Autonomous Terrain Leveling Using a multi robots System: A formal framework for architecture and motion planning

Thanh Binh DO[1] and François GUERIN[1]

***Abstract*— Autonomous terrain leveling operations significantly impact project timelines, cost efficiency, and environmental sustainability within earthworks, which constitute a substantial portion of the public works market. This paper introduces the Optimal Transport Planner Framework(OTPF), an innovative framework for multi-robot systems designed to automate material transport and surface leveling tasks by leveraging the online Multi-Agent Pickup and Delivery (MAPD) problem paradigm. OTPF addresses critical operational layers, specifically an optimal transport (OT) layer for task modeling and a Task Allocation Layer employing heuristic and consensus-based algorithms for efficient agent coordination. To effectively manage dynamic task assignments and collision-free motion, we propose a hierarchical task allocation and motion planning approach. OTPF integrates strategies inspired by MAPD methodologies to dynamically assign robotic agents to tasks, balancing operational efficiency with conflict resolution.**

***Index Terms*— Multi robots path planning, Multi robots task allocation, automation process**

I. INTRODUCTION

Earthworks (Fig 4) constitute constitute approximately 19% of the total public works market, playing a critical role in construction and frequently determining project schedules. Typically isolated from other construction activities, earthworks operations span diverse sectors, including asphalt and concrete mixing plants, waste disposal sites, quarries, and secure operational areas. Despite their importance, the earthworks industry faces several key challenges:

- *Labor Shortages and Seasonal Fluctuations:* Reports indicate that 82% of recruitment efforts are challenging, while nearly half of all earthworks activities are concentrated within a short seasonal period (June to September), significantly constraining productivity.
- *High Carbon Footprint:* Earthworks machinery typically consumes around 200 liters of diesel per machine daily, raising substantial environmental concerns.
- *Operational Complexity:* The repetitive, isolated nature of earthworks tasks combined with the need for precise scheduling complicates effective optimization of operations.

In this paper, we introduce the **Optimal Transport Planner Framework (OTPF)**, an autonomous vehicle-based framework designed to automate earthwork logistics by dividing the overall material transport challenge into manageable sub-problems. OTPF systematically tackles these sub-problems—allocation of robotic agents, optimal material transfer paths, and dynamic coordination—in order to efficiently level ground surfaces. Specifically, OTPF addresses:

- *Enhanced Operational Efficiency:* Streamlining material movement to reduce operational costs %.
- *Environmental Benefits:* Optimizing routes and schedules to reduce carbon emissions %.
- *Robust Integration:* Adapting dynamically to changing operational conditions through integration of complex parametric variations.
- *Scalable Automation:* Leveraging established Automated Guided Vehicles (AGVs) and Multi-Agent Pathfinding (MAPF) technologies to support large-scale, heterogeneous operations.

The implementation of OTPF requires solving several interconnected sub-problems, including integrating simulation models, establishing clear automation tasks, linking performance metrics with control parameters, and developing resilient multi-robot coordination mechanisms suitable for dynamic conditions.

II. RELATED WORKS

A. Industrial Automation

Fleet automation problems have attracted significant attention in various applications. Mansouri et al. [1] addressed the automation of drilling processes in open-pit mines by decomposing the automation into interrelated sub-problems, including sequencing, motion planning, machine allocation, coordination, and temporal planning. They applied a constraint satisfaction problem (CSP) approach, heuristic-guided backtracking search, and online temporal reasoning for efficient management. In recent work, Chen et al. [2] highlighted challenges in mining automation such as environmental complexity, device coordination, and inadequate real-condition validation. Their "Parallel Mining" framework utilizes virtual simulations and digital twins, integrating data-driven learning, virtual scenario generation, and parallel computing for robust system validation. Xie et al. [3] proposed a cooperative framework involving human and cyber-physical systems within mining operations, emphasizing real-time interaction and adaptive task distribution.

B. Multi-Robot Task Allocation

Our work is related to the lifelong Multi-Agent Pickup and Delivery (MAPD) framework [4], [5], [6], which decomposes tasks into sequences of Task Allocation and Path

*This work was supported by Normandy Region, SGPI France 2030, BPI France(Planification Robotique Autonome pour la réalisation de Travaux Éco-responsables) (PROMETE) Project)

[1]Thanh Binh DO, François GUERIN are within the Research Group in Electrotechnics and Automatic Control (GREAH) of Le Havre Normandy University, Le Havre (France), 76600 thanh-binh.do, francois.guerin@univ-lehavre.fr

Finding (TAPF) problems, implemented either centrally or distributively.

C. Multi-Robot Path Finding

Multi-Robot Path Finding (MRPF) is a core robotics problem focused on computing collision-free trajectories for multiple agents navigating a shared environment. Recently, tremendous works [7],[8], [9],[10], [11], [12], have extensively explored Multi-Robot Path Finding (MRPF). Wen et al. [8] introduced decentralized methods for collision avoidance, demonstrating scalability in dynamic environments. Zhou et al. [13] proposed robust conflict resolution strategies for multi-robot systems.

III. Problem description and problem formulation

A. Environment Representation

Definition 1. We model the *working environment* as an $m \times n$ matrix M, where each cell at coordinates (i, j) has:

- *Current Material Level*: $v(i, j)$, the existing quantity of material.
- *Target Material Level*: $T(i, j)$, the desired quantity within allowable deviation Δ_{height}.

1) Cell Classification:

Definition 2. A cell (i, j) is *free* if

$$T_{\min} < v(i, j) < T_{\max}.$$

Definition 3. A cell (i, j) is a *working cell* if it is free but currently restricted due to ongoing operations, making it unavailable for idle or resting positions.

Definition 4. A cell (i, j) is classified as a *deficit cell* if

$$d_{\min} < v(i, j) \leq d_{\max}.$$

Definition 5. A cell (i, j) is classified as a *surplus cell* if

$$s_{\min} \leq v(i, j) < s_{\max}.$$

Definition 6. A cell (i, j) is considered an *obstacle* if

$$v(i, j) \leq a \quad \text{or} \quad v(i, j) \geq b.$$

2) Operational Zone:

Definition 7. An *Operational Zone* $C \subseteq M$ is a connected subset under h-connectivity, where h is an integer. For $h = 8$, two cells (i_1, j_1) and (i_2, j_2) are neighbors if:

$$\max\big(|i_1 - i_2|,\, |j_1 - j_2|\big) = 1.$$

Boundary cells of C must be classified strictly as either surplus or deficit cells, and there must be at least one such boundary cell.

Definition 8. A cell $(i, j) \in C$ is a *boundary cell* if it has at least one neighbor outside C under h-connectivity.

3) Contour of Operational Zone:

Definition 9. A cell (i, j) is **traversable** if it is neither an obstacle nor restricted operationally.

Definition 10. The *contour* of an operational zone C, denoted $\mathrm{Contour}(C)$, is the set of traversable cells that neighbor a boundary cell of C:

$$(i, j) \in \text{Contour}(C) \iff \big((i, j) \text{ is traversable}\big) \wedge \big((i, j) \text{ neighbors a boundary cell of } C\big).$$

4) Directional Contour Cells:

Definition 11. A traversable cell $(i, j) \in C$ is a *Directional Contour Cell* in direction $d = (d_i, d_j)$ if

$$(i, j) \in \text{Contour}(d) \iff \begin{cases} (T_{\min} < v(i, j) < T_{\max}) \\ \wedge (i + d_x, j + d_y) \in C \end{cases}$$

$(i + d_i, j + d_j)$ is either a deficit cell or a surplus

B. Graph-Based Representation

We further represent the environment as an undirected connected graph $G = (V, E)$ similar in [14]. Nodes represent specific locations, and edges represent allowable paths between these locations. Formally:

Definition 12. *Nodes* V represent permissible positions.

Definition 13. *Edges* $E \subseteq V \times V$ represent feasible paths, with each edge (u, v) having a length $l(u, v)$.

Distance between nodes v_1 and v_2 is defined as the shortest path length between them.

C. Task Definitions

A task τ_k is defined as the triplet:

$$\tau_k = (v_k^s,\, v_k^e,\, \mu_k)$$

where:

- v_k^s: start node (supply),
- v_k^e: end node (demand),
- μ_k: material volume to transport.

D. Node Classifications

Definition 14. Nodes in the graph are classified as follows:

- *Working Nodes (Task Endpoints)*: Nodes specifically designated for loading or unloading materials.
- *Idle Nodes (Parking Nodes)*: Nodes where robots remain after completing their missions.
- *Transit Nodes*: Nodes not specifically designated for tasks or idle resting, but where robots can temporarily wait.

E. Robots

Robots are autonomous forklift-like vehicles capable of transporting up to 2 tons of material). They are defined with discrete orientations o_i^t and locations p_i^t

Definition 15. Robots have discrete orientation d_i^t and position p_i^t at time t

Definition 16. Robots perform the following discrete actions:

- *Move*: Traverse edges after adjusting orientation; the duration is given by $T_{mo}(l)$.
- *Rotate*: Adjust orientation by increments of $\pm d$; the duration is $T_{ro}(d)$.
- *Wait*: Remain stationary at the current location; the duration is $T_{wa}(t)$.
- *Load/Unload*: Handle material at specific orientations, with durations T_{ld} and T_{ul} respectively.
- The maximum volume a robot can move per step, $m_{\max}$, satisfies

$$0 \leq m \leq m_{\max}.$$

Each robot starts and ends its tasks at an idle node.

F. Problem Objective

Given k robots, the system transfers material from surplus to deficit cells, optimizing these criteria:

- **Total Transportation Cost**: Minimize the cost proportional to distance moved and volume transported:

$$\sum_{\text{transfers}} (\text{distance}) \times (\text{volume})$$

- **Height Discrepancy**: Minimize difference between actual and target cell heights:

$$\sum_{i,j} |v(c[i,j]) - T[i,j]|$$

- **Collision Avoidance**: Find collision-free trajectories $p_k(t)$ for robots k between initial q_0 and final q_n configurations.

Final condition for every cell must meet:

$$T[i,j] - \Delta_{\text{height}} \leq v(c[i,j]) \leq T[i,j] + \Delta_{\text{height}} \tag{1}$$

IV. Transport plan generation

A. Transport generation

a) Primal Problem: Let I be the index set of supplier nodes and J the index set of demand nodes. Define decision variables

$$x_{ij} \quad \text{for} \quad i \in I,\ j \in J,$$

The primal optimization is:

$$\begin{aligned} \min_{x_{ij}} \quad & \sum_{i \in I} \sum_{j \in J} c_{ij}\, x_{ij}, \\ \text{s.t.} \quad & \sum_{j \in J} x_{ij} = s_i, \quad \forall\, i \in I, \\ & \sum_{i \in I} x_{ij} = d_j, \quad \forall\, j \in J, \\ & x_{ij} \geq 0, \quad \forall\, i \in I,\ j \in J. \end{aligned} \tag{2}$$

where

- x_{ij} is the quantity of material transported from supplier i to demand j.
- c_{ij} is the cost per unit of transporting from i to j.
- s_i is the total supply available at node i.
- d_j is the total demand required at node j.

b) Dual Problem: Introduce dual variables

$$u_i \quad (i \in I), \qquad v_j \quad (j \in J),$$

where u_i and v_j correspond respectively to the supply and demand constraints. The dual formulation is:

$$\begin{aligned} \max_{u_i,\, v_j} \quad & \sum_{i \in I} s_i\, u_i \ + \ \sum_{j \in J} d_j\, v_j, \\ \text{s.t.} \quad & u_i + v_j \leq c_{ij}, \quad \forall\, i \in I,\ j \in J. \end{aligned} \tag{3}$$

B. Dynamic, Batch-Assignment Extension

We extend this to a time-indexed, robot-aware transport problem. Let K be the set of robots and let t index discrete time batches. We define

$$x_{ijkt} \quad i \in I,\ j \in J,\ k \in K,\ t = 1, \ldots, T,$$

where x_{ijkt} denotes the material quantity assigned from supplier i to demand j by robot k in batch t. The dynamic cost per unit is

$$c_{ijkt} = f\big(a_i,\ p_k(t),\ b_j,\ t\big) \tag{4}$$

with

- a_i: coordinate of supplier i,
- b_j: coordinate of demand j,
- $p_k(t)$: position of robot k at time t,
- $f(\cdot)$: cost function.

State Definition

At each time t, the state of the system is given by

$$S^{(t)} = \Big(\mathbf{s}^{(t)},\, \mathbf{d}^{(t)},\, \mathbf{p}^{(t)}\Big),$$

where:

$$\mathbf{s}^{(t)} = \big(s_1^{(t)},\, s_2^{(t)},\, \ldots,\, s_{|I|}^{(t)}\big) \quad \text{(residual supplies)},$$

$$\mathbf{d}^{(t)} = \big(d_1^{(t)},\, d_2^{(t)},\, \ldots,\, d_{|J|}^{(t)}\big) \quad \text{(residual demands)},$$

$$\mathbf{p}^{(t)} = \big(p_1(t),\, p_2(t),\, \ldots,\, p_k(t)\big) \quad \text{(positions of robots)}.$$

The initial state is $S^{(1)} = \big(\mathbf{s}^1,\, \mathbf{d}^1,\, \mathbf{p}^1\big)$.

Immediate Cost Calculation

Given a decision $X^{(t)} = \{x_{ijkt}\}$ made at time t, the immediate cost incurred is

$$g\big(S^{(t)}, X^{(t)}\big) = \sum_{i\in I}\sum_{j\in J}\sum_{k\in K} f\big(a_i,\, p_k(t),\, b_j,\, t\big)\, x_{ijkt}.$$

Here, the function f dynamically computes c_{ijkt} based on the current positions $p_k(t)$ and the static locations a_i and b_j.

State Update Operator

After an assignment decision at time t (for instance, via a modified Vogel's method [15]), assume that at time t we allocate

$$\Delta = \min\{s_{i^*}^{(t)},\, d_{j^*}^{(t)},\, T_{k^*}\},$$

for chosen (i^*, j^*, k^*)

Then the state is updated as follows:

$$s_i^{(t+1)} = \begin{cases} s_i^{(t)} - \Delta, & \text{if } i = i^*, \\ s_i^{(t)}, & \text{otherwise,} \end{cases}$$

$$d_j^{(t+1)} = \begin{cases} d_j^{(t)} + \Delta, & \text{if } j = j^*, \\ d_j^{(t)}, & \text{otherwise,} \end{cases} \tag{5}$$

$$p_k(t+1) = \begin{cases} b_{j^*}, & \text{if } k = k^*, \\ p_k(t), & \text{otherwise.} \end{cases}$$

This update is encapsulated in the operator

$$\boxed{S^{(t+1)} = \Phi\Big(S^{(t)},\, (i^*, j^*, k^*, \Delta)\Big).} \tag{6}$$

Dynamic Programming Recurrence

Let $V\big(S^{(t)}\big)$ be the minimal cost-to-go. Then

$$V\big(S^{(t)}\big) = \min_{X^{(t)}\in\mathcal{X}(S^{(t)})} \Big\{ \sum_{i,j,k} c_{ijkt}\, x_{ijkt} + V\big(S^{(t+1)}\big) \Big\}, \tag{7}$$

with terminal condition $V\big(S^{(T+1)}\big) = 0$.

C. Integration with Modified Vogel's Method[15]

At each batch t, we form the cost matrix

$$c_{ijkt} \;=\; f\big(a_i,\, p_k(t),\, b_j,\, t\big) \tag{8}$$

for every supply–robot pair (i, k) and every demand j. We then apply the following extended Vogel heuristic:

1) **Compute Row Penalties.** For each row (i, k) with remaining supply $s_i^{(t)} > 0$, find the two smallest costs among $\{c_{ijkt} \mid d_j^{(t)} > 0\}$:

$$c_{(i,k)}^{(1)} = \min_{j:\, d_j^{(t)}>0} c_{ijkt}, \quad c_{(i,k)}^{(2)} = \min_{\substack{j:\, d_j^{(t)}>0 \\ j\neq j'}} c_{ijkt}. \tag{9}$$

The row penalty is then

$$P_{\text{row}}(i,k) \;=\; c_{(i,k)}^{(2)} \;-\; c_{(i,k)}^{(1)}. \tag{10}$$

2) **Compute Column Penalties.** For each column j with remaining demand $d_j^{(t)} > 0$, find the two smallest costs among $\{c_{ijkt} \mid s_i^{(t)} > 0\}$:

$$c_j^{(1)} = \min_{i,k:\, s_i^{(t)}>0} c_{ijkt}, \quad c_j^{(2)} = \min_{\substack{i,k:\, s_i^{(t)}>0 \\ (i,k)\neq(i',k')}} c_{ijkt}. \tag{11}$$

The column penalty is

$$P_{\text{col}}(j) \;=\; c_j^{(2)} \;-\; c_j^{(1)}. \tag{12}$$

3) **Compare Max Penalties.** Let

$$P_{\text{row}}^* = \max_{(i,k)} P_{\text{row}}(i,k), \qquad P_{\text{col}}^* = \max_{j} P_{\text{col}}(j).$$

4) **Select and Allocate.**
   - If $P_{\text{row}}^* > P_{\text{col}}^*$:
     - Select $(i^*, k^*) = \arg\max_{(i,k)} P_{\text{row}}(i,k)$.
     - Choose the demand $j^* = \arg\min_{j:\, d_j^{(t)}>0} c_{i^*jk^*t}$.
   - Otherwise:
     - Select $j^* = \arg\max_j P_{\text{col}}(j)$.
     - Choose the supply–robot $(i^*, k^*) = \arg\min_{i,k:\, s_i^{(t)}>0} c_{ijkt^*}$.

5) **Quantity Allocation.** Compute the transported volume

$$\Delta \;=\; \min\big\{s_{i^*}^{(t)},\, d_{j^*}^{(t)},\, T_{k^*}\big\},$$

and set

$$x_{i^*j^*k^*t} \;=\; \Delta.$$

6) **State Update.** Apply the update operator

$$S^{(t+1)} \;=\; \Phi\big(S^{(t)},\, (i^*, j^*, k^*, \Delta)\big).$$

This process is repeated within the batch t until all available forklift robot's capacity is used or the residual supply/demand is exhausted. Then the robots' positions (and capacities) are updated for the next batch.

V. Proposed solution architecture

Figure 1 represent our proposed architecture. We separate the OT layer and OTP planner will plan the free collision path for each robot. Robot will use path planner and trajectory tracking controller to avoid static and dynamic obstacles.

The behavior of the algorithm is summarized in algorithm 1.

VI. Numerical experimentation

A. Simulation Setup and Notation

We validate our framework using three robots of varying capacities on a 25×15 grid (1m resolution). Cell types (free space, surplus, deficit, obstacles, robot start, turning points, idle nodes) and their color codes are given in Table I.

For planning missions, robots employ BFS or Dijkstra for path planning to quickly compute feasible paths while accounting for vehicle constraints. The goal is to transfer material from surplus blocks to deficit blocks without collisions and within loading constraints.

Algorithm 1: k-OTP Central Planner

Data: *token* (stores current paths, tasks, and robot states)
Data: $\mathcal{T}$ (set of available transport tasks)
Data: Procedure *UpdateWorkingPoints*(r_i, *token*) returns new tasks after load/unload

Initialize *token* with each robot r_i's current position $\langle loc(r_i) \rangle$;
while *true* **do**
  `// Include new tasks into global task pool`
  Add new tasks (if any) to $\mathcal{T}$;
  `// Collision checking and conflict resolution`
  $\mathcal{R} \leftarrow CheckCollisions(token)$;
  foreach *robot* $r_i \in \mathcal{R}$ **do**
    `// Resolve collisions using Conflict-Based Search`
    Resolve collisions and update paths in *token*;
  end
  `// Task assignment and path planning`
  foreach *robot* r_i *requesting token* **do**
    $(src, dst, q) \leftarrow \text{ASSIGN}(r_i)$;
    if $(src, dst, q) \neq$ None **then**
      `// Plan path for loading then unloading`
      $\pi_i^{\text{load}} \leftarrow \text{PATHPLANNER}(r_i, src, token)$;
      $\pi_i^{\text{unload}} \leftarrow \text{PATHPLANNER}(r_i, dst, token)$;
      if $\pi_i^{load}, \pi_i^{unload} \neq$ None **then**
        `// Update token with combined paths`
        $\pi_i \leftarrow \pi_i^{\text{load}} \| \pi_i^{\text{unload}}$;
        Update *token* with (π_i, r_i);
      end
      else
        `// If path planning failed, robot stays idle`
        $\pi_i \leftarrow \langle loc(r_i) \rangle$;
        Update *token* with (π_i, r_i);
      end
    else
      `// No task: robot stays idle`
      $\pi_i \leftarrow \langle loc(r_i) \rangle$;
      Update *token* with (π_i, r_i);
    end
    `// Update environment after loading/unloading`
    $\mathcal{T}_{\text{new}} \leftarrow UpdateWorkingPoints(r_i, token)$;
    if $\mathcal{T}_{new} \neq \varnothing$ **then**
      $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_{\text{new}}$;
    end
  end
  `// Execute robot movements`
  Robots move along their paths in *token* for one timestep;
end

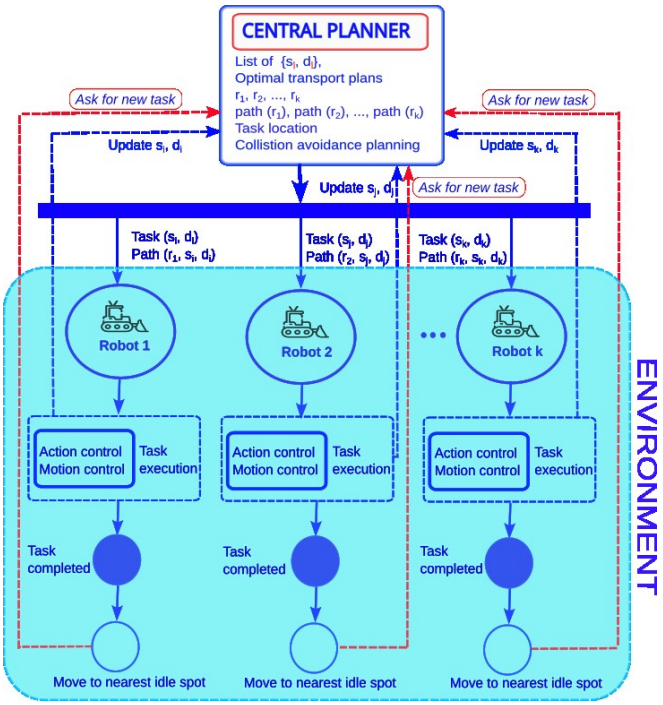

Fig. 1: Multi robots framework architecture for Optimal Transport

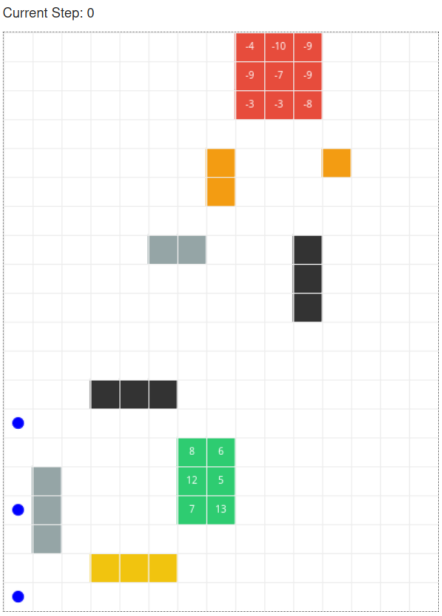

Fig. 2: Starting scenario. White cells represent free locations. Cell annotations indicate height deviation.

TABLE I: Initial layout of the surface in Fig 2, with each cell's value representing its height deviation from the target.

Color / Symbol	Meaning
	Robot's initial position
	Surplus cell (flow: left → right)
	Deficit cell (flow: bottom ↑ top)
	Turning points
	Idle / parking position
	Static obstacle

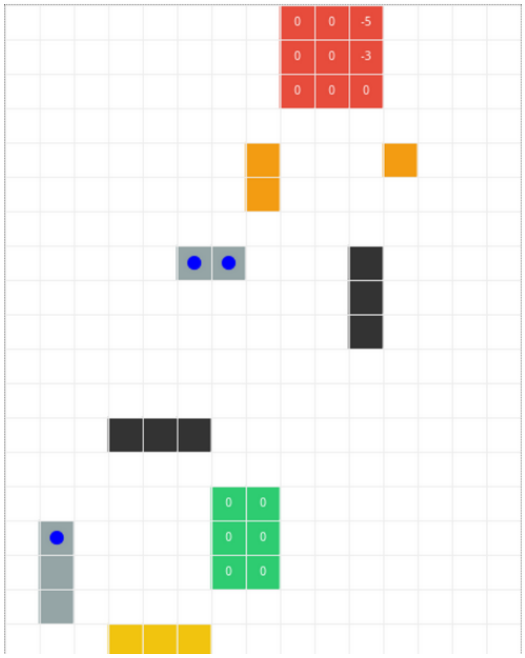

Fig. 3: Filling the targeted surface

Figure 3 proves that following the plan transport plan result in a level ground in both surplus region and deficit region.

For a visual demonstration of the robots in actions: task assignment, path planning, and terrain leveling, see the accompanying video at `https://youtu.be/j_iTxdf22x8`.

Fig. 4: Industrial robot manipulating terrain's surface

VII. CONCLUSIONS

In this work, we introduced a framework for multi-robot material transportation, encompassing task allocation, trajectory planning, and sub-goal definition. Our approach coordinates a fleet of robots in partially known or dynamic environments, guiding each vehicle through consistent, feasible paths toward identified surplus and deficit locations. By explicitly specifying sub-problems such as trajectory design and task bidding strategies, this framework provides a systematic methodology adaptable to various multi-robot scenarios. Future research directions include extending this framework to accommodate heterogeneous robotic systems, integrating drones and diverse types of mobile robots to leverage their complementary capabilities. Additional work could focus on developing cooperative control strategies for these heterogeneous fleets, real-time adaptive path-planning in highly dynamic conditions, and incorporating machine learning techniques for improved task allocation and environment adaptability.

REFERENCES

[1] M. Mansouri, H. Andreasson, and F. Pecora, "Hybrid reasoning for multi-robot drill planning in open-pit mines," *Acta Polytechnica*, vol. 56, no. 1, pp. 47–56, 2016.

[2] L. Chen, Y. Xie, Y. He, Y. Ai, B. Tian, L. Li, S. Ge, and F.-Y. Wang, "Autonomous mining through cooperative driving and operations enabled by parallel intelligence," *Communications Engineering*, vol. 3, no. 1, p. 75, 2024.

[3] J. Xie, S. Liu, and X. Wang, "Framework for a closed-loop cooperative human cyber-physical system for the mining industry driven by vr and ar: Mhcps," *Computers & Industrial Engineering*, vol. 168, p. 108050, 2022.

[4] H. Ma, J. Li, T. Kumar, and S. Koenig, "Lifelong multi-agent path finding for online pickup and delivery tasks," *arXiv preprint arXiv:1705.10868*, 2017.

[5] M. Čáp, J. Vokřínek, and A. Kleiner, "Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures," in *Proceedings of the international conference on automated planning and scheduling*, vol. 25, 2015, pp. 324–332.

[6] M. Cirillo, F. Pecora, H. Andreasson, T. Uras, and S. Koenig, "Integrated motion planning and coordination for industrial vehicles," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 24, 2014, pp. 463–471.

[7] W. Hönig, T. Kumar, L. Cohen, H. Ma, H. Xu, N. Ayanian, and S. Koenig, "Multi-agent path finding with kinematic constraints," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 26, 2016, pp. 477–485.

[8] L. Wen, Y. Liu, and H. Li, "Cl-mapf: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints," *Robotics and Autonomous Systems*, vol. 150, p. 103997, 2022.

[9] D. Silver, "Cooperative pathfinding," in *Proceedings of the aaai conference on artificial intelligence and interactive digital entertainment*, vol. 1, no. 1, 2005, pp. 117–122.

[10] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial intelligence*, vol. 219, pp. 40–66, 2015.

[11] Z. A. Ali and K. Yakovlev, "Prioritized sipp for multi-agent path finding with kinematic constraints," in *Interactive Collaborative Robotics: 6th International Conference, ICR 2021, St. Petersburg, Russia, September 27–30, 2021, Proceedings 6*. Springer, 2021, pp. 1–13.

[12] F. Pecora, H. Andreasson, M. Mansouri, and V. Petkov, "A loosely-coupled approach for multi-robot coordination, motion planning and control," in *Proceedings of the International Conference on Automated Planning and Scheduling*, vol. 28, 2018, pp. 485–493.

[13] Z. Zhou, W. Tang, M. Li, J. Zhang, and X. Wu, "A novel cooperative path planning method based on ucr-fce and behavior regulation for large-scale multi-robot system," *Applied Intelligence*, vol. 53, no. 24, pp. 30 706–30 745, 2023.

[14] Y. Miyashita, T. Yamauchi, and T. Sugawara, "Distributed planning with asynchronous execution with local navigation for multi-agent pickup and delivery problem," *arXiv preprint arXiv:2302.09250*, 2023.

[15] N. V. Reinfeld and W. R. Vogel, "Mathematical programming," *(No Title)*, 1958.