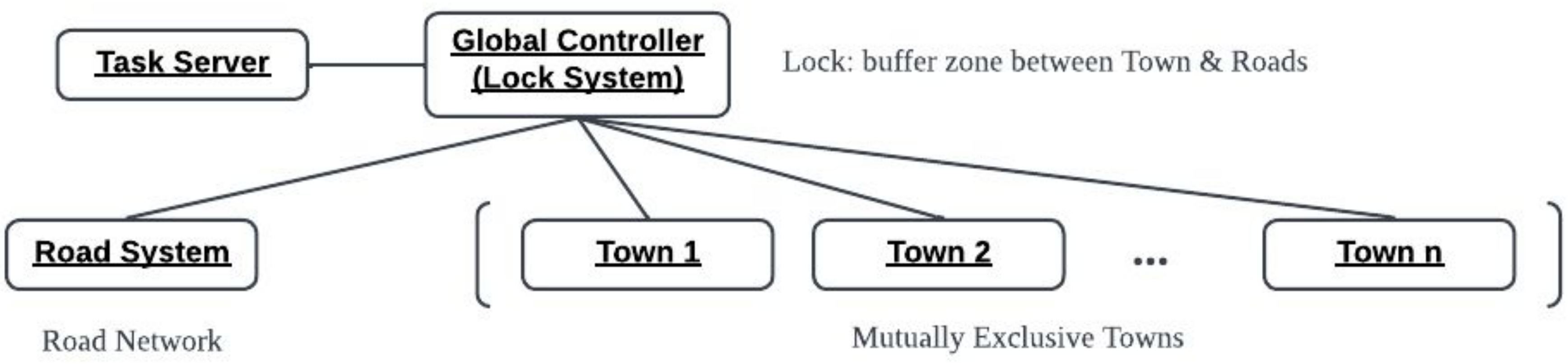
Thread-1 acquires the lock and never releases

Thread-2 infinitely waits for this lock to be released

Thread-2

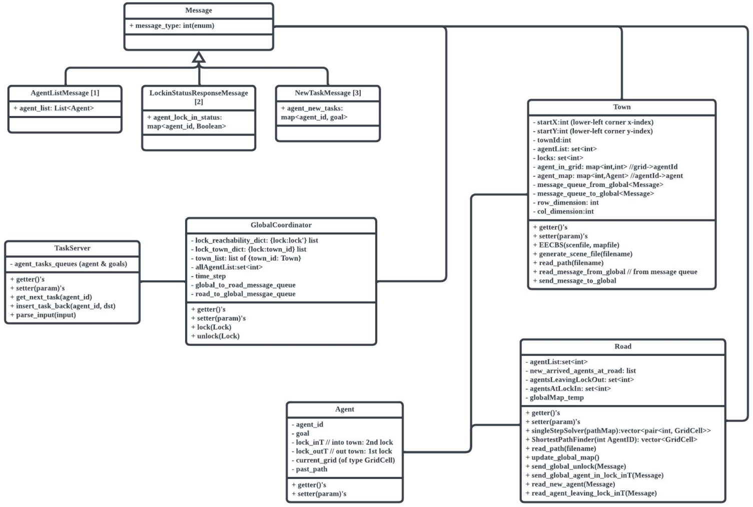
Continuing Professional Development

5. Distributed System Architecture



Continuing Professional Development

6. Architecture Communication - System Diagrams



Message

+ message_type: int(enum)

AgentListMessage [1]

+ agent_list: List<Agent>

LockinStatusResponseMessage [2]

+ agent_lock_in_status: map<agent_id, Boolean>

NewTaskMessage [3]

+ agent_new_tasks: map<agent_id, goal>

Town

- startX:int (lower-left corner x-index)
- startY:int (lower-left corner y-index)
- townId:int
- agentList: set<int>
- locks: set<int>
- agent_in_grid: map<int,int> //grid->agentId
- agent_map: map<int,Agent> //agentId->agent
- message_queue_from_global<Message>
- message_queue_to_global<Message>
- row_dimension: int
- col_dimension:int

+ getter()'s
+ setter(param)'s
+ EECBS(scenfile, mapfile)
+ generate_scene_file(filename)
+ read_path(filename)
+ read_message_from_global // from message queue
+ send_message_to_global

TaskServer

- agent_tasks_queues (agent & goals)

+ getter()'s
+ setter(param)'s
+ get_next_task(agent_id)
+ insert_task_back(agent_id, dst)
+ parse_input(input)

GlobalCoordinator

- lock_reachability_dict: {lock:lock'} list
- lock_town_dict: {lock:town_id} list
- town_list: list of {town_id: Town}
- allAgentList:set<int>
- time_step
- global_to_road_message_queue
- road_to_global_messgae_queue

+ getter()'s
+ setter(param)'s
+ lock(Lock)
+ unlock(Lock)

Road

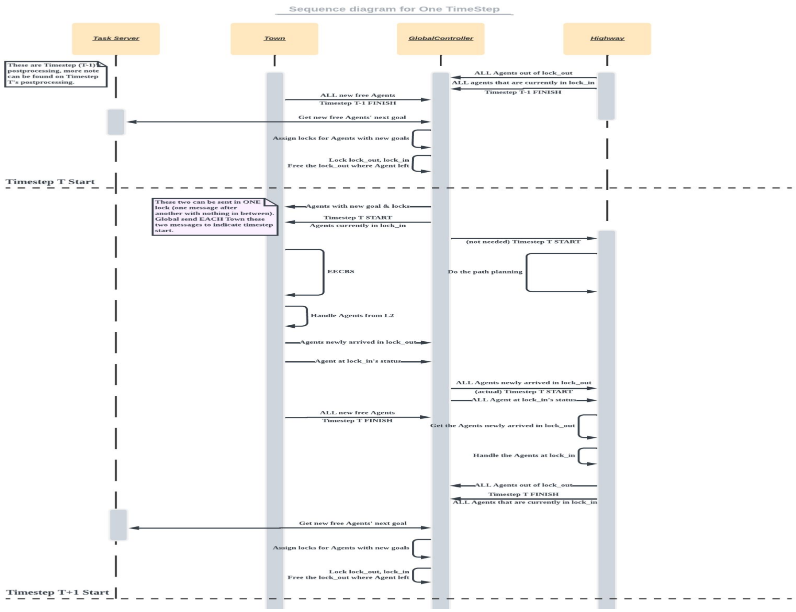
- agentList:set<int>
- new_arrived_agents_at_road: list
- agentsLeavingLockOut: set<int>
- agentsAtLockIn: set<int>
- globalMap_temp

+ getter()'s
+ setter(param)'s
+ singleStepSolver(pathMap):vector<pair<int, GridCell>>
+ ShortestPathFinder(int AgentID): vector<GridCell>
+ read_path(filename)
+ update_global_map()
+ send_global_unlock(Message)
+ send_global_agent_in_lock_inT(Message)
+ read_new_agent(Message)
+ read_agent_leaving_lock_inT(Message)

Agent

- agent_id
- goal
- lock_inT // into town: 2nd lock
- lock_outT // out town: 1st lock
- current_grid (of type GridCell)
- past_path

+ getter()'s
+ setter(param)'s



Sequence diagram for One TimeStep

Task Server

Town

GlobalController

Highway

These are Timestep (T-1)'s postprocessing, more note can be found on Timestep T's postprocessing.

ALL Agents out of lock_out

ALL agents that are currently in lock_in

Timestep T-1 FINISH

ALL new free Agents

Timestep T-1 FINISH

Get new free Agents' next goal

Assign locks for Agents with new goals

Lock lock_out, lock_in

Free the lock_out where Agent left

Timestep T Start

These two can be sent in ONE lock (one message after another with nothing in between). Global send EACH Town these two messages to indicate timestep start.

Agents with new goal & locks

Timestep T START

Agents currently in lock_in

(not needed) Timestep T START

EECBS

Do the path planning

Handle Agents from L2

Agents newly arrived in lock_out

Agent at lock_in's status

ALL Agents newly arrived in lock_out

(actual) Timestep T START

ALL Agent at lock_in's status

ALL new free Agents

Timestep T FINISH

Get the Agents newly arrived in lock_out

Handle the Agents at lock_in

ALL Agents out of lock_out

Timestep T FINISH

ALL Agents that are currently in lock_in

Get new free Agents' next goal

Assign locks for Agents with new goals

Lock lock_out, lock_in

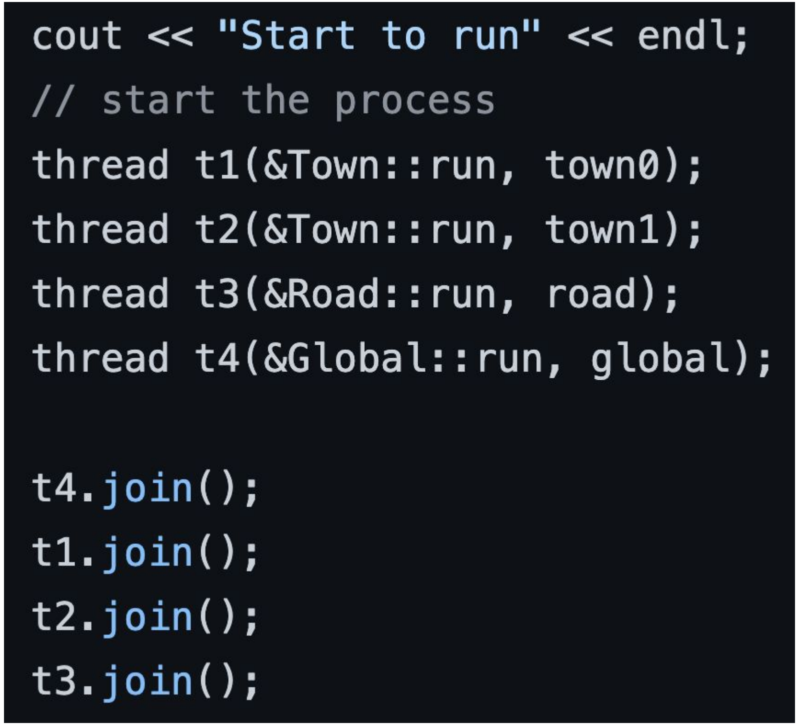
Free the lock_out where Agent left

Timestep T+1 Start

Contents

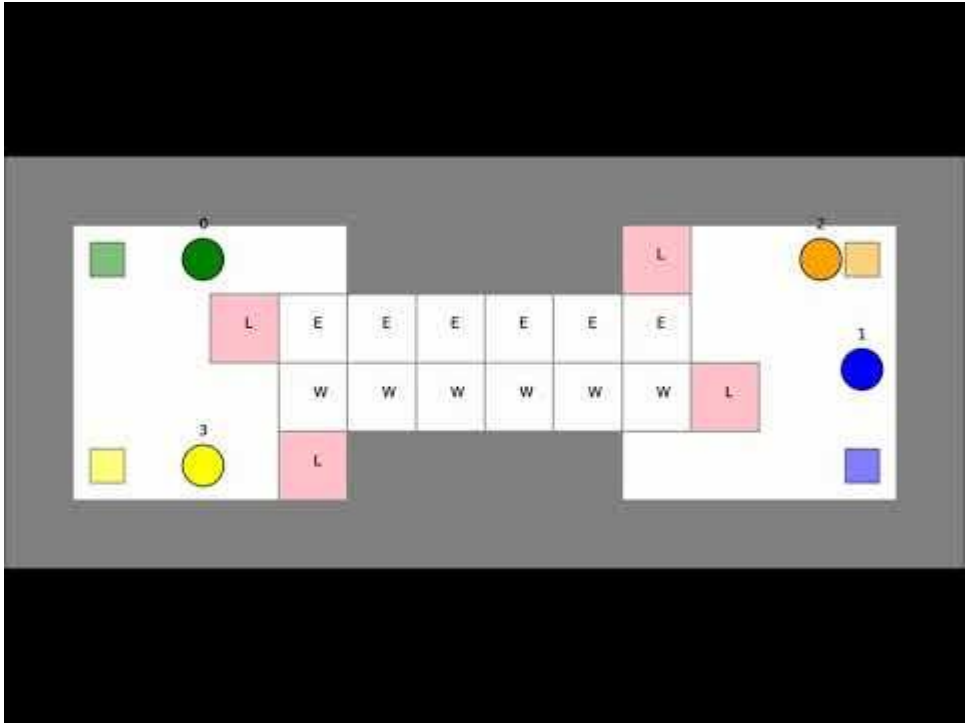
- Introduction, Impacts, and Goals
- Problem Formulation
- Social and Ethical Impact
- System Architecture
- Implementation
- **Result Demonstration**

Code and Result Demonstration



```
cout << "Start to run" << endl;
// start the process
thread t1(&Town::run, town0);
thread t2(&Town::run, town1);
thread t3(&Road::run, road);
thread t4(&Global::run, global);

t4.join();
t1.join();
t2.join();
t3.join();
```



Conclusion: All Goals Accomplished

1. Design an **architecture** for a town and road system.
2. **Implement** this architecture.
3. **Evaluate** our implementation.

Future Extension:

- Finetune and test on larger MAPF problems.

Thank You!

Dongming Shen, Ricardo Xu, Tigo Jiang

Stakeholder: Christopher Leet