International Journal of Control, Automation, and Systems 22(8) (2024) 2537-2550
http://dx.doi.org/10.1007/s12555-023-0745-4
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

Cluster-based Multi-robot Task Assignment, Planning, and Control

Yifan Bai*, Björn Lindqvist, Samuel Nordström, Christoforos Kanellakis, and George Nikolakopoulos

Abstract: This paper presents a complete system architecture for multi-robot coordination for unbalanced task assignments, where a number of robots are supposed to visit and accomplish missions at different locations. The proposed method first clusters tasks into clusters according to the number of robots, then the assignment is done in the form of one-cluster-to-one-robot, followed by solving the traveling salesman problem (TSP) to determine the visiting order of tasks within each cluster. A nonlinear model predictive controller (NMPC) is designed for robots to navigate to their assigned tasks while avoiding colliding with other robots. Several simulations are conducted to evaluate the feasibility of the proposed architecture. Video examples of the simulations can be viewed at https://youtu.be/5C7zTnv2sfo and https://youtu.be/-JtSg5V2fTI?si=7PfzZbleOOsRdzRd. Besides, we compare the cluster-based assignment with a simulated annealing (SA) algorithm, one of the typical solutions for the multiple traveling salesman problem (mTSP), and the result reveals that with a similar optimization effect, the cluster-based assignment demonstrates a notable reduction in computation time. This efficiency becomes increasingly pronounced as the task-to-agent ratio grows.

Keywords: Autonomous robots, Hungarian algorithm, multi-robot systems, task assignment.

1. INTRODUCTION

Multi-robot systems play a pivotal role in industrial automation, finding applications in surveillance [1], formation control [2,3], search and rescue [4,5], and warehouse pick-up and delivery [6]. An integral concern within these systems is the task of assigning a set of tasks distributed across various locations to multiple agents, while concurrently ensuring collision-free paths for each agent—referred to as Multi-robot task assignment and path-finding (TAPF). Despite its significance, both multi-agent path finding (MAPF) [7] and multi-robot task assignment (MRTA) [8]—representing relaxation variations of TAPF—have been established as NP-hard problems. While MRTA focuses on task allocation to agents under the assumption of no collisions between them, MAPF aims to determine collision-free paths for agents assigned unique tasks. Consequently, the amalgamation of MRTA and MAPF in TAPF renders it even more challenging. This paper endeavors to address this challenge by proposing a method tailored for real-world multi-robot applications.

1.1. Related works

A considerable amount of literature has been published on MRTA and MAPF [9].

Multi-robot task assignment: A large amount of algorithms for task assignment is based on the market mechanism, where individual robot bids for tasks in the auction by communicating with each other. There could be a central auctioneer to assign tasks to agents or the agents make decisions with local negotiation protocol distributively [10]. Besides, MRTA is always modeled as the multiple traveling salesman problem (mTSP), which can be formulated as mixed integer linear programming to find a deterministic solution, or approximate solution with meta-heuristic methods such as simulated annealing [11], ant colony optimization [12], etc.

Multi-agent path finding: The typical taxonomy of MAPF includes two types of coupled path planning and decoupled path planning.

In the matter of coupled method, complete path planning algorithms like A^* leads to the exponential growth of the configuration space as the number of agents rises.

Manuscript received November 9, 2023; revised February 20, 2024 and May 7, 2024; accepted May 10, 2024. Recommended by Associate Editor Quoc Van Tran under the direction of Editor-in-Chief Hyo-Sung Ahn. This work is part of the research project SP14 'Autonomous Drones for Underground Mining Operations', which is included in the Sustainable Underground Mining (SUM) project academic programme spanning from 2021 to 2024. The programme is jointly financed by LKAB and the Swedish Energy Agency. Additionally, this work has been partially funded by the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No. 101003591 Nexgen SIMS.

Yifan Bai, Björn Lindqvist, Samuel Nordström, Christoforos Kanellakis, and George Nikolakopoulos are with Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, SE-97187 Luleå, Sweden (e-mails: {yifan.bai, bjorn.lindqvist, samuel.karlsson, christoforos.kanellakis, geonik}@ltu.se).
* Corresponding author.

Conflict-based search (CBS) [13] introduces constrained single path planning to scale down the search space: CBS discovers conflicts and generates constraints to resolve the conflicts at the high level, and at the low level, conducts constrained A^* for each agent in accordance to the generated constraints. Some sub-optimal variants of CBS, such as enhanced conflict-based search (ECBS) [14], improved conflict-based search [15] have been investigated as well. M^* algorithm [16] plans path for each agent separately and only couples those that will collide, thus declining the growth of configuration space. This approach is termed *sub-dimensioanl expansion*.

As a comparison, decoupled path planning may sacrifice the optimality and completeness, but provides scalability and robustness. Velocity obstacle (VO) method [17] reactively resolves potential conflicts by choosing a velocity out of the *velocity obstacle*, which is the set of all velocities of a robot that will result in a collision with other robots. Reciprocal velocity obstacle [18] is an improvement of VO that solves the oscillation problem of VO. In [19], agents are considered as dynamic obstacles, the dynamics of which are formulated as constraints and fed to the nonlinear model predictive controller (NMPC) for guaranteeing collision avoidance.

Integrated task assignment and path finding: To date, the TAPF problem has been researched from different perspectives. while many algorithms are confined to scenarios where the number of agents is the same as the number of tasks, referred as *balanced assignment*. For instance, concurrent assignment and planning of trajectories (CAPT) [20] solves assignment with the Hungarian algorithm and guarantees collision-free trajectories if the optimization objective is the sum of squared distances of agents rather than the sum of distances. However, CAPT assumes the environment is obstacle-free. CBS-TA [21] finds optimal assignment and path in obstacle-ridden environment by constructing a *conflict forest*, but still only for balanced assignment. Recent literature has emerged that provides approaches for unbalanced task assignment and path finding, that assign a set of tasks to each agent and then determine the internal visiting order of the tasks [22-25].

1.2. Contributions

In this paper, we present a novel hybrid framework for multi-robot task assignment and pathfinding that diverges from traditional methodologies by prioritizing time efficiency and online, reactive obstacle avoidance. This approach uniquely integrates task assignment with multi-agent pathfinding, focusing not on achieving the minimum total travel distance, but on enhancing the system's adaptability and responsiveness to dynamic environmental changes.

Our framework (Fig. 1) employs a cluster-based Hungarian algorithm for distributing tasks among agents in an offline phase, while a nonlinear model predictive controller (NMPC) scheme enables dynamic collision avoidance in an online phase during the mission runtime. Leveraging the strengths of both phases, our framework achieves time efficiency, while adeptly adapting to dynamic environments with disturbances.

The rest of the article is organized as follows: In Section 2 mathematical formulation of the assignment problem is discussed. Section 3 proposes the integrated framework for task assignment, path planning, and nonlinear model predictive control for the multi-robot system. Section 4 demonstrates the experimental evaluation of the proposed framework in Gazebo simulation, which proves the feasibility and discusses the efficiency of the presented work. Finally, Section 5 presents the conclusions and open problems for future work.

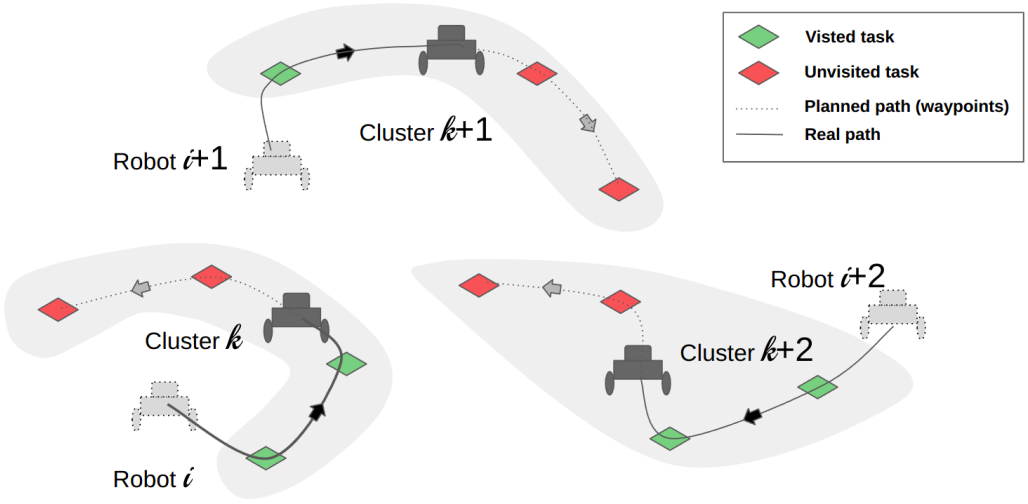

Fig. 1. A conceptual visualization of the cluster-based task assignment.

2. PROBLEM DESCRIPTION

Typically, the number of available robots is less than the number of tasks, thus we consider N ground robots navigating from their N unique depots to M desired goals ($M \geq N$) in a known 2-dimensional environment. The objective is to plan paths for the agents so that every goal can be visited and the total distance of the paths is minimized.

This problem can be generalized as the multiple depot multiple traveling salesman problem (MDMTSP) [26]. We consider a graph $G = (V, E)$, where V denotes vertices and the set E is its arcs. Let $D = \{d_1, d_2, \ldots, d_n\}$ represent the set of depots and $T = \{t_1, t_2, \ldots, t_m\}$ as the set of goal positions. $V = D \cup T$. We use an integer decision variable x_e^k to denote the number of times that robot k traverse edge e in a feasible solution. $e \in E$, $x_e^k \in \{0, 1, 2\}$ and its not allowed to have edge connecting depots. Another decision variable y_i^k, $\forall i \in T$, $k \in D$ is defined, and $y_i^k = 1$ if a target t_i is assigned to tour originating from depot d_k. The objective function is as follows:

$$\text{minimize} \sum_{k=1}^{N} \sum_{e \in E} c_e x_e^k, \tag{1}$$

s.t.

$$\sum_{e \in E} x_e^k = 2y_i^k, \ \forall e \in \{(i,j) : i \in T, \ j \in D\}, \quad k \in \{1, \ \ldots, \ n\}, \tag{2}$$

$$\sum_{e \in E} x_e^k \geq 2y_i^k \ \forall e \in \{(i,j) : i \in S, \ S \subseteq T, \ j \in V \setminus S\}, \quad k \in \{1, \ \ldots, \ n\}, \tag{3}$$

$$\sum_{k=1}^{n} y_i^k = 1 \ \forall i \in T, \tag{4}$$

$$x_e^k \in \{0, \ 1, \ 2\} \ \forall e \in \{(d_k, j) : j \in T\}, \quad k \in \{1, \ \ldots, \ n\}, \tag{5}$$

$$x_e^k \in \{0, \ 1\} \ \forall e \in \{(i,j) : i \in T, \ j \in T\}, \quad k \in \{1, \ \ldots, \ n\}, \tag{6}$$

$$y_i^k \in \{0, \ 1\} \ \forall i \in T, \ k \in \{1, \ \ldots, \ n\}, \tag{7}$$

where c_e is the traversal cost for edge e, constraint (2) guarantees that the number of edges of robot k incident on a target $i \in T$ is 2 if and only if target i is visited by robot k, (3) is the sub-tour elimination and (4) indicates any robot visits each target i. Equations (5)-(7) are decision variables.

3. METHODOLOGY

To address the previously mentioned TAPF problem, we propose a feasible solution composed of clustering-based task assignment, path planning, and model predictive control modules, as shown in Fig. 2. Firstly, M goal positions are clustered into N groups with the hierarchical clustering method, such that the Hungarian algorithm is capable of assigning the N goal clusters to N agents. Then, a TSP solver is deployed to determine the visiting orders of goals in each cluster. During the process, D_+^* is called multiple times to plan the path and calculate the traversal cost of each path. Besides, we leverage NMPC to ensure path following and collision avoidance among robots. The rest of this section elaborates on each module of this architecture.

3.1. Cluster-based assignment

Given M desired goal positions, we want to cluster them into N groups ($M \geq N$) so as to make a balanced assignment for N agents. Thus, we introduce the concept of hierarchical clustering [27], which is a cluster-

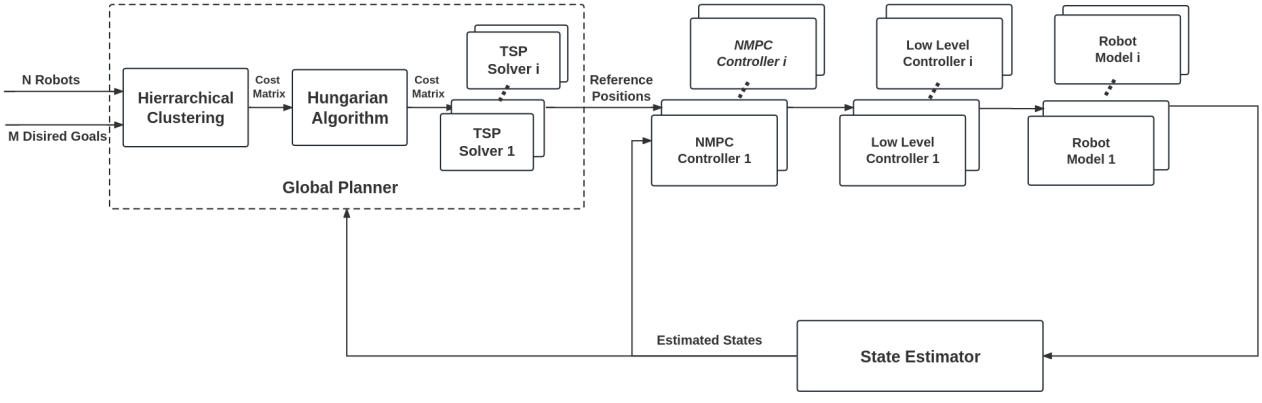

Fig. 2. Block diagram of the assignment-planning-control architecture for $M \geq N$. The dashed part highlights the contribution of the global planner, and the overall scheme demonstrates the integration of the global planner with the NMPC control of each aerial platform.

Algorithm 1: Hierarchical clustering (agglomerative).

Input : T : List of goal positions $T = \{T_1, \cdots, T_M\}$
M : Number of goal positions
N : Number of clusters to find
Output: C : List of N clusters

Declare α, cost matrix of M by M doubles
for $i = 1$ **to** M **do**
  for $j = 1$ **to** M **do**
    if $i = j$ **then**
      $\alpha(i, j) = 0.00$
    else
      $\alpha(i, j) = D_+^*.ComputeCostT(i), T(j))$
    end if
  end for
end for
for i=1 to M **do**
  $c_i = \{T_i\}$
end for
$C = \{c_1, \cdots, c_M\}$
while $C.size() > N$ **do**
  $(c_{min1}, c_{min2}) =$ minimum cost(c_i, c_j) for all c_i, c_j in C
  remove c_i and c_j from C
  add (c_{min1}, c_{min2}) to C
  update α
end while

ing analysis method that allows different metrics to measure the dissimilarity between sets of observations, including Euclidean distance, Manhattan distance, Mahalanobis distance, etc. Here, D_+^* (Subsection 3.3) cost is used to express the dissimilarity between two positions. We achieve an agglomerative-type clustering with scikit-learn, an open-source library for clustering in Python. As depicted in Algorithm 1, the agglomerative hierarchical clustering starts by treating each data point as a separate cluster. Then, it iteratively merges the two closest clusters and updates the distance matrix between clusters until the desired number of clusters is formed. We take the average linkage function to measure the distance between two clusters

$$D(X,Y) = \frac{1}{N_X \times N_Y} \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} d\left(x_i, y_j\right), \quad x_i \in X,\ y_i \in Y, \tag{8}$$

where $N_x \in \mathbb{Z}^+$ and $N_Y \in \mathbb{Z}^+$ are the numbers of objects in clusters X and Y respectively, $d\left(x_i, y_j\right)$ is the traversal distance between objects x and y.

After the hierarchical clustering, the assignment problem is transformed into a linear balanced assignment, in which the number of agents N is equal to the number of task groups N. The Hungarian algorithm is capable of solving this problem with the computational complexity $O\left(n^3\right)$ [28]. The pseudo-codes cost matrix generation and the Hungarian algorithm are depicted in Algorithm 2.

Algorithm 2: The Hungarian algorithm.

Input : A : List that store all start positions
T : List that store all goal positions
M : Number of goal positions
N : Number of agents
C : Clustering result
Output : λ : Assignment result

Declare β, cost matrix of N by N doubles
if $M > N$ **then**
  call Algorithm 1
  for all $i = 1$ **to** N **do**
    for all $j = 1$ **to** N **do**
      $\beta(i, j) = EuclideanDistance(A(i), \bar{C}(j))$
    end for
  end for
else if $M = N$ **then**
  for all $i = 1$ **to** N **do**
    for all $j = 1$ **to** N **do**
      $\beta(i, j) = D_+^*.ComputeCost(A(i), T(j))$
    end for
  end for
end if
$\lambda = \beta$
for all $i = 1$ **to** N **do**
  $\lambda(i,:) = \lambda(i,:) - min(\lambda(i,:))$ {subtract row minima for all elements in the row}
end for
for all $i = 1$ **to** N **do**
  $\lambda(:, j) = \lambda(:, j) - min(\lambda(:, j)$ {subtract column minima for all elements in the column}
end for
Declare a integer $\sigma = 0$ that counts the marked zeros
Declare a list $r_m = []$ that will store the marked rows of λ
Declare a list $c_m = []$ that will store the marked columns λ
while $\sigma < size(\lambda, 0)$ **do**
  $r_m, c_m = MarkMatrix(\lambda)$
  $\sigma = len(r_m) + len(c_m)$
  if $\sigma < size(\lambda, 0)$ **then**
    $\lambda = AdjustMatrix(\lambda, r_m, c_m)$
  end if
end while

Some matrix operation functions to be clarified are the following ones. The $D_+^*.ComputeCost(A(i), T(j))$ is a function of D_+^* that calculates the minimum traversal cost from position A_i to position T_j; the *MarkMatrix* function marks as few rows and columns as possible to cover all zeros in the input matrix and returns two lists r_m c_m containing indices of marked rows and marked columns respec-

tively; the *AdjustMatrix* function takes the current matrix λ and r_m, c_m as input, subtracts the lowest unmarked element from every unmarked element and adds it to the elements that are marked twice.

3.2. Traveling salesman problem

Some agents that are assigned multiple goals are required to visit all the goals at the minimum cost. This can be modeled as a traveling salesman problem (TSP). Assume an agent is required to visit each of n targets, indexed by $1, \cdots, n$. In this case, the agent leaves from an initial start position indexed by 0, visiting each of the n other target positions exactly once in the shortest route. This traveling salesman problem can be formulated as an integer linear program [29], equivalent to

$$\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} d_{ij} x_{ij} \tag{9}$$

over the set determined by the constraints

$$\begin{aligned}
&x_{ij} \in \{0,1\} \ (i,\ j = 1,\ \ldots,\ n),\\
&\sum_{\substack{i=0\\ i\neq j}}^{n} x_{ij} = 1 \ (j = 1,\ \cdots,\ n),\\
&\sum_{\substack{j=0\\ j\neq i}}^{n} x_{ij} = 1 \ (i = 1,\ \cdots,\ n),\\
&u_i - u_j + p x_{ij} \leq p - 1 \ (1 \leq i \neq j \leq n),
\end{aligned}$$

where d_{ij} $(i \neq j = 0, 1, \cdots, n)$.

The mathematical formulation is solved by the optimizer CPLEX [30], which takes the cost matrix γ as input and outputs the visiting order of the goal positions. γ is constructed with D^*_+ cost between any two of the agents' positions and goals' positions. As the agent is not required to return to the start position at last, the cost from each goal position to the starting one $d_{i,0}$ is set to zero.

3.3. Path planning

In terms of path planning, A^* is a best-first search algorithm that aims to find the smallest cost path based on a heuristic function. D^*lite [31] is an incremental heuristic search algorithm, which outperforms A^* in terms of the capability of re-planning when traversing unknown maps. However, D^*lite ignores the physical shape of robots and considers them as particles, which may lead to waypoints adjustments to obstacles and potential collisions. D^*_+ [32] categorizes voxels into free voxels v_f, occupied voxels v_o and a new type of voxels, unknown voxels v_u, in light of sensor imperfection and map sparsity. A risk layer that increases the traversal cost for voxels in the proximity of occupied voxels is introduced to generate a moderate path, as seen in Fig. 3.

More specifically, v_f and v_o has minimum and maximum traversal cost c_f and c_o, respectively. The traversal

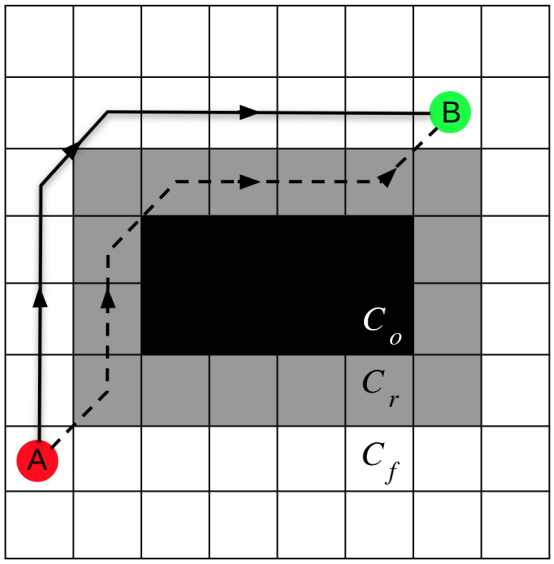

Fig. 3. The black zone is an impenetrable obstacle. The grey zone indicates areas of risk, avoided by the D^*_+ planner. The dashed line shows the shortest path, while the solid line depicts the safer, risk-averse route from point A to B.

cost c_u for v_u can be set in between c_f and c_o. For voxels that are within r distance around the occupied voxels, an extra traversal cost $C_r = C_u/(d+1)$ will be added to its cost C, where d is the distance to the v_o counted in voxels.

In this case, the planner will generate a moderate path with a safety margin next to obstacles. Thus, in this article and without a loss of generality we have selected the D^*_+ path planner.

3.4. Nonlinear model predictive control

With the proposed assignment-planning modules, each robot has a predetermined path to reach the assigned goal position. In the course of trajectory tracking, we use a nonlinear model predictive controller, which has the ability to anticipate future events and take actions accordingly and has been successfully used for multi-agent systems as [33] and [34].

The nonlinear model [35] of the differential drive robot can be formulated as

$$\begin{aligned}
&\dot{p}_x(t) = \cos\psi(t) u_v(t),\\
&\dot{p}_y(t) = \sin\psi(t) u_v(t),\\
&\dot{\psi}(t) = u_\omega(t),
\end{aligned}$$

where $p = [p_x,\ p_y]^\top$ and ψ are the global position and heading angle of the robot, respectively. The states of the robot are $x = [p_x,\ p_y,\ \psi]$ and the control input is $u = [u_v,\ u_\omega]$, where u_v denotes a linear velocity command and u_w is an angular velocity command.

As the control objective is to track reference waypoints with a minimal actuation effort and actuation variation, an objective function $J(\boldsymbol{x}_k, \boldsymbol{u}_k; u_{k-1|k})$ is formed that penalizes the state deviation from the references, inputs costs, deviation in consecutive inputs.

$$\begin{aligned} J\left(\boldsymbol{x}_k, \boldsymbol{u}_k; u_{k-1|k}\right) = \sum_{j=0}^{N-1} \Big(& \underbrace{\left\| x_{\text{ref}} - x_{k+j|k} \right\| Q_x^2}_{\text{position error}} \\ & + \underbrace{\left\| u_{k+j|k} \right\| Q_u^2}_{\text{Input penalty}} \\ & + \underbrace{\left\| u_{k+j|k} - u_{k+j-1|k} \right\| Q_{\Delta u}^2}_{\text{Input change penally}} \Big), \end{aligned} \tag{10}$$

where $Q_x \in \mathbb{R}^{3\times3}$, Q_u, $Q_{\Delta u} \in \mathbb{R}^{2\times2}$ are positive definite weight matrices that reflect the relative importance of each term in the cost function.

Additionally, concerning the collision avoidance requirements between robots, we consider other robots as dynamic obstacles for the ego robot at $p = [p_x, p_y]$, and model the obstacle as a circular area with center $p_{\text{obs}} = [p_{\text{obs,x}}, p_{\text{obs,y}}]$ and radius r_{obs}. Let

$$\begin{aligned} C_{\text{circle}}(p, p_{\text{obs}}, r_{\text{obs}}) = & \, r_{\text{obs}}^2 - (p_x - p_{\text{obs,x}})^2 \\ & - (p_y - p_{\text{obs,y}})^2. \end{aligned} \tag{11}$$

Collision avoidance between robots is guaranteed as long as $C_{\text{circle}}(p, p_{\text{obs}}, r_{\text{obs}}) \leq 0$, that is other robots treated as obstacles should always lie outside the circle around the ego robot. Banding cost function, obstacle constraint, and input bounds lead to the following nonlinear optimization problem

$$\begin{aligned} & \underset{\boldsymbol{u}_k \boldsymbol{x}_k}{\text{Minimize}} \;\; J\left(\boldsymbol{x}_k, \boldsymbol{u}_k, u_{k-1|k}\right), \\ & \text{subject to } x_{k+j+1|k} = f\left(x_{k+j|k}, u_{k+j|k}\right), j \in \mathbb{N}_{[0,N-1]}, \\ & \qquad u_{\min} \leq u_{k+j|k} \leq u_{\max}, j \in \mathbb{N}_{[0,N-1]}, \\ & \qquad C_{\text{circle}}\left(p, p_{\text{obs}}^i, r_{\text{obs}}\right) \leq 0, j \in \mathbb{N}_{[0,N]}, \\ & \qquad i \in \mathbb{N}_{[1,N_{\text{obs}}]}. \end{aligned} \tag{12}$$

The NMPC problem can be solved with an embedded solver generated by optimization engine (OpEn), where proximal averaged Newton-type method for optimal control (PANOC) is used to compute collision-free trajectories [33].

3.5. Reactice collision avoidance

As shown in function (8), hierarchical clustering requires a priori global map to get the travel distance between tasks to cluster. However, there might be some deviation from the priori map to reality or some dynamic obstacles. Thus, robots need local avoidance scheme for robustness. We propose an artificial potential field (APF) that generates a continuous repulsive force with saturation limits within the influence area of the potential field of obstacles.

We denote the local point recognized by the 2D laser as $\{P\}$, where each point is described by a relative Cartesian (x, y) format position to the laser frame as $p = [p_x, p_y]^\top$, converted from a Polar coordination format $p = [r, \theta]$. The subset of such points within the r_F range of the robot $\boldsymbol{\rho}_F \in \{P\}$ are the points that we use to generate *linear repulsive force*, which is expressed as

$$F^{r,lin} = \sum_{i=1}^{N_{\rho_F}} L^r \left(1 - \frac{\left\|\rho_F^i\right\|}{r_F}\right)^2 \frac{-\rho_F^i}{\left\|\rho_F^i\right\|}, \tag{13}$$

where r_F is the influence radius of the potential field, $L^r = \text{diag}(L_x^r, L_y^r)$ is the diagonal matrix of repulsive gains.

Besides, a *critical repulsive force* is also imposed to the points $\rho_c \in \{P\}$ where $\left\|\rho_c^i\right\| \leq r_c$ and $i = 0, 1, \ldots, N_{\rho_c}$. The idea of introducing such *critical repulsive force* is to guarantee the avoidance from smaller obstacles whose linear force response would not be sufficiently large to result in a proper avoidance maneuver.

$$F^{r,c} = \sum_{i=1}^{N_{\rho c}} L^c \frac{-\rho_c^i}{\|\rho_c^i\|}, \tag{14}$$

with diagonal matrix L^c as the critical static force-per-point inside r_c.

With only *linear* repulsive force and *critical* repulsive force can robots be pushed away from but not around the obstacles. Thus, we add a weak rotational component in the xy-directions to make the robot move around obstacles, with the magnitude being the maximum of a fraction of the linear force in the other direction (x-y) and f, in case smaller obstacles only generate linear repulsive force in one direction, and the sign of the rotational force is determined by the sign of the linear force in the same direction.

$$\begin{aligned} F_x^{r,rot} &= \max\left(L^{rot} \operatorname{sgn}\left(F_x^{r,\,\text{lin}}\right) \left\|F_y^{r,\,\text{lin}}\right\|, f\right), \\ F_y^{r,rot} &= \max\left(L^{rot} \operatorname{sgn}\left(F_y^{r,\,\text{lin}}\right) \left\|F_x^{r,\,\text{lin}}\right\|, f\right). \end{aligned} \tag{15}$$

The resulting repulsive force is the sum of all these components

$$F^r = F^{r,lin} + F^{r,c} + F^{r,rot}. \tag{16}$$

Subsequently, the reactive control input for the robot is formulated as

$$\begin{aligned} u_v' &= u_v + F_x^r, \\ u_\omega' &= u_\omega + \operatorname{sgn}(F_x^r) \cdot F_y^r, \end{aligned} \tag{17}$$

where u_v, u_ω denote the primary control inputs, which are derived from an optimization function (12). This function is designed to direct the robot towards its subsequent waypoint, functioning analogously to an attractive force. Concurrently, F^r represents the repulsive force, introduced to mitigate collision risks by dynamically adjusting the robot's trajectory in response to proximate obstacles.

Table 1. Cost matrix α for clustering based on D^*_+ traversal cost between goal positions.

Cost \ Target	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
T_1	0.00	140.00	68.28	76.57	169.70	154.57	229.70	217.40
T_2	140.00	0.00	208.28	204.85	128.28	237.16	188.28	240.59
T_3	68.28	208.28	0.00	49.46	221.42	127.46	246.27	223.42
T_4	76.57	204.85	49.46	0.00	184.85	78.00	209.71	173.97
T_5	169.70	128.28	221.42	184.85	0.00	167.46	60.00	124.02
T_6	154.57	237.16	127.46	78.00	167.46	0.00	177.40	107.68
T_7	229.70	188.28	246.27	209.71	60.00	177.40	0.00	86.28
T_8	217.40	240.59	223.42	173.97	124.02	107.68	86.28	0.00

4. EXPERIMENTAL ANALYSIS

4.1. Gazebo simulation

4.1.1 Static environment

We implement cluster-based assignment and NMPC framework in ROS python and conduct several sets of simulations in a custom Gazebo world. Turtlebot3 Burger (https://emanual.robotis.com/docs/en/platform/turtlebot3/overview/) is used as the experimental ground robot. As seen in Fig. 4, twenty 1 m × 1 m × 0.5 m boxes are evenly distributed in a 16 m × 16 m square area. The corresponding 2D occupancy map is considered as a known priori map. All computations of path planner, clustering, and assignment are done on a single Lenovo ThinkPad P14s with an AMD Ryzen 7 pro 5850u CPU, and controllers for turtlebot are run as ROS nodes.

In the first scenario, three turtlebot depart at $A_1 = [-2.0, -5.0]$, $A_2 = [0.0, 0.0]$, $A_3 = [5.0, 4.0]$, respectively. We randomly click eight points as goal positions on the map: $T_1 = [-4.93, -2.03]$, $T_2 = [-4.93, 4.98]$, $T_3 = [-3.97, -5.03]$, $T_4 = [-1.99, -4.01]$, $T_5 = [1.05, 3.99]$, $T_6 = [1.98, -4.01]$, $T_7 = [4.04, 3.97]$, $T_8 = [5.01, 0.01]$.

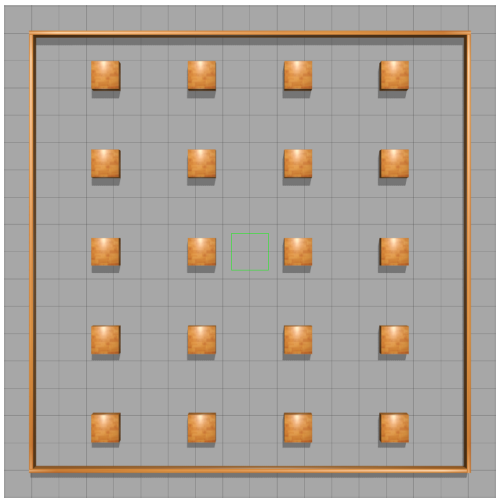

Fig. 4. Top view of the customized Gazebo world.

Table 2. Cost matrix β for assignment using the Hungarian algorithm.

Cost \ Cluster center	G_1	G_2	G_3
Agent			
A_1	1.46	12.91	13.04
A_2	6.00	9.91	6.04
A_3	15.00	10.91	2.96

The D^*_+ planner takes the generated occupancy map of the Gazebo world as input and the safety distance is set to $r = 1$ voxel. Then, we enumerate combinations of the goal positions and publish them to the D^*_+ planner and the paths and traversal costs are obtained. As shown in Table 1, a cost matrix is built with the costs between the goal positions for clustering. As there are three agents, ten tasks are grouped into three clusters by means of hierarchical clustering: $G_1 = \{T_1, T_3, T_4, T_6\}$, $G_2 = \{T_2\}$, $G_3 = \{T_5, T_7, T_8\}$.

Then the cost matrix β Table 2, constructed with the Euclidean distance between each agent and each cluster's mean center, is fed into the Hungarian algorithm for the task assignment. the first group G_1 is assigned to agent A_1, G_2 is allocated to A_2, and G_3 is allocated to A_3.

After assigning the goal clusters to the agents, the visiting order of the goals in each cluster needs to be determined. Table 3 shows the cost matrix γ for G_1-to-A_1 assignment as an example, and the TSP solver result indicates the optimal visiting order: A_1-T_3-T_1-T_4-T_6. Similarly, we can obtain visiting sequences of the other two agents: A_2-T_2 and A_3-T_7-T_5-T_8. During the process of calculating costs with the D^*_+ planner between different positions, the corresponding paths containing waypoints are preserved. By merging the corresponding paths sequentially, the paths of visiting all the assigned goals in order for each agent are obtained. Fig. 5 shows the trajectories of three turtlebot visiting their assigned tasks.

The second scenario shows five turtlebot, starting from $A_1 = [-5.91, 6.69]$, $A_2 = [-2.14, -0.12]$, $A_3 = [1.25,$

Table 3. Cost matrix γ of TSP problem for G_1-to-A_1 assignment.

Start \ Target	A_1	T_1	T_3	T_4	T_6
A_1	0.00	84.02	45.80	20.83	88.28
T_1	0.00	0.00	68.28	76.57	154.57
T_3	0.00	68.28	0.00	49.46	127.46
T_4	0.00	76.57	49.46	0.00	78.00
T_6	0.00	154.57	127.46	78.00	0.00

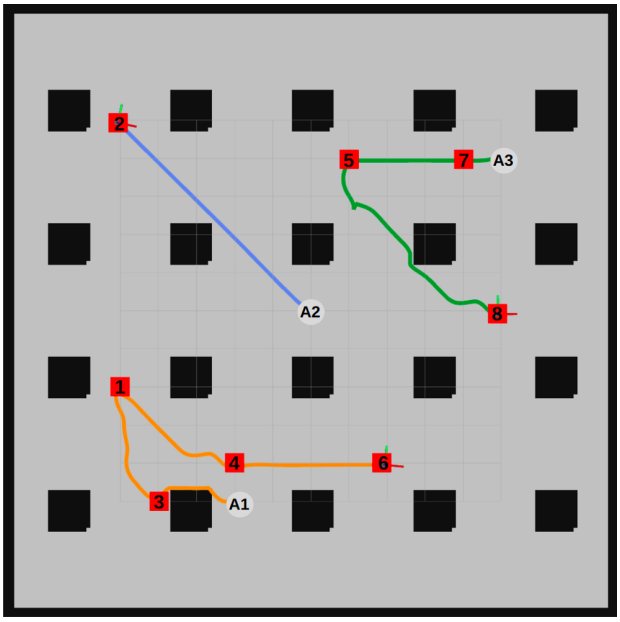

Fig. 5. Trajectory of three turtlebot visiting their assigned tasks.

2.61], $A_4 = [1.26, -5.25]$, $A_5 = [6.21, 3.19]$, respectively, be off to twenty tasks: $T_1 = [-6.25, 3.63]$, $T_2 = [-6.94, -0.08]$, $T_3 = [-4.24, 4.95]$, $T_4 = [-4.52, 2.44]$, $T_5 = [-4.64, -1.43]$, $T_6 = [-5.22, -3.29]$, $T_7 = [-4.51, -5.45]$, $T_8 = [-2.82, -6.64]$, $T_9 = [-0.98, 6.24]$, $T_{10} = [-0.90, 3.95]$, $T_{11} = [-0.41, 0.67]$, $T_{12} = [-0.30, -3.44]$, $T_{13} = [1.73, 3.32]$, $T_{14} = [2.01, 0.13]$, $T_{15} = [2.33, -6.41]$, $T_{16} = [4.08, 6.83]]$, $T_{17} = [4.70, 2.66]$, $T_{18} = [4.78, -1.01]$, $T_{19} = [4.65, -3.05]$, $T_{20} = [4.70, -5.53]$. Cost matrices of the Hungarian algorithm and TSP are omitted for brevity, and the results are shown as follows: the clustering result is $G_1 = \{T_1, T_3, T_4\}$, $G_2 = \{T_9, T_{10}, T_{13}, T_{16}, T_{17}\}$, $G_3 = \{T_2, T_5, T_6, T_7, T_8\}$, $G_4 = \{T_{11}, T_{12}, T_{14}\}$, $G_5 = \{T_{15}, T_{18}, T_{19}, T_{20}\}$, the assignment result indicates G_1-to-A_1, G_2-to-A_5, G_3-to-A_2, G_4-to-A_3, and G_5-to-A_4. With TSP solvers, the visiting order of each turtlebot is regulated as: A_1-T_3-T_4-T_1, A_2-T_2-T_5-T_6-T_7-T_8, A_3-T_{14}-T_{11}-T_{12}, A_4-T_{15}-T_{20}-T_{19}-T_{18}, A_5-T_{17}-T_{13}-T_{10}-T_9-T_{16}. Fig. 6 shows the trajectories of five turtlebot visiting their assigned tasks.

A video demonstration of two scenarios is provided at https://youtu.be/5C7zTnv2sfo.

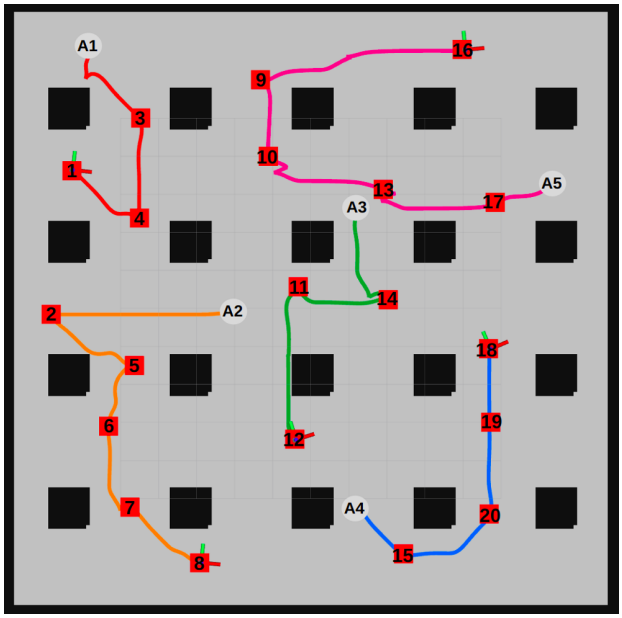

Fig. 6. Trajectory of five turtlebot visiting their assigned tasks.

4.1.2 Dynamic environment

To enhance the validation of our framework's dynamic obstacle avoidance capabilities, we designed a simulated warehouse environment measuring 40 m × 40 m, as depicted in Fig. 7. Within this environment, Gazebo animation model *actors* equipped with collision elements detectable by robots' lasers serve as dynamic obstacles, following predefined trajectories.

We positioned three robots at designated locations: $A_1 = [-2.0, -5.0]$, $A_2 = [0.0, 0.0]$, $A_3 = [5.0, 4.0]$, respec-

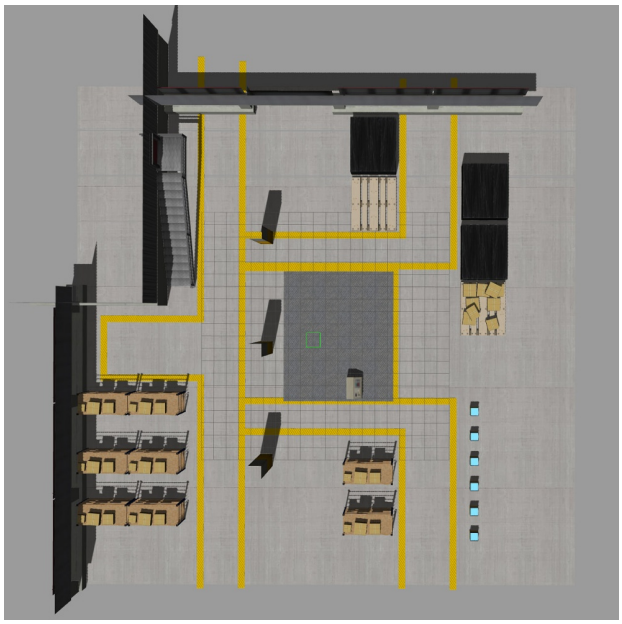

Fig. 7. Top view of the warehouse Gazebo world.

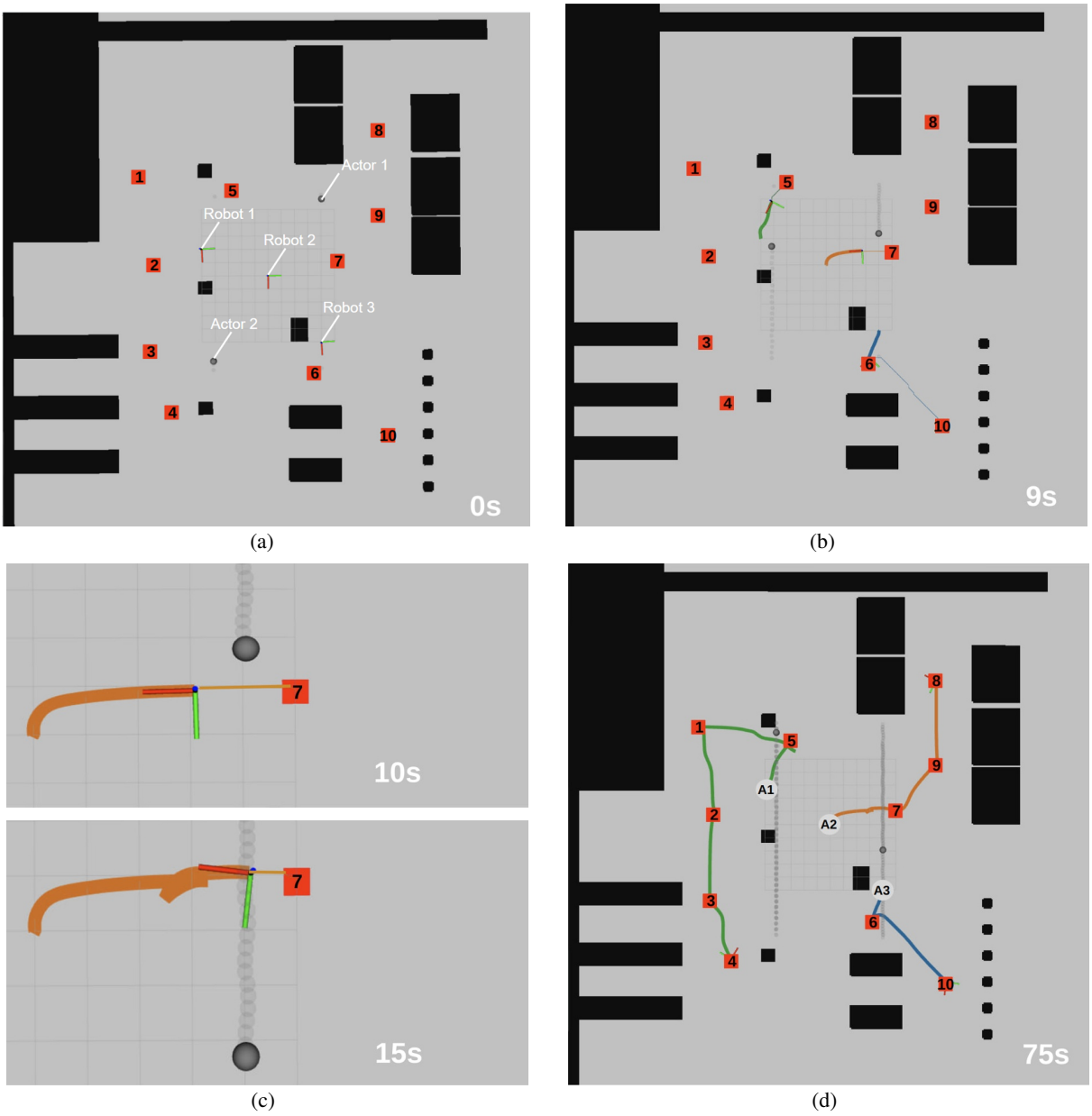

Fig. 8. Robot trajectory adaptation in response to a moving obstacle. Panels (a) and (b) depict initial task locations and robot trajectories at 0 s and 9 s. Panels (c) reflect the robot's actual trajectory (thick line) diverging from its planned path (thin line) at 15 s to avoid an actor. Panel (d) illustrates the trajectories of all robots at 75 s.

tively. Two animated *actor* are employed as dynamic obstacles within the Gazebo framework. The first *actor* starts at $[-6.0, -4.0]$ and moves towards $[7.0, -4.0]$, looping back to its starting position, with specific orientations at each waypoint to mimic a person pacing back and forth. The second *actor* executes a mirrored movement pattern on the opposite side of the simulation space, starting at $[7.0, 4.0]$ and moving to $[-6.0, 4.0]$, also in a looped trajectory, but with a different speed. Ten tasks are randomly selected as $T_1 = [-7.27, -10.36]$, $T_2 = [-3.97, -9.08]$, $T_3 = [5.71, -9.69]$, $T_4 = [10.39, -8.12]$, $T_5 = [-6.43, -2.96]$, $T_6 = [7.42, 3.83]$, $T_7 = [-0.27, 5.07]$, $T_8 = [-11.18, 8.37]$, $T_9 = [-4.80, 8.50]$, $T_{10} = [11.41, 8.67]$.

A 10×10 cost matrix was constructed to cluster these tasks into three groups: $G_1 = \{T_1, T_2, T_3, T_4, T_5\}$, $G_2 = \{T_7, T_8, T_9\}$, $G_3 = \{T_6, T_{10}\}$, which were then assigned to the robots using the Hungarian algorithm. G_1-to-A_1, G_2-to-A_2, G_3-to-A_3. The assigned task sequences for each robot were determined through TSP, resulting in specified visiting orders for each group A_1-T_5-T_1-T_2-T_3-T_4, A_2-T_7-T_9-T_8, A_3-T_6-T_{10}.

Fig. 8 presents a screenshot captured in Rviz, delineating the trajectories of three agents at different timesteps.

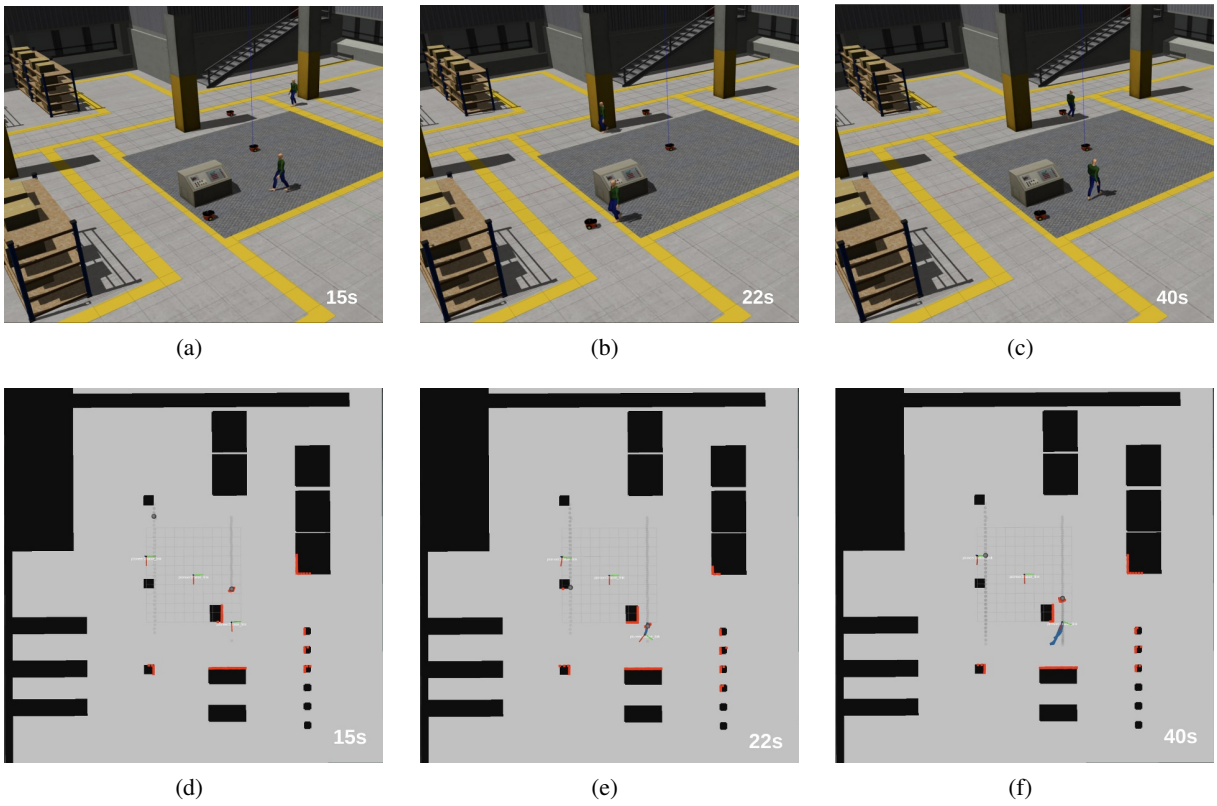

Fig. 9. Sequential demonstration of dynamic obstacle avoidance in a warehouse simulation. Panels (a-c) show the Gazebo environment, and panels (d-f) display the Rviz visualization, capturing robot 3's (blue line) strategic maneuver to evade actor2 (grey sphere) at 22 s and its return to the original position at 40 s. Red dots represent laser detections.

Panels (a), (b), and (d) of the figure sequentially depict the initiation, intermediate stages, and completion of the process wherein each robot navigates towards its designated tasks. The real-time positions of the robots are represented by the red-green axes, while the states of two dynamic obstacles (referred as Actor 1 and Actor 2) are illustrated using grey spheres. The trajectories of these actors are delineated by grey dashed lines. A detailed examination, as shown in Panel (c), elucidates a notable deviation in robot 2's trajectory from its initially planned path, occurring between the timestamps of 10 s and 15 s. This deviation exemplifies an adaptive maneuver executed by robot 2 to circumvent an unexpected encounter with an actor.

To further demonstrate the proficiency of the robots in evading dynamic impediments, a zero-task scenario was established wherein one robot was deliberately positioned along the trajectory of an actor. This intentional setup induced a condition where the robot was required to maintain its stationary position while concurrently circumventing the actor whenever proximity was breached. As depicted in Fig. 9, panels (a), (b), and (c) sequentially represent the states of the Gazebo simulation environment, and panels (d), (e), and (f) correspond to their respective Rviz visualizations, illustrating the odometry of the three robots and the trajectories of two actors (denoted by grey spheres). In panels (a) and (d), the robots are observed to be stationary at their designated coordinates. In panels (b) and (e), one actor enters the repulsive boundary of robot 3, which is evidenced by the laser detection (red dots), prompting robot 3 to retreat to avert a collision. Throughout this maneuver, the robot endeavors to return to its initial position while simultaneously sidestepping the actor. Notably, this iterative process of avoidance and goal reorientation leads to non-smooth movements of the robot. Panels (c) and (f) capture the moment post-encounter, where the actor has vacated the robot's repulsive radius, thereby permitting the robot to reoccupy its original location. The blue line delineates the robot's trajectory, encompassing both its collision avoidance path and its subsequent return.

A video demonstration of inclusion of dynamic obstacles is provided at https://youtu.be/-JtSg5V2fTI?si=7PfzZbleOOsRdzRd. Notably, in the second simulation, the robot is seen deviating from its position to avoid an approaching obstacle and then resuming its course towards the goal once it exits the obstacle's potential field. Nevertheless, the robot re-enters the obstacle's influence range due to its motion, resulting in further deviations. This iter-

ative process of avoidance and goal reorientation leads to the observed non-smooth movements of the robot.

Importantly, the dynamic obstacle avoidance maneuvers showcased do not compromise the results of the task assignment, underscoring the robustness of our proposed framework.

4.2. Computation efficiency

Apart from verifying the feasibility of cluster-based assignments in Gazebo simulations, we also want to evaluate the computation efficiency of the proposed algorithm. As determining a set of routes for N robots starting at different locations to visit M tasks so as to minimize the total travel distance is a typical multi-depot mTSP (multiple traveling salesman problem), solvers for mTSP also fit the problem described in Section 2. Here, we measure the execution time and total travel cost of cluster-based assignment and compare them against that of the simulated annealing (SA) algorithm, a modern solution of mTSP, to compare their computational efficiency and optimization effect. To simplify the problem, test instances are conducted in a 100 m $\times$ 100 m free space with a combination of $N \in \{5, 10, 20\}$ and $M \in \{10, 20, 30, 40\}$. Each instance is run 30 times to collect statistics. As seen in Fig 10, although experiencing slight growth as the number of agents and the number of task rise, the computation time of cluster-based assignment remain at low values from 0.05 s to 0.2 s. While for SA, the computation time rises rapidly from 0.3 s with 5 agents, 10 tasks to nearly 6 s with 20 agents and 40 tasks. As both TSP and mTSP is NP-hard problem, One defect of cluster-based assignment is that it requires more or at least the same number of tasks as agents, thus the data of cluster-based assignment with $N = 20$, $M = 10$ is missing.

In order to analyze the optimized routing result of the cluster-based algorithm, we introduce the *cost ratio* γ that takes SA as a benchmark

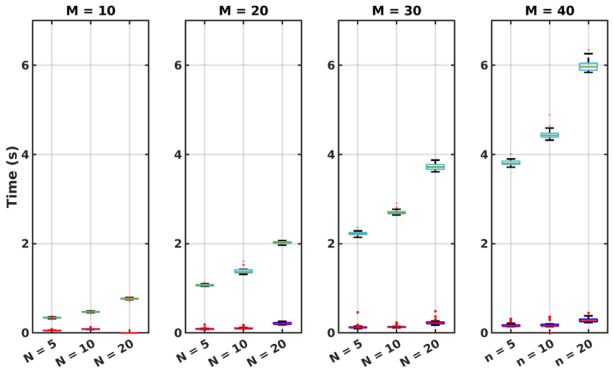

Fig. 10. Run time statistics of the cluster-based assignment (dark blue) and simulated annealing (light blue).

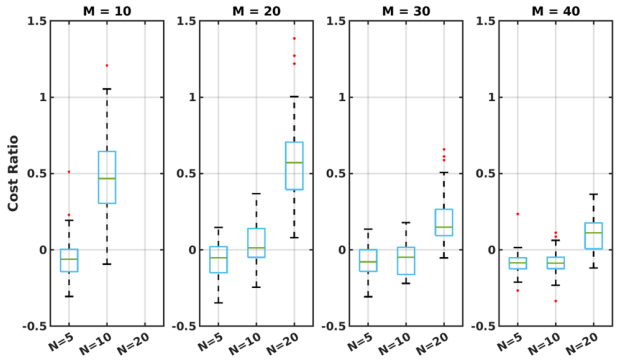

Fig. 11. Cost ratio of the cluster-based assignment compared to simulated annealing.

$$\delta = \frac{\sum C_{Cluster} - \sum C_{SA}}{\sum C_{SA}}, \tag{18}$$

where $C_{Cluster}$ and C_{SA} denote the total travel cost of each robot in the cluster-based assignment and SA, respectively.

Conceivably, $\delta = 0$ indicates the total travel cost of each robot of the two algorithms is the same. Positive δ signifies the outperformance of SA and otherwise cluster-based assignment is better. Fig. 11 depicts the cost ratio of the cluster-based assignment over the simulated annealing algorithm. As aforementioned, $M = 10$, $N = 20$ data is missing. In general, cluster-based assignment finds better routes for robots when the $M - N$ ratio is large. While the total cost of each robot given by SA is much smaller than the cluster-based algorithm when there is the same number of robots and tasks. This is reasonable as the cluster-based assignment distributes one cluster to each robot, resulting in every robot being assigned at least one task regardless of the positions of the robots. However, SA allows not assigning tasks to one robot if it's too far away from all the tasks.

The multi-robot task assignment and path-finding (TAPF) problem investigated in this study is inherently NP-hard, attributed to the extensive array of potential task assignments among robots and the need to mitigate inter-collisions. Both the MRTA and the MAPF problems, which are sub-components of TAPF, are all recognized as NP-hard. Achieving an optimal solution for TAPF is prohibitively time-consuming due to these complexities. The objective within this article is not to attain an optimal solution for TAPF. Instead, our focus is on identifying a near-optimal solution expediently. Our proposed framework addresses the TAPF challenge with a cluster-based assignment algorithm coupled with a TSP solver and the NMPC. Subsequent Figs. 10 and 11 demonstrate that this methodology achieves a solution comparable in cost but with greater efficiency. NMPC offers the advantages of online computation and adaptability, facilitating real-time responses to delays and dynamic changes in the environment.

5. CONCLUSIONS AND FUTURE WORK

In this article, we proposed a novel and complete assignment, planning, and control architecture for a multiple-goal-multiple-robots system, which integrates D^*_+ algorithm, hierarchical clustering, the Hungarian algorithm, traveling salesman problem, and nonlinear model predictive controller.

Tailored for unbalanced assignments where tasks outnumber agents, our experimental results substantiate the architecture's ability to optimize task assignments, minimizing total traversal costs, with all the robots successfully navigating to their goal positions. A comparative analysis with the simulated annealing (SA) algorithm reveals our cluster-based assignment's superior performance in computation time, though it is noted that our approach is specifically designed for scenarios with fewer robots than tasks.

Additionally, we enhanced our system's dynamic obstacle avoidance capabilities by integrating an artificial potential field method as a reactive component. This inclusion ensures that robots can adeptly maneuver around dynamically detected obstacles, validated through precise laser detection. The implementation of this artificial potential field (APF) substantially augments our architecture's robustness, enabling real-time, responsive navigation in complex environments.

Future work will focus on two primary enhancements. Firstly, we aim to integrate predictions of dynamic obstacle trajectories into the model predictive control (MPC) framework to achieve smoother robot trajectories. This improvement is designed to refine collision avoidance strategies, facilitating more fluid navigation in dynamic environments with moving obstacles. Secondly, we plan to extend the current assignment scheme to accommodate online assignments. This will enable the continuous addition of new tasks and allow for dynamic reassignments based on real-time cost evaluations between unvisited tasks and available agents.

CONFLICT OF INTEREST

The authors declare that there is no competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

REFERENCES

[1] M. V. Dileep, B. Yu, S. Kim, and H. Oh, "Task assignment for deploying unmanned aircraft as decoys," *International Journal of Control, Automation, and Systems*, vol. 18, pp. 3204-3217, 2021

[2] D. Koung, I. Fantoni, O. Kermorgant, and L. Belouaer, "Consensus-based formation control and obstacle avoidance for nonholonomic multi-robot system," *Proc. of International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 2020.

[3] Z. Pan, D. Wang, H. Deng, and K. Li, "A virtual spring method for the multi-robot path planning and formation control," *International Journal of Control, Automation, and Systems*, vol. 17, pp. 1272–1282, 2019.

[4] A. Nath, A. R. Arun, and R. Niyogi, "A distributed approach for road clearance with multi-robot in urban search and rescue environment," *International Journal of Intelligent Robotics and Applications*, vol. 3, pp. 392-406, 2019.

[5] J. P. Queralta, J. Taipalmaa, B. C. Pullinen, V. K. Sarker, T. N. Gia, H. Tenhunen, M. Gabbouj, J. Raitoharju, and T. Westerlund, "Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision," *IEEE Access*, vol. 8, pp. 191617-191643, 2020.

[6] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated task assignment and path planning for capacitated multi-agent pickup and delivery," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5816-5823, 2021.

[7] J. Yu and S. M. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," *Proc. of 27th AAAI Conference on Artificial Intelligence*, 2013.

[8] S. Kartik and C. S. R. Murthy, "Task allocation algorithms for maximizing reliability of distributed computing systems," *IEEE Transactions on Computers*, vol. 46, no. 6, pp. 719-724, 1997.

[9] G. M. Skaltsis, H. Shin, and A. Tsourdos, "A survey of task allocation techniques in MAS," *Proc. of International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 488-497, 2021.

[10] H. Zhao, L. Wang, H. Zhou, and D. Du, "Consensus for a Class of Sampled-data Heterogeneous Multi-agent Systems," *International Journal of Control, Automation, and Systems*, vol. 19, pp. 1751–1759, 2021.

[11] Y. Bai, C. Christoforos, and G. Nikolakopoulos, "SAreCBS: Multi-robot task assignment with integrated reactive path generation," *arXiv:2304.02418*, 2023.

[12] C. Jiang, Z. Wan, and Z. Peng, "A new efficient hybrid algorithm for large scale multiple traveling salesman problems," *Expert Systems with Applications*, vol. 139, 112867, 2020.

[13] 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.

[14] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," *Proc. of 7th Annual Symposium on Combinatorial Search*, 2014.

[15] E. Boyarski, A. Felner, R. Stern, G. Sharon, D. Tolpin, O. Betzalel, and E. Shimony, "ICBS: Improved conflict-based search algorithm for multi-agent pathfinding," *Proc. of 24th International Joint Conference on Artificial Intelligence*, 2015.

[16] G. Wagner and H. Choset, "Subdimensional expansion for multirobot path planning," *Artificial Intelligence*, vol. 219, pp. 1-24, 2015.

[17] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using velocity obstacles," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 760-772, 1998.

[18] J. Van den Berg, M. Lin, and D. Manocha, "Reciprocal velocity obstacles for real-time multi-agent navigation," *Proc. of IEEE International Conference on Robotics and Automation*, pp. 1928-1935, 2008.

[19] B. Lindqvist, S. S. Mansouri, A.-A. Agha-mohammadi, and G. Nikolakopoulos, "Nonlinear MPC for collision avoidance and control of UAVs with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6001-6008, 2020.

[20] M. Turpin, N. Michael, and V. Kumar, "CAPT: Concurrent assignment and planning of trajectories for multiple robots," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 98-112, 2014.

[21] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Conflict-based search with optimal task assignment," *Proc. of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '18)*, pp. 757-765, 2018.

[22] V. G. Nair, R. S. Adarsh, K. P. Jayalakshmi, M. V. Dileep, and K. R. Guruprasad, "Cooperative online workspace allocation in the presence of obstacles for multi-robot simultaneous exploration and coverage path planning problem," *International Journal of Control, Automation, and System*, vol. 21, pp. 2338–2349, 2023

[23] X. Zhong, J. Li, S. Koenig, and H. Ma, "Optimal and bounded-suboptimal multi-goal task assignment and pathfinding," *Proc. of International Conference on Robotics and Automation (ICRA)*, pp. 10731-10737, 2022.

[24] Q. Xu, J. Li, S. Koenig, and H. Ma, "Multi-goal multi-agent pickup and delivery," *arXiv preprint arXiv:2208.01223*, 2022.

[25] Z. Ren, S. Rathinam, and H. Choset, "MS*: A new exact algorithm for multi-agent simultaneous multi-goal sequencing and path finding," *Proc. of IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11560-11565, 2021.

[26] S. Kaarthik and S. Rathinam, "An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem," *Proc. of International Conference on Unmanned Aircraft Systems (ICUAS)*, pp. 366-371, 2015.

[27] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241-254, 1967.

[28] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83-97, 1955.

[29] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM*, vol. 7, no. 4, pp. 326-329, 1960.

[30] IBM ILOG Cplex, "V12.1: User's Manual for CPLEX," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[31] S. Koenig and M. Likhachev, "D^* Lite," *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 18, pp. 476-483, 2002.

[32] S. Karlsson, A. Koval, C. Kanellakis, and G. Nikolakopoulos, "D^*_+: A risk-aware platform-agnostic heterogeneous path planner," *Expert Systems with Applications*, vol. 215, 119408, 2023.

[33] X. Tang, M. Li, S. Wei, and B. Ding, "Event-triggered synchronous distributed model predictive control for multi-agent systems," *The Internation Journal of Control, Automation, and Systems*, vol. 19, pp. 1273-1282, 2021.

[34] B. Lindqvist, P. Sopasakis, and G. Nikolakopoulos, "A scalable distributed collision avoidance scheme for multi-agent UAV systems," *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9212-9218, 2021.

[35] M. Kamel, T. Stastny, K. Alexis, and R. Siegwart, "Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system," *Robot Operating System (ROS): The Complete Reference*, vol. 2, pp. 3-39, 2017.

Yifan Bai is currently pursuing a Ph.D. degree at the Robotics and AI team of the Luleå University of Technology, Sweden. His current research interests include developing task assignment and multi-agent path-finding algorithms to enable efficient collaboration of multiple robots.

Björn Lindqvist received his Ph.D. degree from the Robotics and AI team, Luleå University of Technology (LTU), Sweden. He is currently an assistant professor with the Department of Computer Science, Electrical and Space Engineering, LTU. Björn's research interests include collision avoidance and path planning for single and multi-agent systems.

Samuel Nordström is currently pursuing a Ph.D. degree at the Robotics and AI Team at the Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Sweden, working in the field of aerial and ground robotics. Samuel's research interests include path planning and obstacle avoidance, with a focus on real-life verification of methods.

Christoforos Kanellakis received his Ph.D. degree from the Control Engineering Group, Luleå University of Technology (LTU), Sweden. He is currently an assistant professor with the Department of Computer Science, Electrical and Space Engineering, LTU. His research interests include the field of robotics, focusing on the combination of control and vision to enable robots to perceive and interact with the environment

George Nikolakopoulos is acting Chair on Robotics and AI, a Professor on Robotics and Automation at the Department of Computer Science, Electrical and Space Engineering at Luleå University of Technology. His research interests include the area of robotics, control applications, and cyberphysical systems.