

# # Pseudo Lab: AI in Logistics & Transportation

## MAPF-GPT: Imitation Learning for Multi-Agent Pathfinding at Scale

Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov, Alexey Skrynnik, AAAI, 2025.

---

Apr 8, 2025

Seungmin Jeon

[simonjeon825@gmail.com](mailto:simonjeon825@gmail.com)

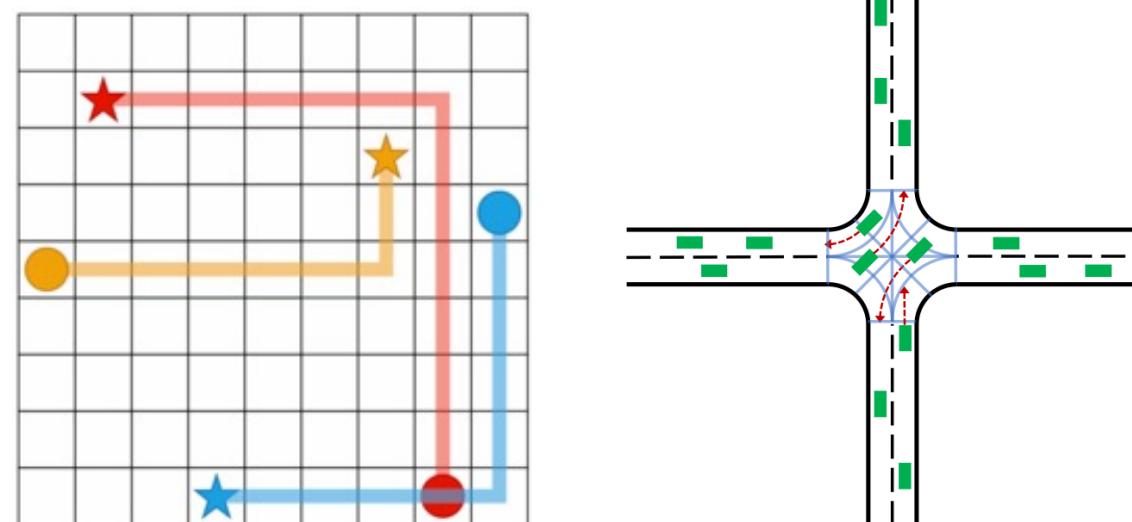
## **# Content**

---

- 1. What is MAPF**
- 2. Introduction**
- 3. Method**
- 4. Experimental Evaluation**
- 5. Conclusion**

# # What is MAPF?

- Multi-agent Pathfinding (MAPF)는 여러 Agent가 동일한 공간에서 출발지에서 목적지까지 서로 충돌(conflict)하지 않는 Path 를 찾는 문제
- MAPF 솔루션은 Warehouse, Autonomous Vehicle, Robotics, Transportation System 등의 여러 현실 세계의 도메인과 어플리케이션에서 활용될 수 있음
- Graph Representation, Discretize Time, Uniform Duration of Actions 등의 단순화를 위한 가정을 하여도 Optimal Solution 을 도출하는 것은 NP-Hard 문제



Multi-agent Pathfinding examples [1, 2]

[1] "Multi-agent pathfinding," Wikipedia. [Online]. Available: [https://en.wikipedia.org/wiki/Multi-agent\\_pathfinding](https://en.wikipedia.org/wiki/Multi-agent_pathfinding). [Accessed: Apr. 07, 2025].

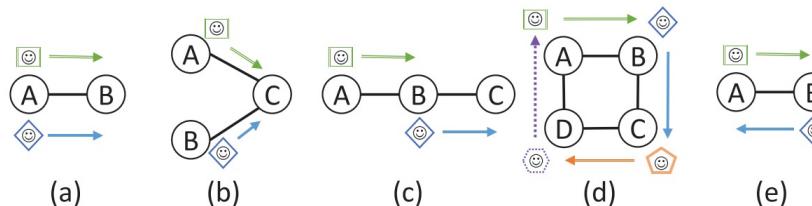
[2] Z. Yan, H. Zheng and C. Wu, "Multi-agent Path Finding for Cooperative Autonomous Driving," 2024 IEEE International Conference on Robotics and Automation (ICRA), Yokohama, Japan, 2024, pp. 12361-12367.

# # What is MAPF?

- Classical MAPF 문제는  $k$  개의 Agent와 Tuple  $\langle G, s, t \rangle$ 로 정의할 수 있음
  - $G = (V, E)$ 는 Undirected Graph
  - $s$ 는 Start Vertex,  $t$ 는 Target Vertex로 각각  $k$ 개 있음
- Time은 Discrete 하다 가정하며, Agent는 Vertex에서 하나의 Time Step에 하나의 Action을 수행
- Agent는 Wait 또는 Move의 Action을 가질 수 있음
- Action ( $a$ )의 Sequence를 Single-agent plan ( $\pi$ )이라 하며, 이의 Set을 Joint plan ( $\Pi$ )

$$\pi = (a_1, \dots, a_n)$$

- Conflict으로는 Edge, Vertex, Following, Cycle, Swapping 이 있음



- 하나 이상의 Valid Solution을 가질 수 있으며, Objective Function을 정의하여 이에 대한 Optimal Solution을 찾는 것을 목표
  - Makespan: 모든 agent가 target에 도달하기까지의 time step
  - Sum of Costs (SoC): 모든 agent의 time step의 합

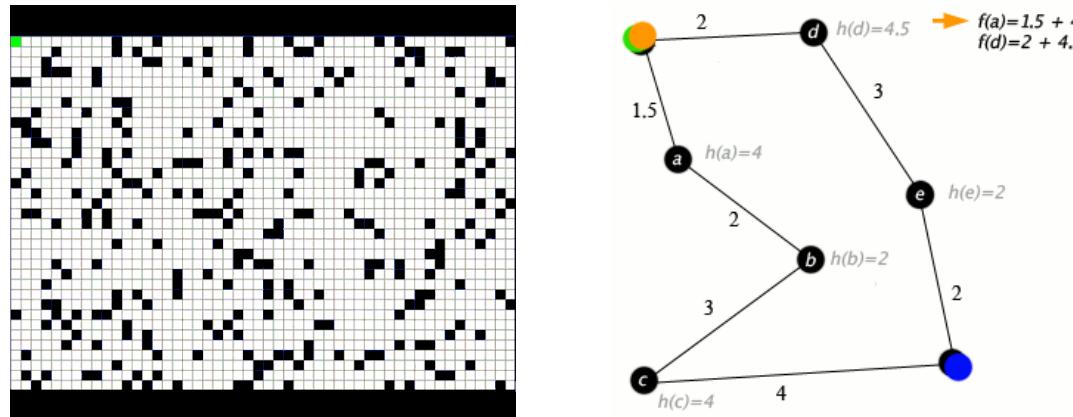
$$M(\Pi) = \max_{1 \leq i \leq k} |\pi_i| \quad \text{SOC}(\Pi) = \sum_{1 \leq i \leq k} |\pi_i|$$

R. Stern, "Multi-Agent Path Finding - An Overview," 2019, pp. 96-115. doi: 10.1007/978-3-030-33274-7\_6.

R. Stern et al., "Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks," in Proceedings of the International Symposium on Combinatorial Search, Sep. 2021, pp. 151-158.

# # What is MAPF?

- Single-agent Shortest-path Problem (SPP)인 경우 A\* Algorithm을 활용하여 Optimal Solution을 찾을 수 있음



A\* Search Algorithm [1]

- MAPF를 k 개 Agent의 Shortest-path Problem으로 보았을 때, Agent의 Path를 찾는 과정에서 다른 Agent가 Time Step에 따라 이동하며 Occupy된 Vertex가 변경되며 이를 고려하여 Path를 찾아야 함
- A\*를 사용하는 경우 Search Space Size ( $|V|^k$ )와 Branching Factor( $(\frac{|V|}{|E|})^k$ )를 통하여 어려움을 대략적으로 표현할 수 있으며, Agent 개수에 따라 Exponential하게 증가하여, 일반적인 A\*를 사용할 수 없음

[1] “A\* Search Algorithm,” Wikipedia. [Online]. Available: [https://en.wikipedia.org/wiki/A\\*\\_search\\_algorithm](https://en.wikipedia.org/wiki/A*_search_algorithm). [Accessed: Apr. 07, 2025].

# # What is MAPF?

- MAPF의 Methodology는 아래와 같이 분류할 수 있음 [1]
  - Heuristic (Fast MAPF): Prioritized Planning (PP)
  - Optimal-based: Conflict-based Search (CBS), Constraints Programming, Extension of A\*

De-centralized		Centralized	
		Heuristic	Optimal-based
Path Generation	Each Agent	Server	
Advantage	Fast Computation		Complete (=if there is a path, it can be found)
Disadvantage	Incomplete (No deadlock is guaranteed)		Slow Computation
Applicable Scenario	Simple environment and/or very large scale fleet		Complicated Environment

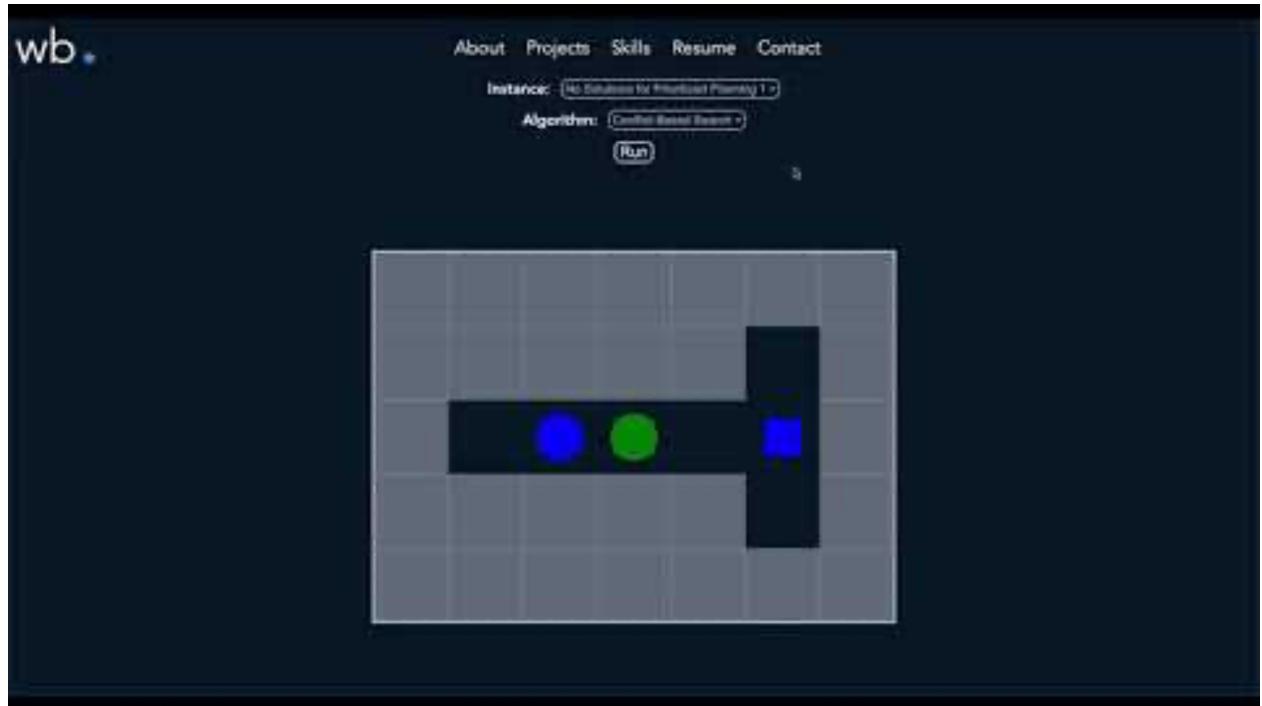
[1] Youngjae Kim, "Cloud Robotics and On-device AI in the Generative AI Era," 21st International Conference on Ubiquitous Robots, New York, USA, 2024

# # What is MAPF?

- 여러 Agent 사이에서 협동이 필요한 복잡한 상황에서 일반적인 De-centralized & Prioritized Planning 같은 Methodology는 한계가 있음



Waymo Self Driving Cars Herding [1]



MAPF Example [2]

[1] David P., "Waymo Self Driving Cars Herding," LinkedIn, [Online]. Available: [https://www.linkedin.com/posts/cyberspacedave\\_aihumor-cyberspacedave-activity-7227779838390976512-ydl\\_](https://www.linkedin.com/posts/cyberspacedave_aihumor-cyberspacedave-activity-7227779838390976512-ydl_) [Accessed: Apr. 07, 2025].

[2] "MAPF Example," You Tube, [Online], <https://youtu.be/P3NZmNuEscM?si=15dPpuLuibJlr2yU&t=16> [Accessed: Apr. 07, 2025].

# # Introduction

---

## Introduction

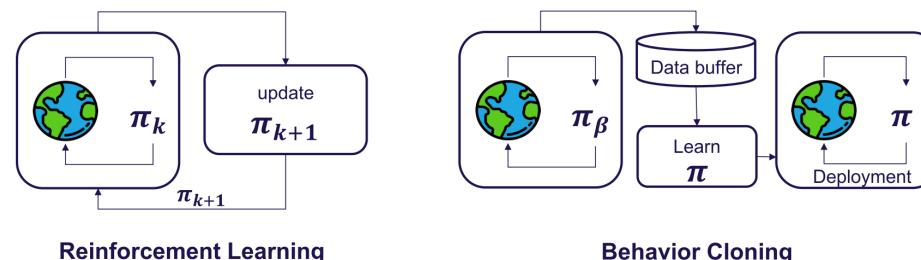
- NP-Hard인 MAPF에서 Heuristics을 기반한 Rule-based나, Well-established Computer Science Problem으로 변환하여 Optimal Solution을 구할 수 있으나, 완전성 부족, 계산 비용, 확장성 및 현실 제약 반영 어려움 등의 여러 문제 사항들이 있었음
- 이러한 한계를 극복하기 위하여 learning-based가 최근에 연구되고 있으며, 특히 Deep Reinforcement Learning (DRL)이 주로 활용되고 있음
- 해당 논문은 SOTA MAPF Solver (LaCAM)을 활용하여 Expert Data를 구성하고, Non-autoregressive Neural Network를 기반한 Transformer에 입력으로 이를 활용할 수 있도록 Token을 정의하여, Observation을 기반해 Action을 예측

## Contribution

- 대규모 Expert Dataset 구축 (1B Observation-action pairs)
- Agent Observation을 MAPF-GPT에서 활용을 위한 Tokenization Procedure 개발
- Imitation Learning 을 기반하여 Transformer-based Neural Network를 학습
- Zero-shot generalization과 fine-tuning에서 범용성과 재사용성 측면에서 MAPF의 Foundation Model로 활용할 수 있음
- De-centralized MAPF Solver로 타 SOTA Learning-based Methodology보다 좋은 성능과 시간 효율을 보임

# # Introduction

- Reinforcement Learning
  - Trial and Error을 통하여 학습하는 방법으로, 주로 Deep Q Network (DQN) 을 다양하게 확대하여 사용하고 있음
  - DQN은 State와 Action Input에서 Action에 따른 Reward를 예측하는 Q Network를 학습하며, 이를 통해 Q Value를 최대로 하는 Policy을 적용하여 Action을 선택
  - Simulation의 결과물을 Real World에서 활용함에 어려움이 있으며, 적절한 Reward Function을 구성하는 것 또한 어려움이 있음
- Imitation Learning (Behavior Cloning)
  - Agent Policy을 학습하기 위해 Expert Policy가 담긴 Dataset을 활용하여 직접 모방을 하도록 학습
  - Supervised Learning을 주로 활용하며 Loss Function로 Expert와 Agent의 Action의 차이를 활용
  - Reinforcement Learning과 같이 어려운 Reward 설계를 하지 않아도 되어 상대적으로 간단하고 효율적인 방법론
  - Training에서 State을 Independently and Identically Distribute (i.i.d)로 다루어, Covariate Shift 문제가 발생



Reinforcement Learning & Imitation Learning [1]

[1] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, “End-to-end Autonomous Driving: Challenges and Frontiers,” arXiv: 2306.16927 [cs.RO], 2023.

# # Method

- Learning Phase의 MAPF-GPT는 4개의 Phase를 가짐

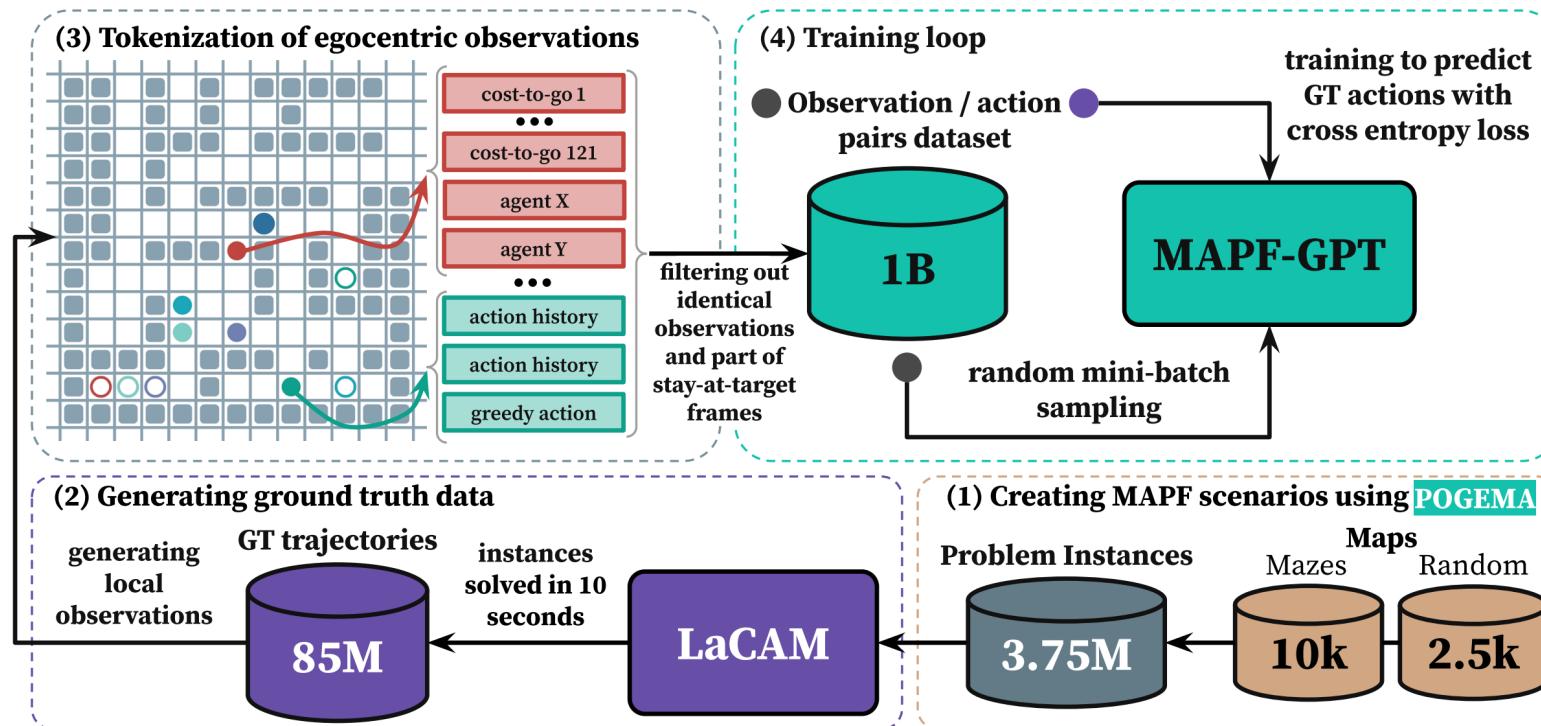
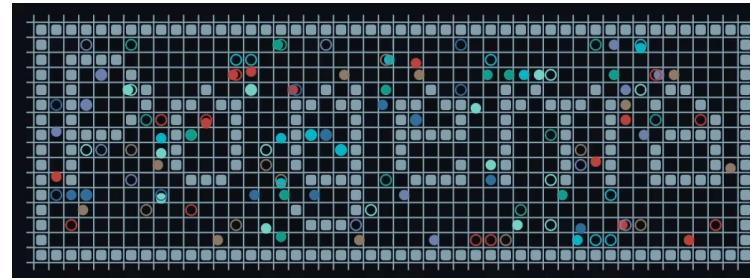


Figure 1: The general pipeline of the MAPF-GPT: (1) Creating MAPF scenarios. (2) Generating ground truth data, i.e. MAPF solutions using an expert solver. (3) Transforming the solutions to the observation-action pairs and tokenization of the observations, which converts them into a format suitable for transformer architectures. (4) Executing the training loop, where observation/action pairs are sampled from the dataset, and the model is trained using cross-entropy loss.

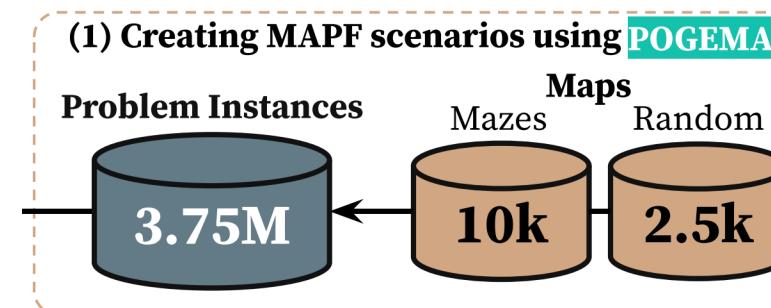
## # Method: (1) Creating MAPF scenarios using POGEMA

- POGEMA (Map을 생성하는 등의 기능을 제공하는 MAPF 개발 Tool) 를 활용하여 Data Set 구축



POGEMA [1]

- 10k Maze와 2.5k Random Map을 기반하여 3.75M의 Problem을 구성
- Map Size는 17x17 - 21x21이며, Agent는 16, 24, 32개로 구성
- De-centralized 이기 때문에 Map Size 보다는 Density가 중요함



[1] “pogema,” Github. [Online]. Available: <https://github.com/CognitiveAISystems/pogema>. [Accessed: Apr. 07, 2025].

## # Method: (2) Generating ground truth data

- SOTA MAPF Solver인 LaCAM [1]을 활용하여 Expert Data를 생성 (Time Budget 10초)

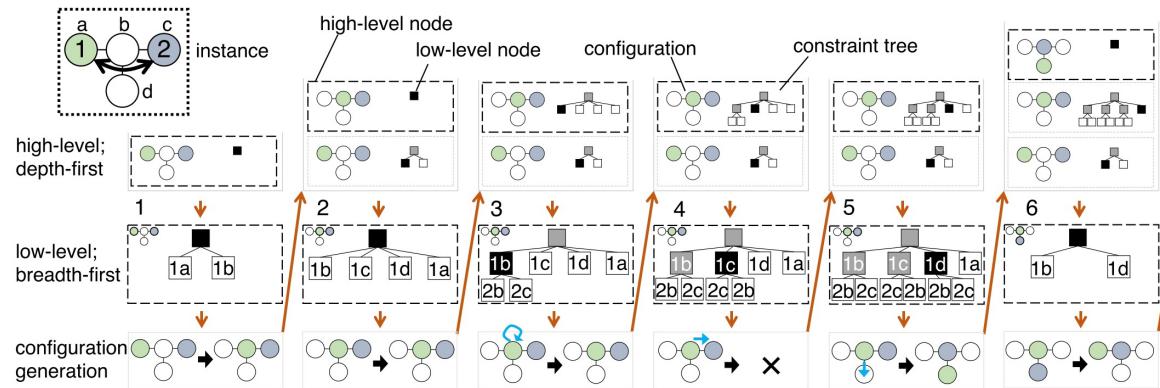


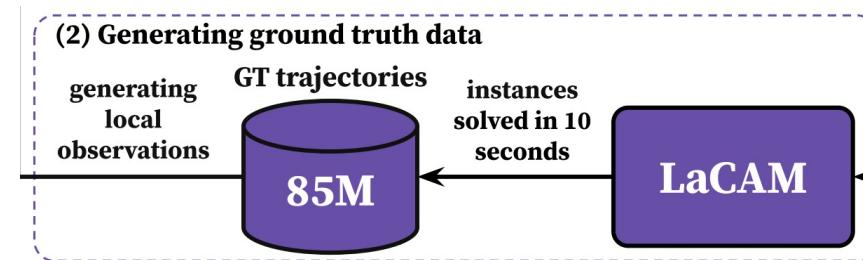
Figure 1: Running example of LaCAM. Orange arrows represent the search progress order. Selected and searched low-level nodes are filled with black and gray, respectively. Constraints are shown by blue-colored arrows.

- LaCAM의 output은 하나의 MAPF에서의 individual plan의 set이며, 이를 agent의 local observation으로 변환 작업 수행

[1] K. Okumura, “LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding,” Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, no. 10, pp. 11655–11662, Jun. 2023, doi: 10.1609/aaai.v37i10.26377.

## # Method: (2) Generating ground truth data

- Post-processing을 수행
  - 만약 동일한 Observation을 공유한다면, 하나만 남김
  - 다른 Agent가 도착하기까지 Goal에서 기다리는 시간이 너무 길면 제거  
(Original Data의 40 %가 waiting이라, imbalance를 해결하고자 80 %의 wait-at-target action 제거)
- 총 1B의 Observation-action Pair 구성 (Maze-like에서 900M, random에서 100M)
  - Maze가 좁은 통로가 있어서, agent 사이에서 더 높은 수준의 cooperation이 필요



# # Method: (3) Tokenization of egocentric observations

- Tokenization은 Observation-action Pair Data를 Token이란 Special Symbol의 Sequence로 변환하는 과정으로, Input Tokens (Sequence)을 학습하여 Neural Network는 Single Token (Action)을 Predict
- Local Observation은 두 개의 Part로 구성
- Map 관련 Part
  - Traversable Cell은 cost-to-go를 계산 (Egocentric하게 변환)
  - Blocked Cell은 Infinite
- 주변 Agent 관련 Part
  - Relative Position
  - Relative Target Position
  - Greedy Action: cost-to-go가 감소하기 위한 action 방향 (여러 방향일 수 있음)
  - Action History

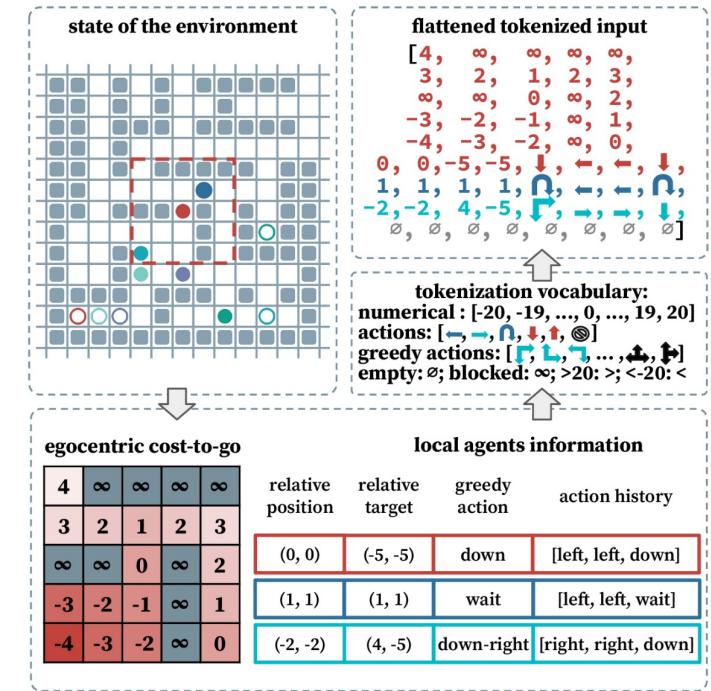


Figure 2: The tokenization process for the MAPF-GPT model uses a vocabulary of 67 tokens, with an input of 256 tokens. Fewer tokens are shown for clarity and visibility.

# # Method: (3) Tokenization of egocentric observations

- Local Observation을 Encode하여 256 Tokens을 생성
- 11x11 Field of View를 구성하여, Map에 대하여 121 Tokens 사용
- 각각의 Agent 별로 10 Tokens (2 Position, 2 Target, 5 Action History, 1 Greedy Action)을 구성하여, Ego를 포함하여 최대 13 개 Agent를 담을 수 있음
  - Agent가 충분하지 않으면 Empty Token 처리
  - Agent 정보는 거리 기준으로 Sort
- 남은 5 Tokens은 Empty Token
- Cost-to-go, Position 정보는 Numerical Value
- Action은 Literal
- Tokenization Vocabulary (67개)

종류	토큰 수	예시
숫자 범위 표현	41개	-20 ~ +20 (cost, 좌표 등)
특수 토큰	3개	<inf>, <-20, >+20
행동 토큰	6개	↑, ↓, ←, →, wait, empty
복합 행동 토큰	16개	↑→, ↓↔ 등
Empty	1개	empty

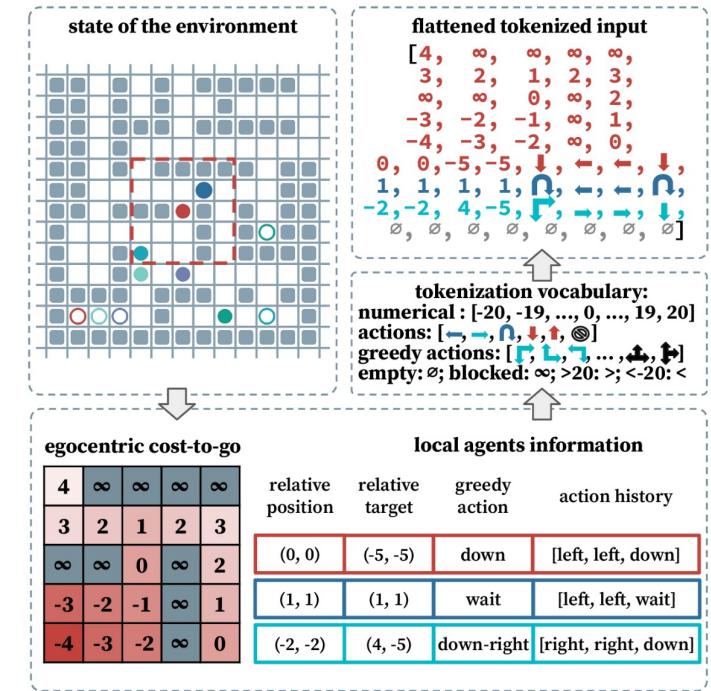


Figure 2: The tokenization process for the MAPF-GPT model uses a vocabulary of 67 tokens, with an input of 256 tokens. Fewer tokens are shown for clarity and visibility.

## # Method: (4) Model Training

---

- Expert solver인 LaCAM이 생성한 Action을 따라하는 Supervised Imitation Learning (Behavior Cloning)
  - Input: Agent의 Local Observation (256 Tokens)
  - Output: 해당 시점에서의 Expert Action
  - Model (Backbone: Decoder-only Transformer)은 주어진 Observation에 대해 다음 Action을 맞추는 Classification
  - Single Token 만을 Predict하기 때문에 일반적인 NLP와 달리 Causal masking은 사용하지 않음
- Loss Function은 Cross-entropy Loss으로, Local Observation이 주어졌을 때 Expert Action을 할 확률을 최대화

$$-\log \mathbf{p}_\theta(a_u^{LaCAM}(s) | o_u)$$

- Policy로 주어진 확률 분포에서 제일 높은 확률의 Action을 Sampling

$$\hat{a}^u(o_u) \sim \mathbf{p}_\theta(o_u)$$

- Parameter를 85M를 사용하였으며, 6M, 2M의 작은 Model도 학습

# # Method: (4) Model Training

- 학습 세부 사항 (Appendix B)
  - 85M Model: 1M iteration 243 시간 소요 (4x H100 80GB NVIDIA)

Parameter	Value
Minimum learning rate	6e-5
Maximum learning rate	6e-4
Learning rate decay	cosine
Warm-up iterations	2000
AdamW optimizer beta1	0.9
AdamW optimizer beta2	0.95
Gradient clipping	1.0
Weight decay	1e-1
Data type for computations	float16
Use PyTorch 2.0 compilation	True
Gradient accumulation steps	16
Block size	256

Table 3: Common hyperparameters for MAPF-GPT models.

Parameter	2M	6M	85M
Number of epochs	46.875	12.5	15.625
Total training iterations	15,000	30,000	1,000,000
Batch size	4096	2048	512
Number of layers	5	8	12
Number of attention heads	5	8	12
Embedding size	160	256	768

Table 4: Model-specific hyperparameters.

# # Experimental Evaluation: Main Result

- SOTA Learnable MAPF solver인 DCC와 SCRIMP를 비교
- Warehouse, MovingAI map은 모든 Learnable Solver에게 out-of-distribution
- MAPF-GPF-85M가 제일 좋은 성능을 보임

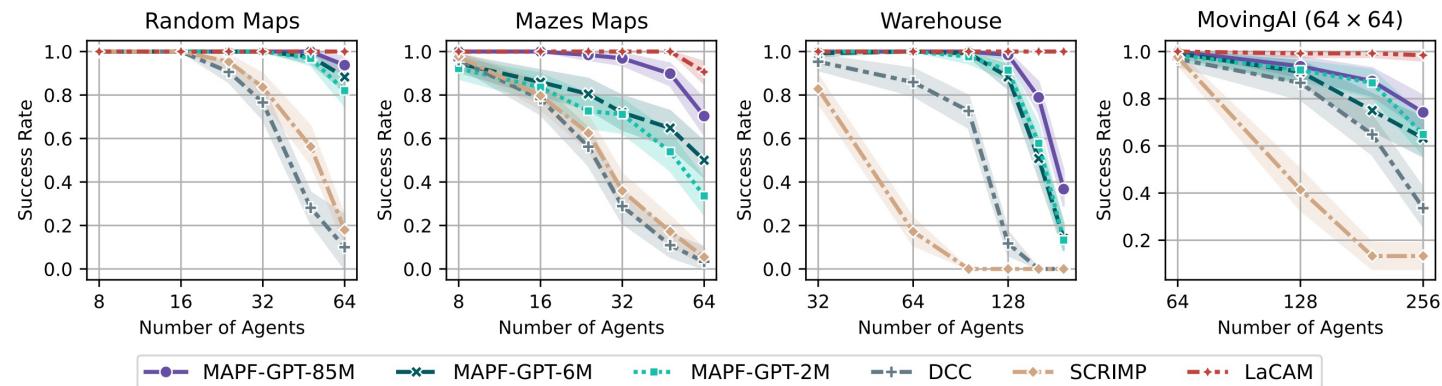


Figure 3: Success rate of the evaluated MAPF solvers on different maps. The shaded area indicates 95% confidence intervals.

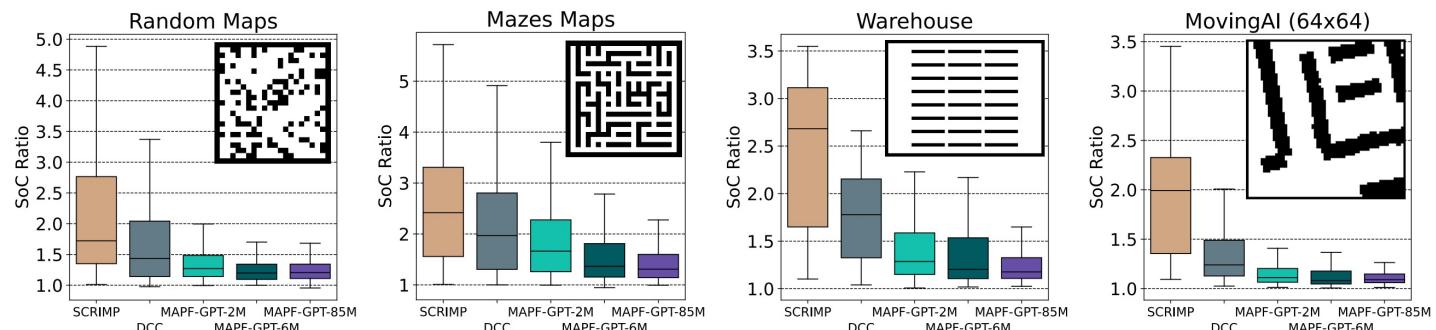


Figure 4: Quality of the obtained solutions relative to the ones of LaCAM (lower is better).

# # Experimental Evaluation: Ablation Study

- Token 중 일부를 Mask 하여 6M Model을 학습하여 어떤 요소가 영향을 주는지 파악
  - noGoal, noGA (Greedy Action), noAH (Action History), noC2G (Cost-to-go)
- Puzzles Map을 추가: 5x5으로 작아 cooperative action이 중요한 Map
- Random, Puzzles에서 Original Model이 좋은 성능을 보임
- MovingAI에서 noGoal이 좋은 성능
  - Large Map에서 충돌이 목표 지점에 도달하는 도중에 발생하기 때문에 정확한 Goal 정보는 중요하지 않음
  - Greedy Action 정보가 Goal 방향성을 대체할 수 있기 때문
- Mazes, Warehouse에서 noAH가 좋은 성능
  - Behavior Cloning에서 Action History가 중요하지 않다는 의미
  - 허나 저자는 Movement History를 유지하는게 fine-tuning에 중요하다 주장 (Future Work)
- noGA와 noC2G는 성능 하락이 확인이 되어 주요한 역할을 함을 확인

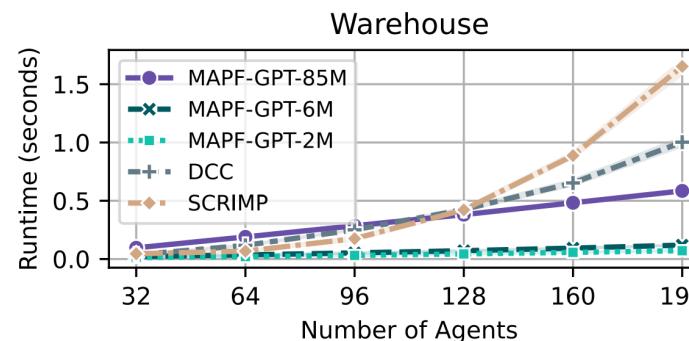
Scenario	6M	noGoal	noGA	noAH	noC2G
Random	97.6%	95.7%	97.0%	95.6%	25.8%
Mazes	74.6%	71.6%	37.6%	85.8%	15.1%
Warehouse	94.1%	92.8%	87.7%	94.8%	11.5%
MovingAI	82.0%	88.4%	79.1%	82.2%	10.2%
Puzzles	94.0%	92.7%	92.7%	91.5%	52.5%

# # Experimental Evaluation: LifeLong MAPF & Runtime

- LifeLong MAPF (LMAPF)
  - LMAPF: 현재 Goal에 도달했을 때 새로운 Goal을 주는 문제로, 전체 Agent가 Goal에 도달한 평균 (Throughput)이 Objective
  - MAPF-GPT를 Zero-shot과, Fine-tune을 위하여 RHCRO이란 LMAPF에 적합한 Solver를 사용하여 Expert Data (90 M)를 구성
  - Zero-shot이지만 기존의 Follower, MATS-LP와 유사한 성능을 보이며, Fine-tune 시 더 좋은 성능을 보임

Scenario	6M	6M tuned	RHCR	Follower	MATS-LP
Random	1.497	1.507	2.164	1.637	1.674
Mazes	0.908	1.087	1.554	1.140	1.125
Warehouse	1.113	1.270	2.352	2.731	1.701
MovingAI	2.840	2.994	3.480	3.271	3.320

- Runtime: 전체 Agent의 Next Action을 정하는데 걸린 평균 시간으로, MAPF-GPT는 선형적인 시간 증가를 보임



# # Conclusion

---

## Conclusion

- MAPF-GPT는 Heuristic이나 Communication 같은 추가적인 Module이나 Reinforcement Learning을 사용하지 않고, Imitation Learning을 통하여 Observation에서 Action을 예측하는 Policy 학습
- 대규모 Expert Data를 구축하였으며, 이를 기반하여 학습
- SOTA Learning-based 보다 우수한 성능과 효율을 보임
- Tokenization을 통한 Input 일반화, 여러 Map 또는 LMAPF 문제 등에서 좋은 성능을 통해, Zero-shot과 Fine-tuning의 범용성과 재사용성을 보여 Foundation Model로 가치가 있음

## Limitation

- 다른 Learnable Method와 같게 Theoretical Guarantee가 부족
- Large Model을 학습하는데 부담이 큼
- Expert Data Set의 Trajectory Quality에 Sensitive
- Centralized Approach를 Replicate하는 방법이 불분명

**Thank You!  
Any Questions?**

---