Conflict Mitigation in Shared Environments using Flow-Aware Multi-Agent Path Finding

Lukas Heuer[1], Yufei Zhu[1], Luigi Palmieri[2], Andrey Rudenko[3], Anna Mannucci[2], Sven Koenig[4] and Martin Magnusson[1]

***Abstract*— Deploying multi-robot systems in environments shared with dynamic and uncontrollable agents presents significant challenges, especially for large robot fleets. In such environments, individual robot operations can be delayed due to unforeseen conflicts with uncontrollable agents. While existing research primarily focuses on preserving the completeness of Multi-Agent Path Finding (MAPF) solutions considering delays, there is limited emphasis on utilizing additional environmental information to enhance solution quality in the presence of other dynamic agents. To this end, we propose Flow-Aware Multi-Agent Path Finding (FA-MAPF), a novel framework that integrates learned motion patterns of uncontrollable agents into centralized MAPF algorithms. Our evaluation, conducted on a diverse set of benchmark maps with simulated uncontrollable agents and on a real-world map with recorded human trajectories, demonstrates the effectiveness of FA-MAPF compared to state-of-the-art baselines. The experimental results show that FA-MAPF can consistently reduce conflicts with uncontrollable agents, up to 55%, without compromising task efficiency.**

I. Introduction

Multi-robot systems have demonstrated their effectiveness across various industries, including warehouse automation, intralogistics, and manufacturing [1]. Nevertheless, their current deployment is typically restricted to controlled environments where they can operate autonomously, isolated from human interactions and other unpredictable variables. Recent advancements in robotics application suggest that overcoming this limitation is crucial for expanding the potential applications of multi-robot systems, allowing greater flexibility and adaptability in diverse settings [2], including environments which are shared with uncontrollable agents. Uncontrollable agents present a significant challenge when deploying multi-robot systems, as unpredictable conflicts require robots to respond in real time [3]. This poses a significant issue for MAPF-based coordination algorithms which typically rely on synchronous execution and generally cannot re-plan paths of individual agents separately.

The problem of execution delays is widely acknowledged within the MAPF community, and various approaches have been developed to enhance the robustness of MAPF solutions in this regard [4], [5], [6], [7]. However, there is little work that investigates the mitigation of delays itself. We hypothesize that even a partial reduction of conflicts with uncontrollable agents can help when deploying multi-robot fleet using centralized MAPF algorithms. While this work does not focus on improving the scalability of MAPF Algorithms, it may unlock applications which are, until now, not suited to deploy centrally coordinated multi-robot systems.

Neither humans nor artificial dynamic agents move randomly. In real-world environments (shopping malls, airports, warehouses or workshops, etc.) dynamic agents will generally follow well recognizable motion patterns. These motion patterns emerge because of the environment's topology, social norms and/or internal movement and behavior rules of the agents. This idea is formalized with the concept of Maps of Dynamics (MoDs) [8], which encode spatial or spatio-temporal motion patterns as features of the environment. MoDs capture structured and multimodal motion patterns and can be constructed from historical data or learned online. These patterns provide strong priors for predicting agent behavior into the future and over extended time horizons [9].

With the goal of improving the quality of paths generated by centralized MAPF algorithms in uncertain and dynamic environments, we propose Flow-Aware Multi-Agent Path Finding (FA-MAPF). The primary contribution in this context is the use of learned motion patterns of dynamic entities in a given environment, such as MoDs, in order to reduce conflicts between uncontrollable agents and the multi-robot system. To achieve this, we integrate multi-modal probability distributions, representing spatio-temporal motion patterns, into search-based state-of-the-art MAPF algorithms. Importantly, FA-MAPF strives for flow awareness implicitly at planning time and thus does not require ad-hoc replanning of specific agents or active sensing and detection of uncontrollable agents. Moreover, it retains most of the theoretical guarantees offered by centralized MAPF algorithms. We employ the idea of guidance, as proposed by Zhang et al. [10], and compute edge-weights using a cost function that matches an agent's possible actions with the local expected movement of a potential uncontrollable agent.

Using established MAPF benchmarks [11], we evaluate the effect of flow awareness on both the efficiency of MAPF algorithms and the number of conflicts with uncontrollable agents that can be expected. Furthermore, we assess the impact of flow awareness in lifelong MAPF settings in terms of runtime, throughput, and conflicts with uncontrollable agents, considering both benchmark maps and a real-world dataset. Our results indicate a clear trade-off between increased runtime and reduced conflicts with uncontrollable agents, while throughput remains largely unaffected.

[1]L. Heuer, Y. Zhu and M. Magnusson - Örebro University {lukas.heuer, yufei.zhu, martin.magnusson}@oru.se
[2]L. Palmieri, A. Rudenko and A. Mannucci - Bosch Research {luigi.palmieri, anna.mannucci}@de.bosch.com
[3]A. Rudenko - Technical University of Munich andrey.rudenko@tum.de
[4]S. Koenig - University of California, Irvine sven.koenig@uci.edu

II. Preliminaries and Related Work

In this section, we introduce the concepts that FA-MAPF is based on: (1) Multi-Agent Path Finding, (2) Maps of Dynamics and (3) Guidance, and overview the related work.

A. Multi-Agent Path Finding

We adopt their notion of a *classical* MAPF problem, as given by Stern et al. [11], which is defined as an undirected graph $G = (V, E)$ of vertices V and edges E, and a set of agents K, each associated with a start (s_k) and target vertex (t_k), i.e., $s_k, t_k \in V,\ \forall k \in K$. Time is discretized, and there exists a set of actions A from which each agent can perform one action per timestep. An action $a \in A$ either moves the agent along an edge from its current vertex $v \in V$ to a new vertex v' or causes the agent to wait in place, in which case $v = v'$. This relation can be formalized with a transition function $l : (V, A) \mapsto V$ such that $l(v, a) = v' : v, v' \in V, a \in A$. A path is defined as a sequence of actions $\psi = (a_0, \cdots, a_n)$. We call ψ a valid path for agent k if and only if, starting at vertex s_k, the execution of the actions in ψ results in being in vertex t_k. A solution to the MAPF problem is a set of valid, non-conflicting paths, one for each agent. This means that no agent occupies the same vertex or traverses the same edge at the same timestep. Ma et al. [12] propose a lifelong extension to the MAPF problem in which the agents are provided with a new goal anytime they reach their current one and the MAPF problem is re-instantiated after a set amount of timesteps.

Uncontrollable Agents in MAPF: Uncontrollable or external agents that can interact with and cause delays to individual agents are a notable problem for multi-agent systems [3]. While there exists a lot of work in this direction for single agent path planning, to our knowledge the only work to tackle this in the context of centrally coordinated multi-agent systems is by Bonalumi et al. [13]. They propose Multi-Agent Pickup and Delivery with External Agents (MAPD-EA), including an extension to the Token Passing (TP) Algorithm [12] called Token Passing with collision avoidance and replanning (TP-CA). Unfortunately, we were not able to obtain TP-CAs implementation, which is why we can not provide a direct comparison to our method.

While we consider a similar problem formulation, FA-MAPF does not require a well-formed MAPD problem and thus integrates naturally with state-of-the-art (lifelong) MAPF solvers like Explicit Estimation Conflict Based Search (EECBS) [14] and Rolling-Horizon Collision Resolution (RHCR) [15]. FA-MAPF also differs from TP-CA methodologically as we do not compute explicit occupancy likelihoods and transition probabilities, tied to the explicit detection of uncontrollable agents at planning time. Instead, FA-MAPF utilizes MoDs to incorporate local motion patterns into the environment model, guiding the planning process to reduce the probability of conflict with uncontrollable agents.

Robust MAPF: Several approaches have been developed to enhance the robustness of MAPF schedules. Atzmon et al. [5] show how a MAPF algorithm can be made resilient to delays spanning multiple timesteps. Phan et al. [7] use agent delays as a basis for the destroy heuristic in MAPF algorithms based on Large Neighbourhood Search (LNS). Hönig et al. [4] and Berndt et al. [6] achieve a certain degree of resilience to execution delays by capturing and enforcing the precedence relationships among robot actions of the original MAPF solution. This refinement ensures completeness and robust execution even in the presence of unknown or time-varying delays.

Unlike these works, our method does not focus on making MAPF solutions more resilient to execution delays. Instead, it aims to reduce interactions with uncontrollable agents that will cause delays, thereby reduce the required robustness of the MAPF solution in the first place. Importantly, FA-MAPF is a complementary to robust MAPF and not substitutive.

B. Maps of Dynamics

Maps of Dynamics (MoDs) encode spatial and spatio-temporal motion patterns as features of the environment. In this work, we exploit the Circular-Linear Flow Field (CLiFF) Map [16], an MoD representation that models local motion patterns as continuous, multimodal distributions over velocity. CLiFF-maps support both offline construction from past observations and online lifelong updates [17].

CLiFF-maps are built from observations of agent motion and assign a probability distribution over velocities to each spatial location. These velocities are represented as direction (θ) and speed (ρ), jointly as $\mathbf{u} = [\theta, \rho]^\mathsf{T}$, where $\theta \in [0, 2\pi)$ and $\rho \in \mathbb{R}^+$.

To capture the joint distribution of θ and ρ, CLiFF-maps use *Semi-Wrapped* Gaussian Mixture Models (SWGMMs), in which the circular variable θ is wrapped around the unit circle and ρ is linear. Each SWGMM is composed of Semi-Wrapped Normal Distributions (SWNDs), allowing the model to represent multimodal motion patterns.

An SWND $\mathcal{N}^{SW}_{\boldsymbol{\Sigma},\boldsymbol{\mu}}$ is formally defined as

$$\mathcal{N}^{SW}_{\boldsymbol{\Sigma},\boldsymbol{\mu}}(\mathbf{u}) = \sum_{m \in \mathbb{Z}} \mathcal{N}_{\boldsymbol{\Sigma},\boldsymbol{\mu}}([\theta, \rho]^\mathsf{T} + 2\pi[m, 0]^\mathsf{T}), \tag{1}$$

where $\boldsymbol{\Sigma}$ is the covariance matrix, $\boldsymbol{\mu}$ is the mean value of the directional velocity $(\theta, \rho)^\mathsf{T}$, and m is a winding number that enumerates overlapping parts of the wrapped θ variable.

An SWGMM is defined as a weighted sum of J SWNDs:

$$p(\mathbf{u}|\xi) = \sum_{j=1}^{J} \beta_j \mathcal{N}^{SW}_{\boldsymbol{\Sigma_j},\boldsymbol{\mu_j}}(\mathbf{u}), \tag{2}$$

where $\boldsymbol{\xi} = \{\xi_j = (\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j, \beta_j) | j \in \mathbb{Z}^+\}$ denotes the finite set of mixture parameters, and β_j denotes the mixing factor which satisfies $0 \leq \beta_j \leq 1$.

CLiFF-map have been applied in various domains, such as task and motion planning and trajectory prediction. Prior work by Palmieri et al. [18] and Swaminathan et al. [19] investigates how MoDs can be used for single robot motion planning. Furthermore, Liu et al. [20] demonstrate how human flow can be incorporated into hierarchical reinforcement learning for task planning. Lastly, Zhu et al. [9] explores how MoDs can be used to predict the motion trajectories of individual humans.

C. Guidance in MAPF

In FA-MAPF, we integrate learned motion patterns into MAPF algorithms by adapting the transition costs in the MAPF graph G to capture these patterns. This approach, recently formalized and termed *guidance* by Zhang et al. [10] has been comprehensively surveyed in prior work, including [21], [22], [10], [23], [24], [25]. While most existing approaches apply guidance primarily to improve the runtime efficiency or throughput of multi-agent systems, FA-MAPF distinguishes itself by focusing on minimizing interactions with uncontrollable agents. We achieve this by modeling their motion patterns and using these patterns to direct the MAPF algorithm.

To formalize the integration of guidance into FA-MAPF, we adopt the framework of a guidance graph. Specifically, we assume a directed guidance graph $G_g = (V_g, E_g, \boldsymbol{\omega})$, as described by Zhang et al. [10], where the vertex set is identical to that of the original MAPF graph (i.e., $V = V_g$) and E_g includes edges representing all possible transitions available to a MAPF agent in G. The vector $\boldsymbol{\omega}$ encodes the weights assigned to each edge, with ω_e denoting the specific cost associated with edge $e \in E_g$. This formalism allows us to encapsulate motion patterns into the computation of $\boldsymbol{\omega}$ as explained in Section III.

III. Flow-Aware Multi-Agent Path Finding

This section presents the core contribution of our paper: a method for accounting of learned motion patterns of uncontrollable agents when solving a MAPF problem.

A. Problem Formulation

We consider a potentially lifelong MAPF problem as described in Section II-A to coordinate a multi-robot system. We consider uncontrollable agents (UAs) that operate in the same environment as the agents controlled by the MAPF algorithm, i.e. MAPF agents. A UA is defined through its trajectory $\boldsymbol{\zeta} = (\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_\zeta)$ with Cartesian positions $\mathbf{x}_i \in \mathbb{R}^2$.

Assume two sets of trajectories of UAs: $H = (\boldsymbol{\zeta}_0, \boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_H)$ as an observed dataset of trajectories in the operational environment and $F = (\boldsymbol{\zeta}_0, \boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_F)$ as the trajectories of the UAs that are present when the MAPF solution is executed. Importantly, we assume knowledge about H and consider F to be unknown, even at planning time.

A conflict between a MAPF agent and a UA occurs if, at any time, the Euclidean distance between them is smaller than the sum of their radii. We now aim to find a solution to the MAPF problem that minimizes conflicts between the paths in the MAPF solution and the trajectories in F.

B. Method

Consider an environment in Euclidean space W, represented by a graph $G = (V, E)$ (as described in Section II-A), a transformation $T_{W \mapsto V}$, and the set of possible MAPF agent actions A. A dataset H of observed trajectories of uncontrollable agents is used to construct the MoD M, which encodes relevant motion patterns in environment W.

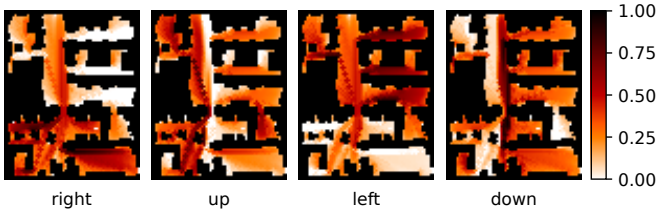


Fig. 1: The cost for an edge transition for every location in the *den312d*-map, for each move action respectively. It is computed using the MoD in Figure 3 and shows how FA-MAPF translates motion patterns into a semantic cost-landscape for the MAPF algorithm.

The core idea of FA-MAPF is to assign a cost-value to account for the alignment between an agent's action $a \in A$ with the motion pattern $M(v)$, at a given location. Formally, let $q : (W, A, M) \mapsto \mathbb{R}_{\geq 0}$ be a function which maps a position in the environment W, an action from the possible set of actions A, and an MoD M, to a cost-value. Using $T_{W \mapsto V}$ to map positions to vertices, we can query the MoD at the position of a vertex $v \in V$ to obtain $M(v)$, and define the flow cost $g_f(a, M(v))$ whose explicit calculation is explained in Section III-C. Using g_f we then compute the edge weights in a guidance graph $G_g = (V_g, E_g, \boldsymbol{\omega})$ which FA-MAPF uses to plan the paths of the individual MAPF agents. The individual edge weights are computed as

$$\omega_e = g_s + g_f\left(a, M\left(v\right)\right), \ \forall \ e \in E_g, \tag{3}$$

where v is the source vertex of e, a is the action taken to follow the edge e, $M(v)$ is the SWGMM obtained from the CLiFF-map at the location v, and g_s is the step-cost defined by the MAPF algorithm cost-function. Fig. 1 illustrates an example guidance graph on a map used in our experiments.

Because FA-MAPF does not require real-time information at planning time, the guidance graph and an admissible heuristic (including the flow cost) can be pre-computed by finding shortest paths in G_g between all pairs of vertices.

C. Flow Awareness

To compute g_f, the flow cost, we use the Mahalanobis distance which measures the distance between a multivariate Gaussian distribution and a vector [26]. The Mahalanobis distance can be extended to Gaussian mixtures by computing the weighted sum over the mixture components. A similar formulation has been proposed for sampling-based planning [19].

Using this approach to measure the distance between an action a and an SWGMM $M(v)$, obtained from a CLiFF map, requires a to match the vector space of $M(v)$. That is, a should also be a velocity vector $\mathbf{u} = [\theta, \rho]^\mathsf{T}$. When implementing a MAPF algorithm on a 4-connected grid, we consider $A = \{a_{\text{right}}, a_{\text{up}}, a_{\text{left}}, a_{\text{down}}, a_{\text{wait}}\}$ as the set of possible actions, and a direct mapping $\alpha : A \mapsto A'$. With

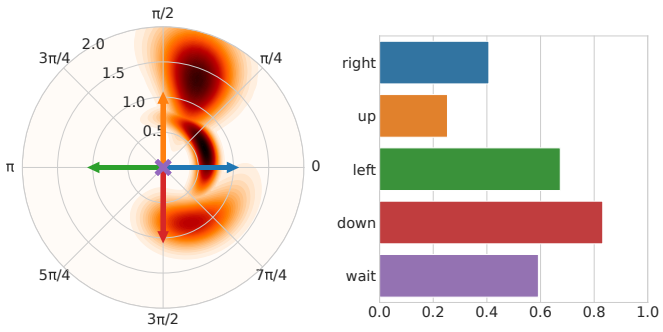


Fig. 2: **Left:** A visualization of an SWGMM, obtained from an MoD, that represents the likelihood of UA movement for a specific location in a map. The colored arrows represent the set of possible actions of a MAPF agent. **Right:** Flow cost calculated using Eq. 4 representing the *distance* from an action to the SWGMM.

$A' = \{[0,1]^\mathsf{T}, [\frac{\pi}{2},1]^\mathsf{T}, [\pi,1]^\mathsf{T}, [\frac{3\cdot\pi}{2},1]^\mathsf{T}, [-,0]^\mathsf{T}\}$, we can use $a \in A$ and $\mathbf{a}' \in A'$ interchangeably.

Given an SWGMM as described in Equation 2, and an action $\mathbf{a}' \in A'$, we now compute

$$g_f\left(a, M\left(v\right)\right) := \log\gamma \sum_{j=0}^{J} \left(\beta_j \sqrt{\left(\boldsymbol{\mu}_j - \mathbf{a}'\right)^\mathsf{T} \boldsymbol{\Sigma}_j^{-1} \left(\boldsymbol{\mu}_j - \mathbf{a}'\right)}\right), \quad (4)$$

where γ is the number of UA observations made at location v. Thus the sum represents the actions alignment with the underlying SWGMM and the logarithmic term scales that value with likelihood that an UA will be encountered in this location in the first place. Importantly, we compute the subtraction between two angles (i.e. $\boldsymbol{\mu}_\theta$ and $\mathbf{a}'_\theta$) as the shortest positive angular distance around the unit circle. After computing g_f for all transitions in the map, we employ min-max normalization to scale the cost values to the range $[0,1]$.

Fig. 2 exemplary illustrates a SWGMM; the way it relates to the actions $\mathbf{a}' \in A'$; and the values of g_f for the corresponding actions $a \in A$.

Implementation assumptions: Applying flow cost in MAPF is based on two key assumptions.

First, as MAPF algorithms generally work with unit time, where one action takes one timestep, we assume that agents move with a constant speed. In accordance with the definition of A', we set the agent's speed to $1\frac{\text{m}}{\text{s}}$, which is slightly below the average human walking speed of $1.42\frac{\text{m}}{\text{s}}$ [27].

Second, the wait actions a_{wait} lacks a defined movement angle, preventing direct computation of g_f for this action. To resolve this, we define $g_f(a_{\text{wait}})$ for each $M(v)$ as the mean value of g_f the other four actions, with ρ set to zero.

D. Completeness and Optimality

The completeness and (sub-)optimality properties of FA-MAPF depend on the underlying MAPF algorithm used. Specifically, when employing a complete and (sub-)optimal search-based MAPF algorithm, such as ECBS [28], FA-MAPF inherits these properties. Completeness is preserved because FA-MAPF only adds a finite positive value to the cost of each edge transition in the graph. When $g_f \in [0,1]$, Equation 3 shows that $g(v) < g(v')\ \forall\ v, v' \in V$. In other words, all edge costs remain positive and the cost strictly increases as the agent progresses, ensuring that the search terminates and all solutions are eventually found, that is, the search remains exhaustive and complete [29].

If the underlying MAPF algorithm uses an admissible heuristic, FA-MAPF also maintains the (sub-)optimality guarantee (up to the suboptimality bound for algorithms like ECBS), though now with respect to the total cost on the guidance graph G_g which includes the adapted edge weights. However, with a simple assumption, this guarantee can be extended to only the path length of the solution.

Bounded (sub-)optimality by the optimal path length: The cost of any path ψ in a solution returned by a ω_1-bounded suboptimal FA-MAPF algorithm, as described in Section III, is composed of two terms: the path-length cost term $\sum_\psi g_s$ and the flow cost term $\sum_\psi g_f$ (we generally omit function arguments here for better readability). Consider a path ψ^*, with cost $\sum_{\psi^*} g_s$, to be optimal. We prove that the cost of ψ, $\sum_\psi g_s + \sum_\psi g_f$, is bounded suboptimal with respect to $\sum_{\psi^*} g_s$ under a simple condition.

Remark. The path ψ is ω_1-bounded suboptimal w.r.t. the cost function $g_s + g_f$. The path ψ' is optimal w.r.t. the cost function $g_s + g_f$. The path ψ^* is optimal w.r.t. the cost function g_s.

Lemma 1. *If g_f is positive finite in the entire search space such that $\max_{a\in A, v\in V}(g_f(a, M(v))) \leq \omega_2\ g_s$, then the cost of a ω_1-bounded suboptimal path ψ, $\sum_\psi g_s + \sum_\psi g_f$, has the upper bound $(\omega_1 + \omega_1\ \omega_2)\sum_{\psi^*} g_s$ with $\omega_1 \geq 1$ and $\omega_2 \geq 0$.*

Proof. Since ψ is ω_1 bounded suboptimal, the following relation between ψ and an unknown optimal path ψ' upholds:

$$\textstyle\sum_\psi g_s + \sum_\psi g_f \leq \omega_1 \left(\sum_{\psi'} g_s + \sum_{\psi'} g_f\right). \quad (5)$$

From $\max_{a\in A, v\in V}(g_f(a, M(v))) \leq \omega_2\ g_s$, we can derive the following equation, which is valid for any path $\psi^\#$:

$$\textstyle\sum_{\psi^\#} g_f \leq \omega_2 \sum_{\psi^\#} g_s. \quad (6)$$

This also includes ψ^*, so we can calculate the potential flow cost along ψ^*, and obtain

$$\textstyle\sum_{\psi^*} g_s + \sum_{\psi^*} g_f \leq \sum_{\psi^*} g_s + \omega_2 \sum_{\psi^*} g_s. \quad (7)$$

Multiplying with ω_1 and rearranging yields:

$$\textstyle\omega_1 \left(\sum_{\psi^*} g_s + \sum_{\psi^*} g_f\right) \leq (\omega_1 + \omega_1\ \omega_2) \sum_{\psi^*} g_s. \quad (8)$$

As ψ' is defined as the optimal for the cost function $g_s + g_f$, we can then write

$$\textstyle\omega_1 \left(\sum_{\psi'} g_s + \sum_{\psi'} g_f\right) \leq \omega_1 \left(\sum_{\psi^*} g_s + \sum_{\psi^*} g_f\right). \quad (9)$$

Finally, using the transitivity of Equations 5, 9, and 8 we obtain:

$$\textstyle\sum_\psi g_s + \sum_\psi g_f \leq (\omega_1 + \omega_1\ \omega_2) \sum_{\psi^*} g_s. \quad (10)$$

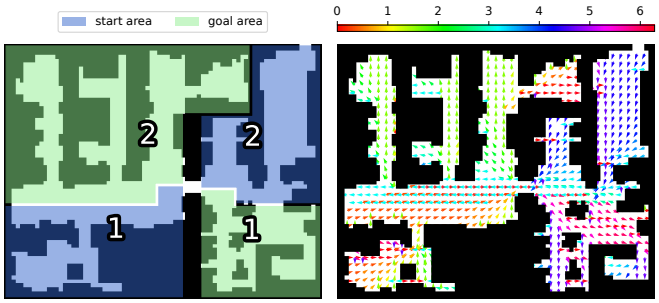


Fig. 3: **Left:** It shows the areas from which the start and goal points are sampled for *directed* UA movement. The numbers indicate the corresponding areas. **Right:** The corresponding CLiFF-map of dynamics, with the arrows representing the means of the SWGMM mixture components, and color indicates their direction in radians.

Thus, the cost of the ω_1-bounded suboptimal path in the solution of a FA-MAPF problem has the upper bound $(\omega_1 + \omega_1\omega_2) \sum_{\psi^*} g_s$. Summing over all paths extends this to the entire FA-MAPF solution. □

IV. Evaluation

As our main objective is to reduce conflicts with uncontrollable agents (UAs), we focus our evaluation on how much such reduction we can expect on different maps and different types of motion, and how much it impacts computation runtime and task efficiency.

Experiments: In our first experiment, we use EECBS, with a suboptimality factor of 1.2, to solve a standard MAPF problem. We specifically want to investigate the computational efficiency of FA-MAPF and use maps from the established benchmark by Stern et al. [11].

The second experiment solves a lifelong MAPF problem using RHCR with ECBS, with a suboptimality factor of 1.5, as a high-level solver and SIPP [30] as a low-level solver. We refer to the flow-aware version of RHCR as FA-RHCR. This experiment focuses on showing how much reduction in conflicts we can expect as well as the tradeoff in terms of runtime and efficiency. If not stated differently, RHCR is run with a simulation time, replanning window and conflict resolution horizon of 2000, 20, and 40 timesteps respectively. The tasks for each agent are chosen randomly and to ensure reproducibility across methods, we use a predetermined task queue per map and agent.

All experiments are run on a Intel i7-12700K CPU with 32GB of RAM.

Metrics: In our experiment we evaluate several metrics. *solved* is specific to the first experiment (Fig. 4) and refers to the percentage of instances solved out of the 25 scenarios contained in the benchmark, within a time-limit of 5 seconds. Since the UAs are not directly considered at planning time they can not cause a MAPF problem instance to become infeasible. Therefore a reduction in the *solved* metric is always attributed to a worse runtime-performance and not collisions between UAs and MAPF agents.

We evaluate algorithm *runtime* directly for both experiments. In the first experiment it refers to the time EECBS takes to find a solution. In the second experiment it refers to the average solving time per RHCR iteration. To asses FA-MAPF's impact on the efficiency we provide *throughput* as the average number of tasks completed per timestep. *UA Conflicts* is the average number of conflicts between MAPF agents and UAs per timestep. As described in Section III-A a conflict occurs if, at any one time, the Euclidean distance between a MAPF agent and a UA is less than the sum of their radii (each radii set to 0.3 m).

Maps: We perform experiments on 8 different maps, namely *empty-32-32*, *random-32-32-10*, *random-32-32-20*, *maze-32-32-2*, *maze-32-32-4*, *ht_chantry*, *den312d* and *warehouse-10-20-10-2-1* [11]. In addition we run an experiment on the ATC dataset map [31] which allows us to test our approach against real-world recorded human motion data. On all maps, one grid-cell corresponds to 1×1 meters.

Uncontrollable Agents: We simulate UAs on the benchmark maps using SIPP with a branching factor of 5 to allow diagonal movement. The UAs move at a constant speed of $1\frac{\text{m}}{\text{s}}$ and to not consider inter-agent collisions.

We hypothesize that the way UAs move affects how well MoDs can help in avoiding conflicts. For instance agents moving randomly in the environment are much harder to synthesize into specific movement patterns then agents that stream between to specific map regions. We consider 3 types of UA movement, defined by how the start and goal positions of the UAs are chosen: *random* refers to the start and goal positions being chosen randomly on the map. *directed* refers to the start and goal location being sampled from dedicated areas such that clear movement patterns are enforced. *speed* refers to the start and goal locations being sampled from dedicated areas, with some areas being associated with twice the movement speed of others. For each map and movement type we sample 10,000 trajectories, which are used to compute the MoDs.

For the first experiment, on each map, we consider 100 UAs starting at $t = 0$ and moving towards their respective goal states, vanishing upon arrival. For the second experiment we consider one UA to appear every timestep, follows a trajectory based on the movement type, and vanish at its goal location. This ensures there are UAs moving continuously through the environment.

Fig. 3 shows the start and goal areas on the *den312d*-map for *directed* UA movement, along with its CLiFF-map visualization.

V. Results and Discussion

In the following we present our experimental results.

Experiment 1: Fig. 4 shows the results obtained from the first experiment, where we evaluate the scalability of FA-MAPF by solving one-shot EECBS on the different benchmark maps. MAPF without flow cost is more computationally efficient and typically maintains 100% solved rate (within 5 s) for larger numbers of agents compared to FA-MAPF. Equivalently, the runtime starts to increase later.

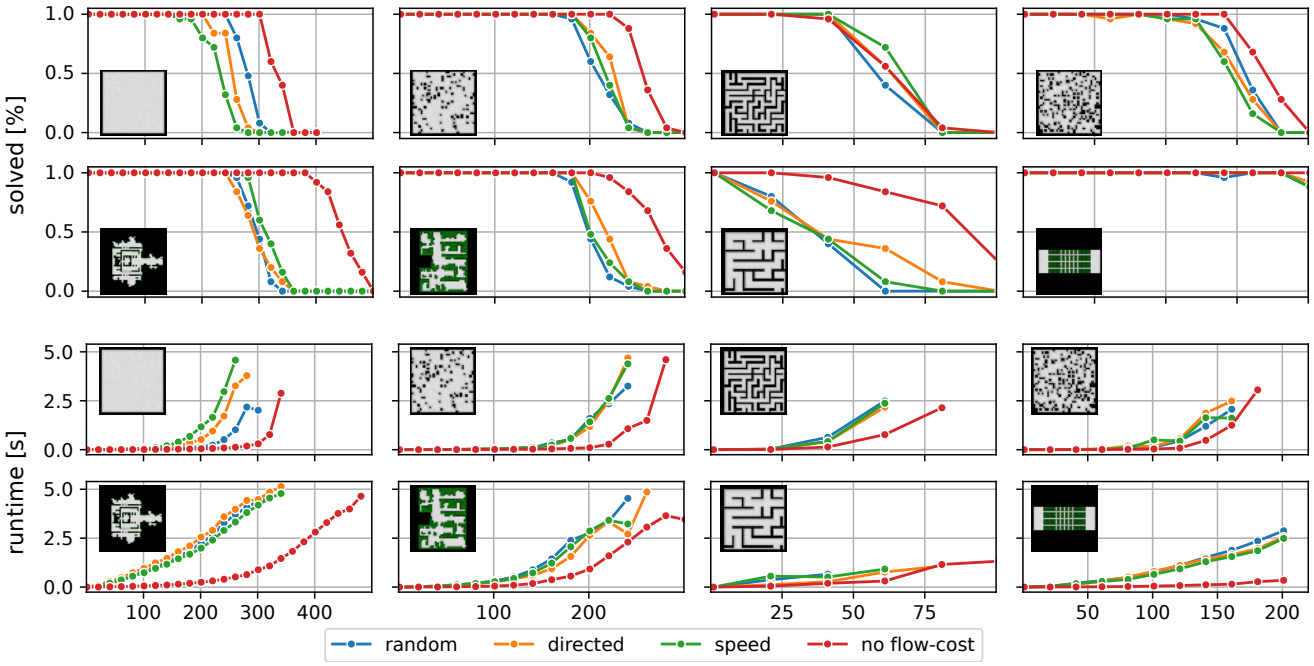


Fig. 4: Result of one-shot EECBS on the different benchmark maps (see Table I for the map names), for different UA movement types (as described in Section IV). x-axis is the number of MAPF agents. y-axis shows either the percentage (**top**) of instances solved (out of 25) or the average runtime (**bottom**). The runtime cutoff is 5 seconds.

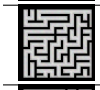
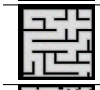
	map name	MAPF method movement type	UA Conflicts RHCR	 FA-RHCR	Runtime [s] RHCR	 FA-RHCR	Throughput RHCR	 FA-RHCR
	den312d # agents: 200	directed random speed	23.47 ± 3.04 19.97 ± 2.37 21.55 ± 3.17	**10.52 ± 1.43** **15.30 ± 1.97** **15.44 ± 2.31**	**1.68 ± 1.74** 1.68 ± 1.74 **1.68 ± 1.74**	3.22 ± 2.95 **1.07 ± 0.64** 7.47 ± 3.82	2.73 2.73 **2.73**	**2.89** **2.83** 2.57
	empty-32-32 # agents: 300	directed random speed	10.65 ± 0.92 13.93 ± 1.02 9.50 ± 0.82	**6.82 ± 0.81** **11.99 ± 0.97** **6.52 ± 0.70**	**0.37 ± 0.07** **0.38 ± 0.08** **0.37 ± 0.07**	1.87 ± 0.59 0.52 ± 0.05 3.73 ± 1.79	**8.28** 8.28 **8.28**	8.27 **8.44** 8.19
	ht_chantry # agents: 500	directed random speed	21.15 ± 2.86 22.19 ± 2.67 21.17 ± 2.65	**9.77 ± 1.32** **16.07 ± 1.88** **10.14 ± 1.25**	**0.60 ± 0.34** **0.59 ± 0.34** **0.61 ± 0.35**	5.28 ± 2.06 2.95 ± 0.62 6.81 ± 1.76	4.67 4.67 **4.67**	4.67 **4.70** 4.63
	maze-32-32-2 # agents: 30	directed random speed	8.99 ± 1.25 7.95 ± 1.02 9.09 ± 1.85	**8.55 ± 1.23** **7.59 ± 1.05** **8.50 ± 1.73**	**0.01 ± 0.01** **0.01 ± 0.01** **0.01 ± 0.01**	1.30 ± 0.84 0.09 ± 0.04 5.20 ± 4.62	0.40 0.40 0.40	0.40 0.40 0.40
	maze-32-32-4 # agents: 30	directed random speed	5.80 ± 1.46 5.69 ± 1.24 5.76 ± 1.85	**4.76 ± 1.43** **4.75 ± 1.19** **4.87 ± 1.27**	**0.26 ± 2.00** **0.26 ± 2.00** **0.26 ± 2.00**	9.19 ± 7.97 0.78 ± 2.79 10.47 ± 7.67	0.43 **0.43** **0.43**	0.43 0.42 0.42
	random-32-32-10 # agents: 300	directed random speed	12.19 ± 0.82 15.43 ± 1.12 10.85 ± 0.89	**8.78 ± 0.78** **14.71 ± 1.10** **8.10 ± 0.86**	**1.08 ± 0.29** **1.09 ± 0.29** **1.08 ± 0.29**	7.51 ± 4.23 1.46 ± 0.90 10.89 ± 6.16	**7.27** 7.27 **7.27**	7.09 **7.30** 6.28
	random-32-32-20 # agents: 200	directed random speed	10.09 ± 1.02 12.66 ± 1.08 9.15 ± 0.97	**6.74 ± 0.72** **12.21 ± 1.02** **6.99 ± 0.85**	**0.69 ± 0.31** 0.69 ± 0.31 **0.69 ± 0.31**	1.96 ± 0.93 **0.65 ± 0.26** 2.45 ± 2.14	4.62 **4.62** 4.62	**4.63** 4.57 **4.64**

TABLE I: Results from running (FA-)RHCR on different benchmark maps for a specific number of MAPF agents. The results compare the relevant metrics for RHCR with and without flow-awareness. Mean and standard deviation are computed across the different RHCR iterations.

However, the extent to which FA-MAPF scales worse, seems to depend on the type of map. That *den312d* and *ht_chantry* scale more poorly than the random maps suggests that on maps with more *structure* the guiding effect of FA-MAPF is stronger. The movement type of the UAs does not seem to have a general impact on the scalability in this case.

Experiment 2: Table I presents results from the second experiment using RHCR. The results show that FA-MAPF reduces *UA conflicts* by up to 55% compared to the baseline, in exchange for an increase in runtime. As in Experiment 1,

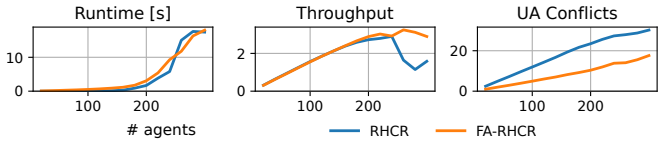


Fig. 5: Results of running RHCR on the *den312d*-map. x-axis is the number of MAPF agents, y-axis the respective metric value. UA movement type is *directed*.

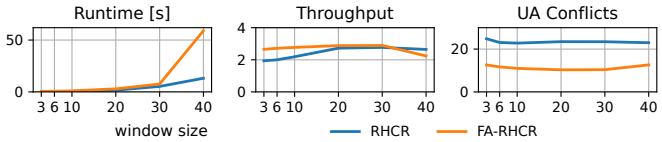


Fig. 6: Results of running RHCR with 200 agents on the *den312d*-map. x-axis is the replanning window size of RHCR, y-axis the respective metric value. UA movement type is *directed*.

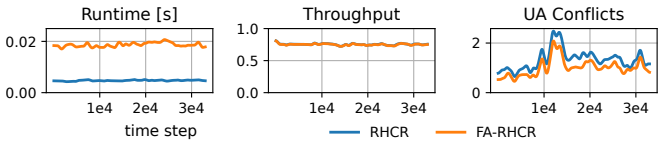


Fig. 7: Results of running RHCR on the ATC-map. x-axis is the number of steps, y-axis the respective metric value. RHCR runs for 33000 timesteps with 40 MAPF agents.

the increased *runtime* is due to the non-uniform transition cost in the guidance graph, which causes the low-level search algorithm to expand more nodes and thus taking longer to find the final (sub-)optimal path for a MAPF agent. While *throughput* is generally an important metric, it is not notably effected by using FA-MAPF. Comparing the results across the different maps suggests that FA-MAPF excels on maps with a room-like structure (i.e. *den312d* and *ht_chantry*) but is less suited for maze-like environments. We believe this is because, on maps with rooms, flow patterns emerge more clearly and it is easier for the MAPF agents to adapt their paths accordingly. In contrast, maze-like maps do not provide sufficient room to maneuver, limiting ability of MAPF agents to adjust to the motion patterns of the UAs.

We want to highlight that the *runtime* in Fig. 4 and Table I are not directly comparable. As the first experiment uses one-shot EECBS and reports the number of unsolved instances, unsolved instances where the runtime limit is reached do not contribute to the metric calculation. This means that the reported values represent the runtime for the instances where a solution was eventually found within 5 seconds. In contrast to EECBS, RHCR handles infeasible iterations differently. Valid paths for the agents are still used, and only ones that are conflicting are frozen for the respective planning window. Thus infeasible iterations in RHCR are considered for the metric calculation, including the runtime which contributes to these iterations with its maximum value.

Additional Evaluations: Fig. 5 and 6 show results on the *den312d* map with *directed* UA movement, varying the

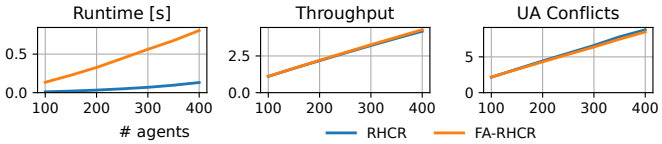


Fig. 8: Results of running RHCR on the *warehouse-10-20-10-2-1*-map. x-axis is the number of agents, y-axis the respective metric value. UA movement type is *directed*.

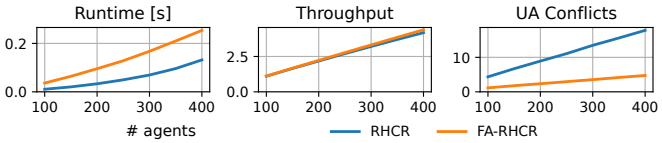


Fig. 9: Results of running RHCR on the *warehouse-10-20-10-2-1*-map. x-axis is the number of agents, y-axis the respective metric value. UA movement type is *directed* but enforces highway patters [32].

number of agents and the replanning horizon. It is worth noting that, for larger numbers of agents, FA-MAPF seems to dampen the drop-off in *throughput*. We cannot make a definitive claim as to why this is, but it suggest that FA-MAPF produces more feasible paths and live agents even in situations where a full solution can not be found in time.

Importantly, we also evaluate how FA-MAPF works with real-world human motion data. We run RHCR with 40 MAPF agents for 33000 timesteps, i.e. 9 hours, on a grid-map representing the ATC shopping mall. For evaluation we replay the first day of the ATC-dataset [31] to compute conflicts of the MAPF solutions with the trajectories in the dataset. Fig. 7 shows that in real-world scenarios, FA-MAPF is able to consistently reduce *UA conflicts* by a notable margin, at the cost of an increase in runtime.

We also investigate how FA-MAPF works in warehouse-like environments. In order to do so, we consider two settings, whose evaluation confirms the strengths and weaknesses of FA-MAPF we identified in the other experiments. Firstly, Fig. 8 shows results where UAs use the shortest path to their goal without restriction. This causes corridors to be traversed in both directions and no clear flow-pattern to emerge, hence FA-MAPF is unable to improve *UA conflicts* in this case. In a second case shown in Fig. 9, we impose highway patterns [32], only allowing the UAs to traverse corridors in one direction. The allowed directions alternate from top to bottom and left to right. The results show that FA-MAPF is successfully able to infer the highway patterns from the MoD, significantly reducing *UA conflicts.*

VI. Conclusion and Future Research

We propose a novel method to incorporate statistical learned information about motion patterns of uncontrollable agents into centralized MAPF. Our evaluation demonstrates that FA-MAPF significantly reduces conflicts with uncontrollable agents, reducing the need for low-level collision avoidance mechanisms to repair local paths.

The concept of FA-MAPF establishes a foundation for several research directions: **(1)** Integrating flow awareness into large-scale MAPF algorithms such as PIBT [33] or LaCAM [34]. **(2)** Investigating the impact of conflicts with UA on efficiency metrics of a coordinated multi-robot system. **(3)** Extending FA-MAPF to incorperate time-dependent Maps of Dynamics [17]

In conclusion, this paper lays groundwork for transitioning multi-robot systems from fully controlled spaces with no external disturbances to more challenging and interactive environments.

REFERENCES

[1] A. Sharma, "Mobile robots on the march – 53,000 warehouses & factories will have deployed amrs & agvs by end of 2025," Available at https://www.roboticstomorrow.com/article/2021/10/mobile-robots-on-the-march-53000-warehouses-factories-will-have-deployed-amrs-agvs-by-end-of-2025/17627 (2025/05/27), 2021.

[2] Proteus, "I'm amazon's first autonomous robot. follow me around on my typical workday (watercooler break included)." Available at https://www.aboutamazon.com/news/operations/amazon-robotics-autonomous-robot-proteus-warehouse-packages (2025/04/10), 2024.

[3] J.-M. Alkazzi and K. Okumura, "A comprehensive review on leveraging machine learning for multi-agent path finding," *IEEE Access*, 2024.

[4] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Persistent and robust execution of mapf schedules in warehouses," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1125–1131, 2019.

[5] D. Atzmon, R. Stern, A. Felner, N. R. Sturtevant, and S. Koenig, "Probabilistic robust multi-agent path finding," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 30, 2020, pp. 29–37.

[6] A. Berndt, N. Van Duijkeren, L. Palmieri, A. Kleiner, and T. Keviczky, "Receding horizon re-ordering of multi-agent execution schedules," *IEEE Transactions on Robotics*, 2023.

[7] T. Phan, B. Zhang, S.-H. Chan, and S. Koenig, "Anytime multi-agent path finding with an adaptive delay-based heuristic," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 2025, pp. 23 286–23 294.

[8] T. P. Kucner, M. Magnusson, S. Mghames, L. Palmieri, F. Verdoja, C. S. Swaminathan, T. Krajník, E. Schaffernicht, N. Bellotto, M. Hanheide, *et al.*, "Survey of maps of dynamics for mobile robots," *The International Journal of Robotics Research*, vol. 42, no. 11, pp. 977–1006, 2023.

[9] Y. Zhu, A. Rudenko, T. P. Kucner, L. Palmieri, K. O. Arras, A. J. Lilienthal, and M. Magnusson, "Cliff-lhmp: Using spatial dynamics patterns for long-term human motion prediction," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 3795–3802.

[10] Y. Zhang, H. Jiang, V. Bhatt, S. Nikolaidis, and J. Li, "Guidance graph optimization for lifelong multi-agent path finding," in *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, K. Larson, Ed. International Joint Conferences on Artificial Intelligence Organization, 8 2024, pp. 311–320, main Track. [Online]. Available: https://doi.org/10.24963/ijcai.2024/35

[11] R. Stern, N. R. Sturtevant, A. Felner, S. Koenig, H. Ma, T. T. Walker, J. Li, D. Atzmon, L. Cohen, T. K. S. Kumar, E. Boyarski, and R. Bartak, "Multi-agent pathfinding: Definitions, variants, and benchmarks," *Symposium on Combinatorial Search (SoCS)*, pp. 151–158, 2019.

[12] H. Ma, J. Li, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," in *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2017, p. 837–845.

[13] L. Bonalumi, B. Flammini, D. Azzalini, and F. Amigoni, "Multi-agent pickup and delivery with external agents," *Robotics and Autonomous Systems*, vol. 191, p. 105000, 2025.

[14] J. Li, W. Ruml, and S. Koenig, "Eecbs: A bounded-suboptimal search for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, 2021, pp. 12 353–12 362.

[15] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. S. Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 11 272–11 281.

[16] T. P. Kucner, M. Magnusson, E. Schaffernicht, V. H. Bennetts, and A. J. Lilienthal, "Enabling flow awareness for mobile robots in partially observable environments," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1093–1100, 2017.

[17] Y. Zhu, A. Rudenko, L. Palmieri, L. Heuer, A. J. Lilienthal, and M. Magnusson, "Fast online learning of cliff-maps in changing environments," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 10 424–10 431.

[18] L. Palmieri, T. P. Kucner, M. Magnusson, A. J. Lilienthal, and K. O. Arras, "Kinodynamic motion planning on gaussian mixture fields," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 6176–6181.

[19] C. S. Swaminathan, T. P. Kucner, M. Magnusson, L. Palmieri, and A. J. Lilienthal, "Down the cliff: Flow-aware trajectory planning under motion pattern uncertainty," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7403–7409.

[20] Y. Liu, L. Palmieri, I. Georgievski, and M. Aiello, "Human-flow-aware long-term mobile robot task planning based on hierarchical reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 8, pp. 4068–4075, 2023.

[21] H. Zang, Y. Zhang, H. Jiang, Z. Chen, D. Harabor, P. J. Stuckey, and J. Li, "Online guidance graph optimization for lifelong multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, 2025, pp. 14 726–14 735.

[22] Z. Chen, D. Harabor, J. Li, and P. J. Stuckey, "Traffic flow optimisation for lifelong multi-agent path finding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 2024, pp. 20 674–20 682.

[23] L. Cohen, T. Uras, T. S. Kumar, H. Xu, N. Ayanian, and S. Koenig, "Improved solvers for bounded-suboptimal multi-agent path finding." in *IJCAI*, 2016, pp. 3067–3074.

[24] M. R. J. M. R. Jansen and N. Sturtevant, "Direction maps for cooperative pathfinding," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 4, 2008, pp. 185–190.

[25] K.-H. C. Wang, A. Botea, *et al.*, "Fast and memory-efficient multi-agent pathfinding." in *ICAPS*, vol. 18, 2008, pp. 380–387.

[26] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[27] R. C. Browning, E. A. Baker, J. A. Herron, and R. Kram, "Effects of obesity and sex on the energetic cost and preferred speed of walking," *Journal of applied physiology*, vol. 100, no. 2, pp. 390–398, 2006.

[28] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proceedings of the international symposium on combinatorial Search*, vol. 5, no. 1, 2014, pp. 19–27.

[29] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Pearson, 2016.

[30] M. Phillips and M. Likhachev, "Sipp: Safe interval path planning for dynamic environments," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, 2011, pp. 5628–5635.

[31] D. Brščić, T. Kanda, T. Ikeda, and T. Miyashita, "Person tracking in large public spaces using 3-d range sensors," *IEEE Transactions on Human-Machine Systems*, vol. 43, no. 6, pp. 522–534, 2013.

[32] M.-F. Li and M. Sun, "The study of highway for lifelong multi-agent path finding," *arXiv preprint arXiv:2304.04217*, 2023.

[33] K. Okumura, M. Machida, X. Défago, and Y. Tamura, "Priority inheritance with backtracking for iterative multi-agent path finding," *Artificial Intelligence*, vol. 310, p. 103752, 2022.

[34] K. Okumura, "Lacam: Search-based algorithm for quick multi-agent pathfinding," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 10, 2023, pp. 11 655–11 662.