2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Multi-Agent Pathfinding with Obstacle Movement for Realistic Virtual Tactical Simulations on Topographic Terrains

Luigi Perotti Souza*, Edison Pignaton de Freitas†, Raul Ceretta Nunes*, Luis A. L. Silva*

*Federal University of Santa Maria, Brazil
{lpsouza, ceretta, luisalvaro}@inf.ufsm.br

‡Federal University of Rio Grande do Sul, Brazil
edison.pignaton@inf.ufrgs.br

***Abstract*—Multi-Agent Pathfinding (MAPF) algorithms represent a powerful tool for modeling realistic tactical movements of troops in military simulation systems. Solving MAPF problems while dealing with topographic terrains involves computing the most cost-effective and safe relief routes for agents with movement constraints. To minimize the overall topographic cost of agents' movement and the need to deviate from other stationary agents, this work considers a MAPF approach that respects real-world agents' movement characteristics, such as the agents' orientation, the limits of the agents' turning angles, and the relief inclinations they can safely navigate. To solve conflicts between agents while navigating uneven terrains, the proposed approach explores the attribution of agents' movement priorities related to the need to execute given missions. Most importantly, other agents without planned movement at the current mission situation, as they are stationary on safe and low-cost routes according to the terrain relief, are minimally displaced to nearby locations to give passage to the mission-critical agents. The MAPF algorithm is evaluated on a comprehensive set of test scenarios, with results analyzed using Generalized Linear Regression models. This analysis provides valuable insights into the MAPF algorithm's effectiveness in virtually modeling organized agent movement behaviors for developing simulation-based training and instruction activities.**

***Index Terms*—Tactical Agent Movement; Topographic Pathfinding; Multi-Agent Pathfinding; Agent-Based Simulation.**

I. Introduction

Multi-Agent Pathfinding (MAPF) algorithms are crucial to address agent navigation problems in Agent-Based Simulation Systems [1]. MAPF is dedicated to finding paths for agents to reach their goals while reducing path costs and avoiding agent collisions. To create more immersive experiences for simulation-based training and instruction activities, enhanced MAPF approaches for tactical simulation purposes can be investigated. These advancements entail the exploration of increasingly realistic agents and virtual terrain scenarios. Hence, achieving fidelity between the modeled virtual agent behaviors and those observed in the real world becomes critical.

Many defense simulation systems present requirements for military units' behaviors modeled as Multi-Agent Systems, particularly regarding the doctrine-based coordinated movement of these agents in the virtual environments. In the Artificial Intelligence (AI) literature, whenever MAPF algorithms are properly adjusted to the modeling of such simulation system needs, they effectively resolve tactical and strategic agent navigation problems. For instance, this is the case of agents' navigation capabilities implemented in simulation systems that permit users to plan and execute tactical missions involving terrestrial military vehicles [2], [3]. Systems like these require that the combined movement of the simulated vehicles be as realistic as possible to show the users how tactical missions would be executed in the real world.

The problem is that most MAPF algorithms proposed in the AI literature, such as [4]–[6], still superficially consider, or even neglect, the computation of low-cost and topographically safe routes in terrains with uneven relief. Fig. 1 illustrates a common real-world problem where originally static agents (with no pre-planned movements in the current mission situation) are located at terrain positions that obstruct the use of more direct routes for other agents that, due to the need of attributed missions, have higher movement priorities. Such clearance of low-cost terrain relief corridors is still more important when mission agents are forced to navigate rough terrain regions because they must deviate from the static agents. As depicted in Fig. 1, the need is to determine whether the movement of the stationary agents is fundamental to the search for higher-quality tactical routes for agents with increased military value according to the mission situation. In simulation systems that consider real-world maps to model the virtual environment, such as those used in military training, solving this kind of tactical navigation problem is crucial.

This work proposes a novel MAPF approach for computing more direct and realistic routes for agents in terrains with uneven topography. The approach considers that the modeled agents have limited movement capabilities due to the steep terrain inclinations, the agents' orientation (direction), and the maximum angles for executing curves. Most importantly, the proposed MAPF algorithm addresses scenarios where other

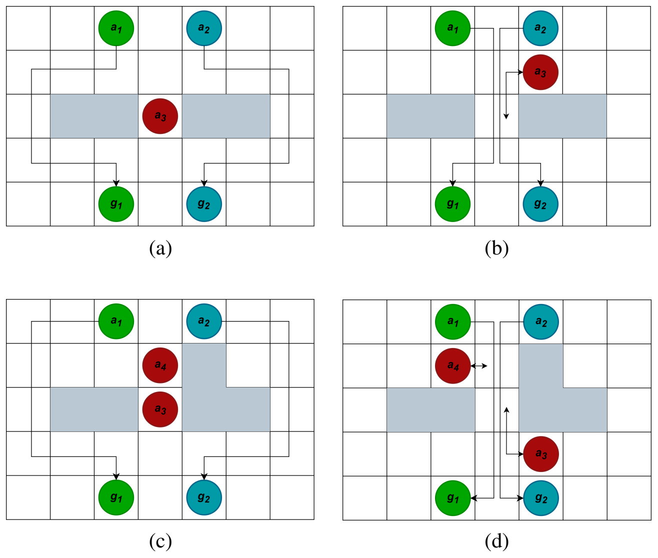


Fig. 1: MAPF for a problem with *task agents* a_1 and a_2 moving towards goals g_1 and g_2, considering obstacles on the map. (1a) presents a *support agent* a_3 as an obstacle in the planned path. This agent is stopped in a potential low-cost and topographically safe route that could be used to compute another agent's path. (1b) shows that the agent with no originally planned movement a_3 frees up terrain regions to compute a more direct route for other agents with higher movement priority. (1c) presents a better path blocked by agents a_3 and a_4. Due to their movement priority, the agents obstructing the passage of others do not release the route on the terrain relief. In contrast, (1d) presents a situation where a more direct path is calculated where the movement priority of agent a_1 is higher than that of agent a_2.

stationary agents (called *support agents*) are taken as (mobile) obstacles to the safe and low-cost topographic movement of the agents (called *task agents*) executing given military missions. As investigated through different experiments, whenever the overall *movement density* - a metric that considers the number of movement stops and curves resulting from the planned routes of the MAPF algorithm - associated with moving such static agents is better than that of avoiding them, the algorithm computes topographic routes by searching additional displacement routes for these support agents. With a statistical model of Generalized Linear Regression [7], the efficiency of the proposed MAPF solution for agent-based simulation is analyzed, providing evidence of its benefits compared to cases in which the obstacles impair the computation of more realistic and, therefore, higher-quality tactical route solutions.

II. Background to This Work

MAPF is a computationally complex problem, classified as NP-Complete [8] for optimal solutions. Consequently, its resolution needs ongoing research into implementing and refining novel strategies that yield computationally efficient solutions. Such MAPF problems can be formulated as graph search problems, which correspond to the tuple [9]:

$$\langle \mathcal{G}, \mathcal{A}, \varnothing, \mathcal{V}_{start}, \mathcal{V}_{goal} \rangle$$

such that $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is a graph, $\mathcal{V}$ is the set of terrain nodes and $\mathcal{E}$ is the set of edges that represent the allowed movements between these nodes, $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ is the set of agents, $\mathcal{V}_{start} = \{s_1, s_2, ..., s_n\}$ and $\mathcal{V}_{goal} = \{g_1, g_2, ..., g_n\}$ are the sets of start and goal agent locations, and $\varnothing = \{c_1, c_2, ..., c_k\}$ is the set of movement constraints.

According to [6], an action is a function $a : V \to V$ where $a(v) = v'$ indicates that when an agent performs an action on a vertex v, the result of this action is v'. The objective of each agent is to find a set of actions a forming a path p_i such that: the path is collision-free, meaning two agents cannot occupy the same vertex at the same time; the paths found for each agent are optimal or sub-optimal, ensuring that the distance traveled by each agent is minimized.

As investigated in many application scenarios, the Conflict-Based Search (CBS) [10] algorithm can effectively solve complex MAPF problems. It is organized in two phases: i) the high-level search to minimize the sum of the costs of the agents' paths, considering possible conflicts between them, and ii) the resolution of the conflicts found at the global level, making adjustments in the agents' paths.

The CBS high-level search uses a search tree to generate candidate solutions, identifying possible conflicts between the agents' paths and storing this information on each node of the search tree. At each algorithm iteration, the node with the lowest total cost is selected. Then the children of this node are expanded if there is a conflict in the calculated paths, applying a constraint to avoid agents' conflicts. A conflict is represented by the tuple $\langle A, v, t \rangle$, A being the conflicting agents occupying a vertex v at time t. If no conflicts exist on the current node, it is selected as the problem solution.

The CBS low-level search aims to resolve the previously identified conflicts. To do this, the search tree is traversed again, and each conflict is resolved through a spatio-temporal reservation technique. This technique adjusts the start or finish time of an agent's task. These conflicts can be of the vertex type: agents cannot occupy the same vertex at the same time, or edge type: agents cannot occupy the same edge at the same time. Constraints are added to the tree nodes, and the algorithm checks whether they are satisfied before expanding a node. A constraint $\langle i, v, t \rangle$ corresponds to an impediment of the agent a_i from occupying the vertex v at time t. This low-level resolution step is developed by adapting a single-agent pathfinding algorithm, such as A^*.

The work in [11] discusses a path search approach where the A^* algorithm uses an agent prioritization ranking in the search. That is relevant when dealing with the modeling of MAPF problems where agents have different responsibilities in the executed simulations. Instead of planning the path of all agents simultaneously, this strategy applies the concept of Cooperative Pathfinding [12], which calculates the path of each agent according to a defined priority. In doing so, the vertices and edges of each agent's path are added to a reservation table. Then, the next agent's path is calculated to avoid colliding with the reserved nodes. This algorithm is extremely efficient in calculating the agents' paths because it does not need to recalculate several paths in each collision.

The work in [13] investigates the navigation behavior of car-like robots. In particular, it considers the size of such agents and their movement limitations based on the axis and rotation of the vehicle wheels, therefore limiting the movement according to the agent's position and direction within the environment. Considering the kinematic and spatio-temporal constraints of such agents, the CL-MAPF algorithm optimizes the paths of the various agents while moving in a shared environment with obstacles and dynamic changes. Moreover, its CBS algorithm uses a Hybrid-State A* planner for the low-level solution, restricting occupied areas in a continuous time according to the speed, size, and rotation of the agent to terrain areas in the discrete-time t.

The work presented in [14] and [9] investigates the concept of dynamic terrain configuration in MAPF, which refers to strategies that modify the terrain structure to improve the agents' navigation. With such a *terraforming*, environment configurations can be defined as states of a graph, where a family of $\mathcal{G} = \{G_1, G_2, ..., G_n\}$ represents all possible graphs of a specific environment. Each configuration may represent, for instance, different positions of obstacles on the terrain. To identify potential route improvements for agents, the terraforming approach suggests that these improvements may be related to the changes computed in the environment configuration. This implies a series of graphs where the configuration graphs represent the vertices, and the route improvements are the edges. For graphs that can be dynamically configured, the work in [9] investigates the presence of mobile obstacles and changing agents. These changing agents are responsible for configuring the mobile obstacles on the terrain according to the current movement needs of the considered agents.

To sum up, the terraforming-like concept analyzed in this work involves considering that obstacles have their own movement capabilities. This is the case for vehicles parked on the terrain (support agents) that perform minimal movements to enable other agents (task agents) with higher movement priorities to use optimized paths. Thus, this concept is relevant to modeling realistic paths for the tactical movement of simulated forces, provided that the devised MAPF method is properly augmented to fulfill the needs of the targeted agent-based simulations. Moreover, this work is motivated by CL-MAPF [13] as it aims to simulate the movement of car-like agents, where this research is particularly focused on agents that have to safely navigate in terrains with topography [15].

III. A MAPF Approach with Support Agents' Navigation

The research problem investigated in this work is the realistic tactical movement of mission-critical agents in topographic terrains. It means that the simulated agents should execute navigation behaviors as closely as possible to the real-world agents' behaviors. This is fundamental for simulation-based training and instruction activities to effectively provide human learners with experiences that can be leveraged in solving similar real-world problems.

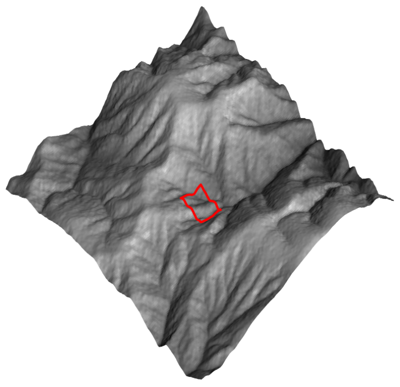
In virtual environments used in military simulation systems, agents representing modeled vehicles assess the terrain relief and navigate on its safe and low-cost routes while executing planned tactical activities [2], [3]. When agents are located at specific tactical positions and need to occupy other positions, they ought to exhibit an organized movement. One of the challenges in scenarios where the vehicles are deployed to tactical positions is that these positions are located between terrain areas with rough terrain relief. Moreover, other modeled agents are often displaced along with these terrain positions. Despite the presence of these stationary agents on low-cost corridor routes of the terrain, which can be considered static obstacles for the agents that need to move to complete a given mission safely, the navigation behaviors executed by all agents should explore route choices that would be observed in the represented real-world scenarios.

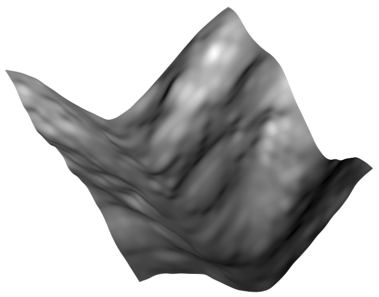
In many simulation problems where the space for the topographic agent movement is limited, such as within tactical positions in military operations executed in hilly terrains, certain agents end up performing more frequent stop-and-go and winding navigation behaviors. These attempts to stop, start, and change direction end up causing more congestion in the tactical area, leading to chaotic movement patterns. In extreme situations, this unrealistic tangled, messy movement scenario resembles a "spaghetti". Therefore, this work approaches the planning of agent routes to avoid the "spaghetti" behaviors in virtual simulations. Furthermore, it does that considering that the pathfinding models the simulated vehicles' movement characteristics besides realistic terrain conditions.

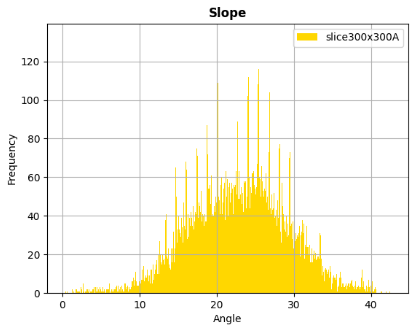
A. Topographic Terrain Selection and Agents' Behavior

The elevation map in Fig. 2a (available at [16]) was selected to develop this work. It is represented as a square matrix in which each position stores a height value. The terrain selection considers the histogram formed from the elevation angles of the terrain, aiming for a normal distribution graph. A histogram with low values referring to the inclination angles would result in a flat terrain with few natural obstacles to test the agents' limits. On the other hand, very high slope angles would result in various path corridors throughout the terrain, limiting the freedom of movement for the agents.

To navigate on terrains with topography, the selected maps represent the navigation costs, which are based on the terrain resolution. This information is used when calculating the distance between two adjacent nodes on the terrain during path planning. The movement between nodes representing the terrain respects the terrain's movement restrictions and the distance cost between them. As described in Fig. 3, each agent moves on the navigation map according to its orientation (N, S, E, and W), where perpendicular movement to the agents' orientation is not allowed.

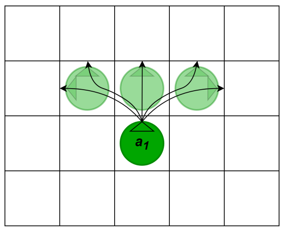

(a)


(b)

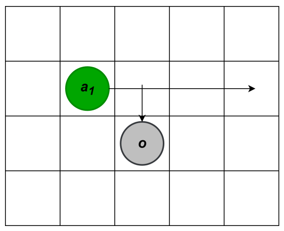


(c)

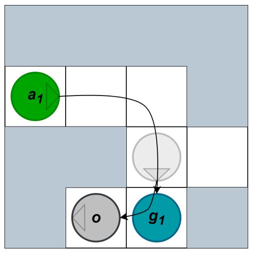
Fig. 2: The chosen terrain has a resolution of 3,000 × 3,000 pixels. Each pixel corresponds to a rectangular area of 3×3 squared meters in the real-world terrain, totaling an area of 81 km^2. (a) Map visualization. (b) 3D representation of the selected terrain slice. (c) Histogram showing the distribution of elevations in the terrain slice used in the tests conducted in this study.



(a)



(b)

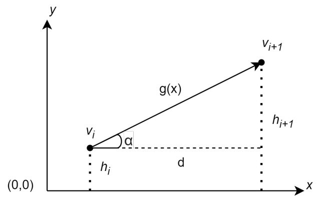


(c)

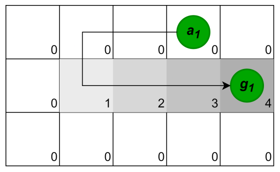
Fig. 3: Constraints and minimal movement of agents. (a) Agent movement according to the direction they are oriented (N, S, E, and W) while respecting the maximum curve angles of the agents. (b) Minimal movement of support agents (obstacle) o. (c) Support agent o clears up a path for task agent a_1.

Considering an elevation map of the terrain, the navigation restriction between the vertices of the navigation graph is determined by the agents' movement capabilities while moving through the relief elevations. The elevation map is represented as a three-dimensional matrix (x, y, z), where each (x, z) pair of coordinates has a height y to be used in the local $g(n)$ and heuristic $h(n)$ (in this work, the Euclidean distance function in the $\mathbb{R}^3$) path cost calculations of the A^* cost function $f(n) = g(n) + h(n)$. As described in [17], the navigation cost between two vertices can be computed as presented in Fig. 4a, where $g(n)$ is given by (1):

$$g(x) = \sqrt{d^2 + \Delta h^2} \tag{1}$$



(a)



(b)

Fig. 4: Example of movement limitation defined by $d = 1$ and $\alpha_{max} = 45°$. The agent a_1 is unable to move between vertices where Δh is greater than 1.0, causing the agent to take a longer path on the map. (a) Movement cost between two vertices. (b) Route of agent a_1 to the destination g_1 given the terrain heights h represented at each vertex.

In Fig. 4a, d indicates the distance between each vertex node, determined by the resolution of the elevation map, and Δh is the difference between the heights of vertices v_i and v_{i+1}. For an agent to navigate between two nodes v_i and v_{i+1}, the angle formed by $\overrightarrow{v_i v_{i+1}}$ and the x-axis must be less than or equal to α_{max}, as shown in Fig. 4b. In this work, ascending and descending a terrain inclination have the same topographical cost, causing the same effect in a route.

B. A Two-Step MAPF Solution

The steps for defining agent priorities and executing the MAPF algorithm depend on the attributed missions that one or more agents must execute in the simulations. Once the movement agent limitations are defined, the MAPF algorithm computes collision-free paths for the agents on the topographic terrain. It involves (i) the calculation of the route for *task agents* and (ii) the calculation of the route for *support agents*, as detailed in Algorithm (2).

First, the MAPF approach proposed in this work explores the Cooperative Search [11], [12] to compute paths for the task agents. That is because this algorithm internally uses priority information in its agent path computations. Moreover, paths in terrains with topography are often computed within path corridors determined by the terrain relief. Due to the need for computing routes where high-priority agents execute an organized movement behavior through these corridors, tests developed in this work showed the Cooperative Search algorithm is the best choice.

Second, the proposed approach also explores the Conflict-Based Search (CBS) method [10] to solve conflicts between paths computed to support agents. That is because multiple support agents may need to move from their current stationary positions to allow the Cooperative Pathfinding method to determine more direct routes for task agents. While multiple support agents need to move, the CBS method solves the conflicts between the returned paths. Algorithm 1 demonstrates how paths are calculated for support agents that unconditionally avoid task agents.

Algorithm 1 Avoidance Method

```
1: $pos \leftarrow Agent.position$
2: $pos.t \leftarrow 0$
3: OPEN $\leftarrow pos$
4: while OPEN $not\ empty$ do
5:     $pos \leftarrow$ best node from OPEN
6:     if has constraints for $pos$ then
7:         $Path$ get path using $pos.parent$
8:         return $Path$
9:     end if
10:    for $n_i \in pos.Neighboors$ do
11:        if $n_i$ $not\ visited\ in\ time\ t$ then
12:            Insert $\langle n_i, pos.t + 1\rangle$ into OPEN
13:        end if
14:    end for
15: end while
```

Algorithm 2 MAPF Mobile Agents Method

```
1: $Solutions \leftarrow \varnothing$
2: $ReservationTable \leftarrow \varnothing$
3: $TaskAgents \leftarrow selectTaskAgents(Agents)$
4:         according to $TargetMission$
5: while $TaskAgents\ is\ not\ empty$ do
6:     $taskAgent_i \leftarrow removeHighPriority(TaskAgents)$
7:     $taskAgent_i.Constraints \leftarrow ReservationTable$
8:     $taskAgent_i.Path \leftarrow$
9:         $PathAvoidingReservation(taskAgent_i)$
10:    $ReservationTable \leftarrow$
11:        $ReservationTable + taskAgent_i.Path$
12:    Insert $taskAgent_i$ to $Solutions$
13: end while
14: $MobileAgents \leftarrow selectMobileAgents(Agents)$
15:        according to $TargetMission$
16: $MobileAgents.Paths \leftarrow$
17:     $ConflictBasedSearch(MobileAgents)$
18:        using $Avoidance()$ with $ReservationTable$
19: Insert $MobileAgents$ to $Solutions$
20: return $Solutions$
```

1) Movements Constraints and Spatio-Temporal A^*: As shown in Fig. 3a, the agents have limited movement capabilities based on their orientations, making perpendicular movements to their orientation impossible. All task and support agents have the same movement constraints determined by the terrain inclinations. This study defines the maximum inclination an agent can use when moving at 20°. This angle was established after conducting tests on the terrain in Fig. 2a.

The movement possibilities for agents used in the development of this work, as shown in Fig. 3, are (i) move forward, maintain the current direction; (ii) move left, maintain the current direction; (iii) move right, maintain the current direction; (iv) move left, turn 90° to the left; and (v) move right, turn 90° to the right. The agent can turn without changing its geographic direction, or it can turn and change its orientation based on the side of the curve (turning left involves a 90° left turn while turning right involves a 90° right turn). This implementation represents a "lane change" and a "closed curve". For the spatio-temporal A^* algorithm, in addition to the movement characteristics described above, a "Wait" action was considered in the implementations conducted in this work.

In this case, an agent can occupy the same vertex more than once by taking the "Wait" action. The problem is that a wrong choice of heuristic to use in the MAPF algorithm can result in excessive path repetition, as the Euclidean distance may lead to routes with vertices spatially closer to the goal but without direct paths to it. As observed in this work, the MAPF algorithm will tend to visit the neighboring vertex of an agent due to their shorter Euclidean distance to the destination, causing them to be revisited multiple times when the agent can remain stationary. This results in unnecessary repetition of already visited nodes. To address this issue, the concept of *True Heuristic Distance* [18] was implemented in this work. In it, the A^* algorithm is executed in reverse order, starting from the goal towards the visited vertex. This enables the storage of the cost of each visited node in a separate table. Whenever the A^* needs to check the distance from a node to the goal, it verifies if this distance is already present in the table.

2) Moving the Task Agents: The MAPF approach investigated in this work initially computes a solution for each task agent using the Cooperative Search algorithm [12]. This algorithm implements a prioritized search, executed individually for each agent. The search is based on a pre-defined ordering of the agents, where higher priorities are attributed to agents that must execute a given mission in the current simulations.

Due to the need to model convoy moment behaviors, which is quite common in the modeling of movement of forces in tactical simulation scenarios, the agent with the highest priority is the one occupying the front position of the convoy formation, meaning there should be no other agents in front of it. The priority order must follow these criteria since a vehicle with higher priority naturally passes through the positions of lower-priority ones to reach its deployment location. During the initial calculation of the paths for task agents, the positions of support agents are disregarded. Task agents are not responsible for avoiding collisions unless they are with other task agents of higher priority. As a result, support agents are not considered in this stage of MAPF computation.

3) Moving the Support Agents: The CBS method is applied at a later path search stage to solve path conflicts among support agents. It is adapted to solve path conflicts between support agents scattered throughout the terrain, where these agents do not have predefined goal positions within the current mission simulations. This allows the agents to have the freedom to search for alternative paths to evade conflicts. Since support agents do not have a defined order of movement priority, adapting the Cooperative Search algorithm to handle these conflicts has shown to be unrealistic and sometimes inefficient. That is because an inappropriate ordering of these agents leads to costly paths to avoid unrealistic conflicts.

In essence, once the path for each past agent is calculated and stored in the *Reservation Table*, this table holds predefined terrain obstructions for paths that can be computed for the support agents. A support agent must avoid colliding with a task agent without interfering with the latter's trajectory. As implemented in this work, support agents unconditionally deviate from task agents, as they cannot occupy any vertex v that is reserved in the table for a time step t. To solve potential collisions among support agents, therefore, the CBS algorithm, along with the *Avoidance()* function in Algorithm (1), is used.

The *Avoidance()* function executes a low-level search: the step in which conflicts between previously computed paths are resolved. Since the position of the support agent is irrelevant to the solution, and it only matters whenever it clears a path for another task agent, this function implements the A^* algorithm without defined goals. It means that the algorithm searches for a path until it finds a vertex without defined restrictions, i.e., a vertex that no other agent will visit when the concerned support agent occupies it. The final position of the support agent in this search process does not matter in the performed simulations. The agent also does not need to return to the initial position since it originally cleared up a low-cost route with decreased topographic costs between the considered tactical positions.

In the conflict resolution step, therefore, the initial node of the constructed search tree receives the entire *Reservation Table* described earlier. Thus, it used this table as a movement constraint in the latter MAPF computations. Once it is applied to the initial node, all agent positions and conflict expansions will have the positions of task agents along their trajectories represented as movement restrictions.

In summary, the CBS is based on applying the *Avoidance()* function as the *Low-Level Search* for the MAPF. Despite the reduction of the conflicts between support agents achieved by the CBS, the time-saving computations obtained in the first stage of the proposed MAPF approach by ignoring the obstacles become indifferent when combined with the cost of conflict resolution among support agents.

IV. Experiments and Results

The experiments conducted in this work evaluate the impact of support agents' movement to enable the computation of more direct paths for task agents, thereby improving path quality for mission-critical agents. The tests compare two path planning methods: a) a MAPF method considering support agents/obstacles that task agents must deviate, and b) a MAPF method considering support agents that must clear paths for task agents. The methods consider the movement restrictions in Fig. 3a, which include the ability to move with terrain inclinations less than 20°.

The set of task agents was associated with identical movement priorities and other movement capabilities. Each agent's initial positions and orientations were randomly generated within specific quadrants on the terrain. Similarly, the destination positions of the agents were also randomly generated in another quadrant. Using these predefined terrain areas, which represent initial and final tactical positions associated with a given mission, the computed paths for the agents resulted in different degrees of conflict among them. Therefore, the tests aimed to examine the quality of the resulting paths.

In the test executions, once the tested Cooperative Search algorithm calculates the individual path for each task agent, the stationary (but mobile) support agents are placed on the terrain. Each support agent o_i is positioned in such a way that $o_i \in \Pi_i$, where Π_i represents the pre-computed path of a task agent. Consequently, n positioned support agents will obstruct the route of n other task agents, either needing the movement of these obstacles to clear up the route of such task agents or requiring the computation of task agents' routes that deviate from the stationary support agents.

To assess the impact of adding obstacles to the previously computed task agents' paths, the tests conducted in this work considered setups with 15, 30, 45, and 60 task agents with 15 obstacles along their paths. The distances between the task agents' starting and ending tactical positions were reduced, and the support agents were placed closer to the task agents, thereby reducing the distance between agents and obstacles and increasing the conflicts to be solved.

The behavior associated with the support agents acting as obstacles remains the same in all test scenarios, with one configuration acting as mobile support agents and another as static support agents. 22 initial and final tactical configurations were used in the tests, each executed 5 times with varied positions for the support agents, resulting in a set of 110 tests. The same set of tests was applied for all numbers of task agents, resulting in a total of 660 executed configurations. To the statistical analysis, the number of test configurations adheres to the sample size required for averaging with a 5% margin of error and a 99% confidence level.

This work proposes a "density" metric to assess a solution's quality level, following the metric proposals of [19]. The metric, defined as *Movement Density (MD)*, is expressed as the inverse of the sum of the number of movement turns and the simulation times a task agent remained stationary (i.e., due to the used spatio-temporal path planning approach to avoid path conflicts with other agents) in a given solution for all task agents, as shown in (2):

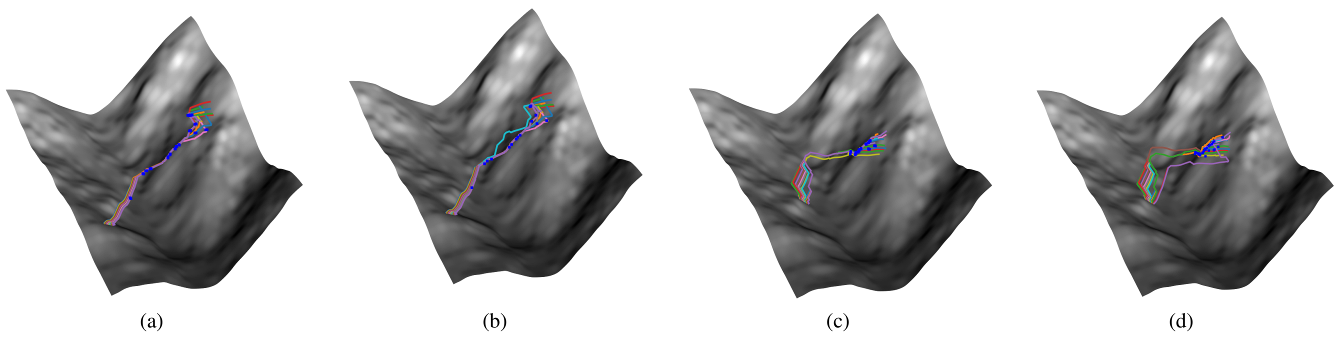


Fig. 5: Path results from (a and c) the MAPF Mobile Agents (MA) method where (a) support agents (blue dots) are moved from their stationary positions to permit computing less winding, topographically safe and low-cost tactical routes, and (b and d) the MAPF Static Agents (SA) method where these stationary agents are kept on their obstructive route locations.

$$MD = \frac{1}{1 + \sum_{i=1}^{n} (t_i + b_i)} \tag{2}$$

where b_i represents the number of stops for agent i, and t_i represents the number of path turns. The MD function has a domain of $0 < MD \leq 1$, such that *the smaller the MD value, the greater the number of turns and stops in a solution, indicating a lower quality solution* (in extreme simulation cases, *spagetti*-like agents' navigation behaviors). This result can be explained by the fact that paths with fewer deviations and waiting times imply a solution with fewer conflicts between task agents. A solution with $MD = 1$ indicates a solution with no stops and turns.

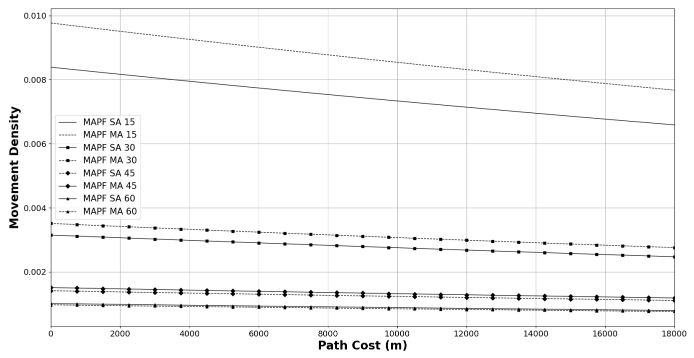


Fig. 6: MD metric results × path cost in meters.

TABLE I: Statistical results of the MD metric.

	Estimation	Std Deviation	t Value	$P(>\|t\|)$
Intercept	-4.78E+00	2.14E-02	-223.915	<2e-16
MAPF SA 15	-1.34E-05	3.93E-06	-3.413	0.000656
MAPF MA 15	1.52E-01	2.38E-02	6.384	2.17E-10
MAPF SA 30	-9.80E-01	3.03E-02	-32.302	<2e-16
MAPF MA 30	-8.71E-01	2.96E-02	-29.407	<2e-16
MAPF SA 45	-1.78E+00	3.69E-02	-48.248	<2e-16
MAPF MA 45	-1.71E+00	3.70E-02	-46.383	<2e-16
MAPF SA 60	-2.11E+00	4.96E-02	-42.602	<2e-16
MAPF MA 60	-2.16E+00	4.97E-02	-43.32	<2e-16

The experimental results of the MAPF algorithms were statistically analyzed using a Generalized Linear Model (GLM) (details of how to set up such statistical models can be found in [7]). These models were implemented using the R software and the Gamlss tool [20]. This statistical approach enables the usage of different distributions to analyze the obtained results. Thus, it is possible to establish a relationship between a link function (here, a logarithmic function) and a linear combination of explanatory variables D_i. The base method (D_1) in this analysis is the MAPF SA - 15 Agents configuration that computes paths to avoid static obstacles.

Fig. 5 presents path instances returned with the proposed MAPF Mobile Agents (MA) method compared to the MAPF Static Agents (SA) method. The test results for the MD metric are presented in Table I and Fig. 6. The results in Table II show that the MAPF MA algorithm with the 15, 30, and 45 agent's configurations achieved a better Movement Density (MD) than their corresponding MAPF SA results (i.e. when the MD values are negative in Table II, the higher the negative value, the worst the MD result; Fig. 6 better allows one to assess these MD results). Moreover, the MD results for the higher number of agents' configurations optimized the results of the base method used in the constructed statistical model (i.e., as indicated by the negative values in Table II). An exception of this trend was observed on the tests executed with the 60 agents' configuration. In this case, the resulting paths from the MAPF MA method involving 60 agents had worse MD values (-87.92) than its corresponding MAPF SA method results (-88.41) because an increase in the number of agents also led to increased stops, detours, and conflicts. In reduced terrain spaces where many agents are involved in the tactical movement tasks, moving the obstacles to clear the routes may cause additional and unwanted obstructions for other agents. However, as the MD results for the MAPF MA method are, in general, higher than those for the MAPF SA methods with the same number of agents, better quality paths are generated when support agents move to allow computing improved paths for task agents, reducing the number of stops and curves in the computed trajectories.

TABLE II: Overall results for the tested configurations compared to the base method MAPF SA 15 Agents.

Configuration	Execution Time	Expanded Nodes	Movement Density
15 Agents - MAPF MA	-8.61%	-16.77%	16.44%
30 Agents - MAPF SA	-26.16%	-47.08%	-62.45%
30 Agents - MAPF MA	-37.08%	-62.64%	-58.13%
45 Agents - MAPF SA	-73.53%	-81.87%	-83.15%
45 Agents - MAPF MA	-78.15%	-88.53%	-81.99%
60 Agents - MAPF SA	-84.56%	-94.70%	-87.92%
60 Agents - MAPF MA	-84.63%	-95.94%	-88.41%

Table II summarizes other test results. They show that the MAPF MA algorithm is faster than the MAPF SA algorithm when considering the same number of agents (e.g., for 45 agents, -78.15 is a better result than -73.53). Due to the execution of conflict resolution procedures, however, the computing time of all the MAPF-tested methods is on the scale of minutes, where optimizations are still needed to deal with real-time virtual simulation problems. Moreover, the MAPF MA algorithm expanded fewer nodes on the grid terrain representation during the pathfinding execution (e.g., for 45 agents, -88.53 is a better result than -81.87). The explanation for this phenomenon is that static obstacles force collision and recalculation of the previously returned paths. Since these obstacles obstruct the previous lower-cost topographic routes, the search needs to expand more terrain nodes along the path to find new routes. This result also advocates for the benefit of the proposed approach in moving the obstacles.

V. Final Remarks

MAPF algorithms, due to their complexity, require continuous research whenever one requires realistic path-planning solutions for agent-based simulations. In this setting, this work presents a MAPF approach for computing non-conflicting paths in multi-agent simulation environments. The work analyzes important MAPF requirements for modeling tactical agent navigation behaviors in uneven terrains. A key proposal presented here involves the possibility of moving stationary agents with no planned movement in the current simulation situation. That aims to provide more organized, safer, lower-cost topographic routes for mission-critical agents. The proposed MAPF approaches this problem by combining and extending the MAPF cooperative and conflict-based search methods to construct enhanced paths for task agents to move between tactical positions. Finally, it is possible to name important directions for future work, such as MAPF implementation enhancements to optimize the computing time of solutions for real-time simulation applications; the study of different tactical MAPF pathfinding algorithms for approaching other task and support agents deployment situations, among others.

Acknowledgment

We thank the Brazilian Army Strategic Program ASTROS for financial support through the SIS-ASTROS GMF project (898347/2020) - TED 20-EME-003-00.

References

[1] C Macal. Everything you need to know about agent-based modelling and simulation. *Journal of Simulation*, 10:144–156, 2016.

[2] Wojciech Dawid and Krzysztof Pokonieczny. Methodology of using terrain passability maps for planning the movement of troops and navigation of unmanned ground vehicles. *Sensors*, 21(14):4682, 2021.

[3] Cesar Pozzer, João Martins, Lisandra Fontoura, Luis A. L. Silva, Mateus Rutzig, Raul Nunes, and Edison P. Freitas. Sis-astros: An integrated simulation system for the artillery saturation rocket system (astros). In *Proc. of the 12th International Conference on Simulation and Modeling Methodologies, Technologies and Applications - Volume 1: SIMULTECH,*, pages 194–201. INSTICC, SciTePress, 2022.

[4] Taoan Huang, Sven Koenig, and Bistra Dilkina. Learning to resolve conflicts for multi-agent path finding with conflict-based search. In *Proc. of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11246–11253, 2021.

[5] Sumanth Varambally, Jiaoyang Li, and Sven Koenig. Which mapf model works best for automated warehousing? In *Proc. of the International Symposium on Combinatorial Search*, volume 15, pages 190–198, 2022.

[6] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proc. of the International Symposium on Combinatorial Search*, volume 10, pages 151–158, 2019.

[7] John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

[8] Bernhard Nebel. On the computational complexity of multi-agent pathfinding on directed graphs. In *Proc. of the International Conference on Automated Planning and Scheduling*, volume 30, pages 212–216, 2020.

[9] David Vainshtain and Oren Salzman. Multi-agent terraforming: Efficient multi-agent path finding via environment manipulation. In *Proc. of the International Symposium on Combinatorial Search*, volume 12, pages 239–241, 2021.

[10] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[11] Hang Ma, Daniel Harabor, Peter J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proc. of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7643–7650, 2019.

[12] David Silver. Cooperative pathfinding. In *Proc. of the AAAI conference on artificial intelligence and interactive digital entertainment*, volume 1, pages 117–122, 2005.

[13] Licheng Wen, Yong Liu, and Hongliang Li. Cl-mapf: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints. *Robotics and Autonomous Systems*, 150:103997, 2022.

[14] Matteo Bellusci, Nicola Basilico, and Francesco Amigoni. Multi-agent path finding in configurable environments. In *Proc. of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 159–167, 2020.

[15] Qinghe Liu, Lijun Zhao, Zhibin Tan, and Wen Chen. Global path planning for autonomous vehicles in off-road environment via an a-star algorithm. *International Journal of Vehicle Autonomous Systems*, 13(4):330–339, 2017.

[16] ASF DAAC. Palsar_radiometric_terrain_corrected_high_res, 2015. NASA Alaska Satellite Facility DAAC.

[17] K Zhigalov, D KS Bataev, E Klochkova, OA Svirbutovich, and GA Ivashchenko. Problem solution of optimal pathfinding for the movement of vehicles over rough mountainous areas. In *IOP Conference Series: Materials Science and Engineering*, volume 1111, page 012033. IOP Publishing, 2021.

[18] Ariel Felner and Nathan R Sturtevant. Abstraction-based heuristics with true distance computations. In *Symposium on Abstraction, Reformulation and Approximation*, 2009.

[19] Amudapuram Mohan Rao and Kalaga Ramachandra Rao. Measuring urban traffic congestion-a review. *International Journal for Traffic & Transport Engineering*, 2(4), 2012.

[20] D Mikis Stasinopoulos and Robert A Rigby. Generalized additive models for location scale and shape (gamlss) in r. *Journal of Statistical Software*, 23:1–46, 2008.