Multi-Agent Terraforming: Efficient Multi-Agent Path Finding via Environment Manipulation

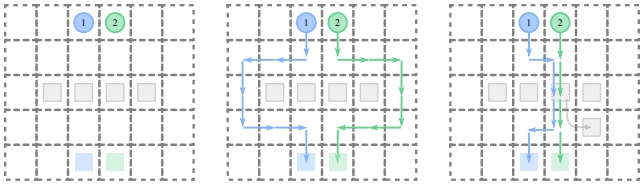
David Vainshtein and Oren Salzman
Technion – Israel Institute of Technology
dudiwa@campus.technion.ac.il, osalzman@cs.technion.ac.il

Abstract

Planning collision-free paths for multiple agents operating in close proximity has a myriad of applications ranging from smart warehouses to route planning for airport taxiways. This problem, known as the Multi-Agent Path-Finding (MAPF) problem, is highly relevant to real-world applications in automation and robotics, and has attracted significant research in recent years. While in many applications, the robots are tasked with transporting objects and thus have the means to move obstacles, common formulations of the problem prohibit agents from moving obstacles en-route to a task. This often causes agents to take long detours to avoid obstacles instead of simply moving them to clear a path. In this work we present multi-agent terraforming, a novel extension of the MAPF problem that can exploit the fact that the system contains movable obstacles. We build upon leading MAPF solvers and propose an efficient method to solve the multi-agent terraforming problem in a manner that is both complete and optimal. We evaluate our method on scenarios inspired by smart warehouses (such as those of Amazon) and demonstrate that, compared to the classical MAPF formulation, the extra flexibility provided by terraforming facilitates a notable improvement of solution quality.

Introduction

The MAPF problem (Stern et al. 2019) calls for planning collision-free paths for a set of agents from given start to goal locations. A motivating example is the management of smart warehouses, in which agents carry shelves to and from packing stations (Wurman, D'Andrea, and Mountz 2008). The standard MAPF approach perceives idle shelves as obstacles that often elicit winding paths and funnel the agents into congested bottlenecks. However in many domains, such as our motivating smart warehouse example, agents have the capability to move obstacles and indeed do so as part of their task. Our key insight is that this capability can be used to increase the system's throughput which is our ultimate goal. Several relevant works allow obstacles to be displaced or removed by a robot, with a recent variant called C-MAPF (Bellusci, Basilico, and Amigoni 2020), venturing to reconfigure the environment itself to better serve a preferable solution. These works demonstrate the advantage of adjusting obstacles to open up shortcuts in the environment.



(a) MAPF task (b) Static obstacles (c) Terraforming

Figure 1: An obstacle is displaced to reduce flowtime.

In this work we set the ground to allow such capabilities. As a simplifying assumption, here we do not require the obstacles to be moved by agents and assume that they are "self propelled". Future work will remove this assumption and incorporate the additional cost of bringing agents to the movable obstacles.

Classical MAPF Given a set of agents $\mathcal{A} = \{a_1, \ldots, a_n\}$ inhabiting an undirected graph $G = (V, E)$, the MAPF problem assigns each agent with start and goal locations (vertices). Let π_i be a single-agent path where $\pi_i(t)$ corresponds to the position of agent a_i at timestep t. We define the single agent *service cost* as the number of timesteps $|\pi_i|$ until the goal is reached. Agents must avoid obstacles that obstruct locations and transitions (edges), and cannot simultaneously be at the same spot or swap places. Therefore a solution $\boldsymbol{\pi}$ is a plan of non-colliding paths $\{\pi_1, \ldots, \pi_n\}$, and a common measure of a plan's cost is its *flowtime* (also referred to as Sum of Costs) defined as the sum of all service costs. Another quality measure, *latency*, quantifies the cost agents impose on each other through their interactions: path detours, congestion and delays. If we regard the distance cost d_i, i.e. the cost of the shortest path of a solitary agent a_i from start to goal, then the best attainable service cost maintains $|\pi_i^*| \geq d_i$. The *latency* is the sum of differences between service and distance costs.

Multi-agent terraforming We introduce an extension to the classical MAPF problem where we are also given a set of "self-propelled" obstacles $\mathcal{O}$ that can move while avoiding other obstacles and agents (similar to the classical MAPF problem). We define the cost of an obstacle's path as the

Algorithm 1: Terraforming using TF-CBS

Input: Graph G, agents $\mathcal{A}$, movable obstacles $\mathcal{O}$
Returns: An optimal plan $\boldsymbol{\pi}^*_{\text{TERRA}}$
$R.constraints \leftarrow \emptyset$ // root state
$agents \leftarrow \mathcal{A} \cup \mathcal{O}$
$R.paths \leftarrow \text{findPaths}(G, agents, R.constraints)$
$R.cost \leftarrow \text{flowtime}(R.paths)$
$R.conflicts \leftarrow \text{detectConflicts}(R.paths)$
$\text{insert}(R, \text{OPEN})$
while OPEN *not empty* **do**
  $N \leftarrow \text{pop}(\text{OPEN})$ // best first candidate
  $\langle i, j, p, t \rangle \leftarrow \text{getConflict}(N)$
  for $c \in \langle i, p, t \rangle, \langle j, p, t \rangle$ **do**
    $C \leftarrow N.constraints \cup c$
    $N' \leftarrow \text{clone}(N)$
    $N'.paths \leftarrow \text{findPaths}(G, agents, C)$
    $N'.cost \leftarrow \text{flowtime}(N'.paths)$
    $N'.conflicts \leftarrow \text{detectConflicts}(N'.paths)$
    $N'.constraints \leftarrow C$
    $\text{insert}(N', \text{OPEN})$

number of timesteps excluding *wait* actions, and mitigate the buildup of clutter by requiring that all obstacles return to their initial positions after moving. We call this new type of problem Terraforming-MAPF, or TF-MAPF, and denote its solution by $\boldsymbol{\pi}_{\text{TERRA}}$. Figure 1 illustrates how the search creates a shortcut by displacing a single obstacle to reduce *flowtime*.

Algorithm

To solve the Terraforming problem, we adapt the Conflict Based Search algorithm, or CBS (Sharon et al. 2015), which is complete and optimal. The search starts at the root state where it obtains the individual shortest paths for all agents. Next, a single collision $\langle i, j, p, t \rangle$ involving agents a_i, a_j is selected as a conflict at position p (vertex/edge) and timestep t. This conflict is converted into two constraints $\langle i, p, t \rangle, \langle j, p, t \rangle$ targeting a_i, a_j respectively, to prevent the agent from being at p at timestep t. Each constraint spawns a new child state that alters the agent's path, after which the child state is inserted into a priority queue OPEN. Proceeding in a best-first manner, the search extracts the state with the lowest *flowtime* from OPEN and resolves the next conflict. This cycle continues until a collision-free plan of minimal cost is found, i.e. an optimal solution $\boldsymbol{\pi}^*$.

Terraforming-CBS We apply our assumption of self-propelled obstacles, and require that they return to their initial location. Building upon CBS, we propose TF-CBS, where the key idea is to regard the movable obstacles as additional agents initialized at their goal location, and have their path cost account only for movement. Alg. 1 highlights the modifications to CBS that allow movable obstacles to move in response to colliding agents. The algorithm searches for single-agent paths that conform to constraints imposed by collisions between agents and obstacles. It then updates the *flowtime*, accounting for the path cost of agents and obstacles. Continuing in a best-first manner until a conflict-free plan is found, obstacles move only when deemed beneficial by the search. This way, TF-CBS maintains the completeness and optimality of CBS.

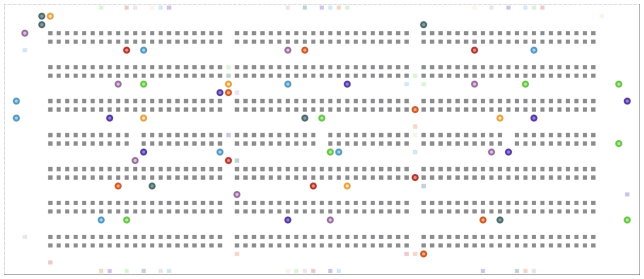


Figure 2: A 32×75 sized warehouse, densely packed with 876 shelves (in gray) and operated by 54 agents (coloured dots). Experiment video: https://bit.ly/3aMIomO

Evaluation

We implement our approach in C++ by extending IDCBS (Boyarski et al. 2020). We apply Terraforming to warehouses (one such warehouse is visualized in Figure 2) where 54 agents work together to determine the ideal obstacle displacement and cost, thereby fulfilling the task faster. An evaluation of four warehouses with various start and goal locations demonstrated how movable obstacles induce an average improvement of 15.1% in *flowtime* (from 1,794 to 1,522) and a reduction in *latency*, from a delay of +37 to a speedup of −245. Interestingly, the ability to achieve negative *latency* values represents a potential productivity boost beyond the warehouse's original design, thanks to dynamic shortcuts and congestion mitigation. The average planning time has increased from 6,338 seconds to 20,609 seconds due to the added complexity. Future work will concentrate on reducing the algorithm's running time.

Discussion and Future Work

In this work we introduce TF-MAPF—a novel formulation of the MAPF problem. We propose the TF-CBS algorithm which solves the problem in a manner that is complete and optimal. While our empirical evaluation indicates that movable obstacles allows for a non-trivial improvement in solution quality, the time-complexity of CBS may grow exponentially in the number of agents (Gordon, Filmus, and Salzman 2021). Thus, our approach of considering the obstacles as self-propelled agents can incur an exponential increase in the algorithm's running time. Therefore, future work will form heuristics to guide the search so that it focuses on prospective obstacles. In addition, we're interested in having the agents themselves displace the obstacles. This will relax the assumption of self-propelled obstacles.

References

Bellusci, M.; Basilico, N.; and Amigoni, F. 2020. Multi-Agent Path Finding in Configurable Environments. In *AAMAS*, 159–167.

Boyarski, E.; Felner, A.; Harabor, D.; Stuckey, P. J.; Cohen, L.; Li, J.; and Koenig, S. 2020. Iterative-Deepening Conflict-Based Search. In *IJCAI*, 4084–4090.

Gordon, O.; Filmus, Y.; and Salzman, O. 2021. Revisiting the Complexity Analysis of Conflict-Based Search: New Computational Techniques and Improved Bounds. *CoRR* abs/2104.08759. URL https://arxiv.org/abs/2104.08759.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence* 219: 40–66.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, S.; et al. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *SoCS*.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Magazine* 29(1): 9.