Human-Flow-Aware Long-Term Mobile Robot Task Planning Based on Hierarchical Reinforcement Learning

Yuchen Liu, Luigi Palmieri, *Member, IEEE*, Ilche Georgievski, *Member, IEEE*, and Marco Aiello

***Abstract*—The difficulty in finding long-term planning policies for a mobile robot increases when operating in crowded and dynamic environments. State-of-the-art approaches do not consider cues of human-robot-shared dynamic environments. Aiming to fill this gap, we present a novel Human-Flow-Aware Guided Hierarchical Dyna-Q (HA-GHDQ) algorithm, which solves long-term robot task planning problems by using human motion patterns encoded in Maps of Dynamics (MoDs). To tackle the complexity of long-term robot operation in dynamic environments, we propose a combination of symbolic planning and Hierarchical Reinforcement Learning (HRL) that generates robot policies considering cost information derived from MoDs. We evaluated HA-GHDQ in a factory environment with two simulation and one real-world datasets to complete a transportation-and-assembly task. Our approach outperforms the baselines with respect to sample efficiency and final plan quality. Moreover, we show that it is more adaptable and robust against environmental changes than the baselines.**

***Index Terms*—Task planning, integrated planning and learning, human-robot collaboration.**

I. Introduction

NOWADAYS, robots have become important helpers in industrial and household settings [1], [2], [3], [4]. In such environments, they need to constantly take the presence of humans into account. For example, a mobile robot operates in a factory to transport and assemble different parts, while there are humans moving around, as shown in Fig. 1. While completing its tasks, the robot should efficiently navigate and avoid collisions with humans.

This example can be treated as a task and motion planning (TAMP) problem. Two major classes of task planning techniques are the learning-based and model-based approaches [5].

Manuscript received 18 December 2022; accepted 24 April 2023. Date of publication 29 May 2023; date of current version 5 June 2023. This letter was recommended for publication by Associate Editor S. Wang and Editor J. Yi upon evaluation of the reviewers' comments. This work was supported by EU Horizon 2020 Research and Innovation Program under Grant 101017274 (DARKO). *(Corresponding author: Yuchen Liu.)*

Yuchen Liu is with the Corporate Sector Research and Advance Engineering, Robert Bosch GmbH, 71272 Renningen, Germany, and also with the Service Computing Department, Institute of Architecture of Application Systems, University of Stuttgart, 70569 Stuttgart, Germany (e-mail: yuchen.liu2@de.bosch.com, yuchen.liu@iaas.uni-stuttgart.de).

Luigi Palmieri is with the Corporate Sector Research and Advance Engineering, Robert Bosch GmbH, 71272 Renningen, Germany (e-mail: luigi.palmieri@de.bosch.com).

Ilche Georgievski and Marco Aiello are with the Service Computing Department, Institute of Architecture of Application Systems, University of Stuttgart, 70569 Stuttgart, Germany (e-mail: ilche.georgievski@iaas.uni-stuttgart.de; marco.aiello@iaas.uni-stuttgart.de).

Digital Object Identifier 10.1109/LRA.2023.3280816

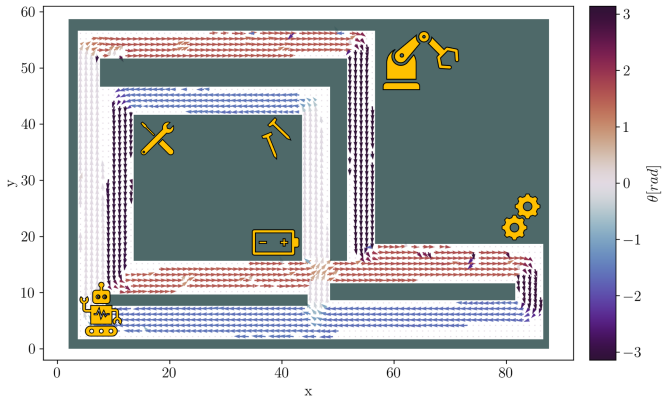


Fig. 1. An example of a factory environment and the human motion pattern. The human motion pattern is modeled using a CLiFF-map [12] represented with the arrows, where the arrow length and color denote the velocity and orientation of the observed human flow at a location respectively.

Numerous recent works focus on learning-based approaches, for instance, using Reinforcement Learning (RL). The model-based approaches, namely automated task planning, use classical symbolic formalisms such as the Planning Domain Description Language (PDDL) [6] to define the planning problems, whereas its development appears highly isolated from the research of learning-based approaches [7]. Many learning-based robot task planning methods have already achieved robust performance in static environments [7], [8], [9] and dynamic environments that consider interaction and collision avoidance with moving objects using RL [10]. However, due to the performance drop of the RL approaches with growing state and action spaces as well as planning horizon, they can only handle relatively simple tasks [11].

Enabling the awareness of dynamic objects in long-term planning still remains a crucial research topic. An efficient way of achieving human awareness is to extract motion patterns of human flow using a Map of Dynamics (MoD), which represents the spatial information of dynamics (e.g., flow density and direction) at different positions, such as Circular-Linear Flow Field map (CLiFF-map) [12]. Predicting human motions individually in crowded environments and subsequently considering them in the planning phase significantly increases the computational effort [10]. On the contrary, MoDs consider the crowd as a whole entity and model its own long-term behaviors. Taking MoDs as prior knowledge of the environment's dynamics enables a simple and efficient derivation of the cost to cross the human flow. The latter can be used to generate robot behaviors that avoid traveling

against the human flow. However, works to combine MoDs in task planning are so far unseen.

Nevertheless, human flow differs from time to time, thus, static MoDs can be easily invalidated. A learning-based approach such as Hierarchical Reinforcement Learning (HRL) can help the task and motion planners to be more adaptable to environmental changes. Aiming to improve the performance of long-term mobile robot task planning in dynamic environments, we propose a Human-Flow-Aware Guided Hierarchical Dyna-Q (HA-GHDQ) algorithm, where the robot uses HRL to learn the costs to cross the human flow from the MoDs and completes long-term tasks. HRL allows the agent to learn through tasks with long planning horizons more efficiently than standard RL, while also being adaptable to environmental changes. Our work is inspired by the Guided Dyna-Q (GDQ) algorithm [7], which only focused on completing short-term navigation tasks using non-hierarchical RL in a static environment. Similar to GDQ, we also use an automated task planner to guide the learning process. In summary, we present the following key contributions:

i) Our approach is the first to combine human flow awareness, hierarchical reinforcement learning, and automated task planning to solve long-term mobile robot task planning problems in dynamic environments.
ii) Our approach outperforms various baselines with respect to sample efficiency, path length, and final plan quality. Our approach achieves similar high-quality plans while being up to 2000 times faster than an optimal search-based planner. Moreover, we show that it is more adaptable and robust against environmental changes than the baselines.

II. Related Work

The presented work lies at the intersection of three fields: MoDs, HRL, and robot task and motion planning.

1) Map of Dynamics: MoDs are powerful in representing typical human movements with flow-like motion patterns and serve as a-priori knowledge in the planning phase [13]. Kucner et al. [12] use Semi-Wrapped Gaussian Mixture Models (SWGMMs) to map the human velocities, and represent the human motion patterns with the Circular-Linear Flow Field map (CLiFF-map). In our previous work [1], we use them to improve conventional robot motion planners to plan smoother and shorter trajectories.

2) Hierarchical Reinforcement Learning: HRL can successfully overcome the drawback of performance drop in standard RL approaches in large state and action spaces as well as long planning horizon, and brings higher learning efficiency, where HRL decomposes large problems into a hierarchy of sub-problems, and each sub-problem can be implemented with a standard RL approach [11]. In recent years, many HRL approaches have been proposed such as Hierarchies of Abstract Machines (HAMs) [14], the option framework [15] and Hierarchical Dyna (H-Dyna) [16]. In this letter, we combine H-Dyna into the option framework to learn additionally from the "replayed" experiences [17].

3) Learning-Based Robot TAMP: Many approaches use RL for solving robot TAMP problems. For instance, Jiang et al. [9] propose a framework that integrates mobile robot TAMP and RL to combine their ability of high-quality offline planning and generalizability, but was applied in a static environment. The GDQ algorithm proposed in the base work [7] uses an automated task planner to guide the RL process for a mobile robot to navigate through a static office environment, but with a short planning horizon. Some recent works also combine automated task planning with HRL [18], [19] to speed up learning for long-term tasks, whereas they are evaluated with relatively simple navigation tasks in static and small-scale environments. For tackling human-robot shared environments, various social navigation approaches are developed to avoid collision with humans in low-level motion planning [1], [10], [20], where the robots should first predict the human motions within a short planning horizon. However, predicting all human motions is computationally very costly in crowded environments [10]. Recent learning-based social navigation methods [21], [22] achieve solid performance in goal-reaching tasks in crowded scenarios, but do not tackle long-term task planning problems.

III. Preliminaries and Notations

1) Task Planning Problems: A task planning problem is a tuple $\langle \boldsymbol{S}, \boldsymbol{A}, T, R, s_0, \boldsymbol{s}_G \rangle$, where $\boldsymbol{S}$ is a finite set of state variables, $\boldsymbol{A}$ is a finite set of actions, T is a state transition function and R is the corresponding state transition reward, s_0 is an initial state and $\boldsymbol{s}_G$ is a set of goal states. The solution of a task planning problem is a plan π, i.e., a list of state-action pairs (s, a) whose execution leads from s_0 to s_G. A plan π^* is optimal when it results in the highest reward.

2) Circular-Linear Flow Field Map: A CLiFF-map uses multimodal statistics to represent motion patterns. For modeling human flow in an environment, the velocities of humans $\boldsymbol{V} = (\theta, \rho)^T$ with speed $\rho \in \mathbb{R}^+$ and orientation $\theta \in [0, 2\pi)$ observed at a location are mapped into a SWGMM $\boldsymbol{\xi}$. A CLiFF-map is defined as a set of SWGMMs along with their motion probability $\bar{p}_n$, observation ratio $\bar{q}_n$ and location l_n: $\boldsymbol{\Xi} = \{(\boldsymbol{\xi_n}, \bar{p}_n, \bar{q}_n, l_n) | n \in \mathbb{Z}^+, l_n \in \mathbb{R}^2\}$. An example of a CLiFF-map is shown in Fig. 1.

3) Hierarchical Reinforcement Learning: HRL approaches decompose large RL problems into a hierarchy of sub-problems, and jointly use the policies in each hierarchical level to interact with the environment. For example, in a two-level option framework, the policy at a higher level π_Ω can be denoted as *policy over option*, and the policy at a lower level π_ω is called *intra-option policy*, namely an action sequence with respect to a given *option* [15]. An *option* $\omega \in \boldsymbol{\Omega}$ is a triple $\langle \boldsymbol{I}_\omega, \pi_\omega, \beta_\omega \rangle$ in which $\boldsymbol{I}_\omega \subseteq \boldsymbol{S}$ is an initiation set of states, $\pi_\omega : \boldsymbol{S} \times \boldsymbol{A} \to [0, 1]$ is an intra-option policy, and $\beta_\omega : \boldsymbol{S} \to [0, 1]$ is a termination condition, with $\boldsymbol{\Omega}$ representing the option space. An option ω is available in state s_t if and only if $s_t \in \boldsymbol{I}_\omega$, then the agent picks an option ω according to its policy over options π_Ω, then follows the intra-option policy π_ω until the termination condition β_ω is met. The option value function $Q_\Omega(s, \omega) : \boldsymbol{S} \times \boldsymbol{\Omega} \times \boldsymbol{A} \to \mathbb{R}$ is the action value function of executing an action with respect to the given state and option. The optimal policy over option π^*_Ω can then be found by maximizing the option value function Q_Ω.

IV. Approach

Hereinafter we focus on solving a long-term mobile robot task planning problem, where the agent should learn to produce a high-level task plan to achieve the given goal while minimizing the overall cost, i.e., traveling distance and the cost to cross human flow derived from the human motion patterns in a given environment.

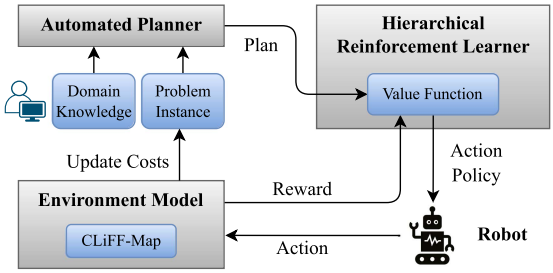


Fig. 2. An overview of the architecture of HA-GHDQ.

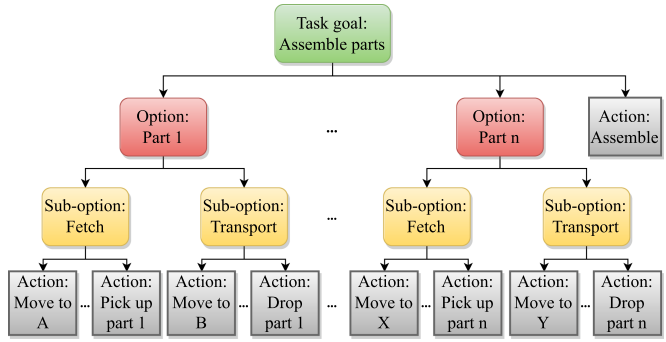


Fig. 3. Task decomposition of HRL.

Running example: We consider a mobile robot that moves in a factory environment and performs an e-bike assembly task. There are five parts of the e-bike (steering wheel, rear wheel, body frame, battery, and saddle) distributed in the factory at corresponding locations. The robot should gather the parts and bring them back to the assembly zone one by one, and assemble them. The robot can perform four actions: *move*, *pick*, *drop*, and *assemble*. The human motion pattern is initially unknown to the robot and can be learned by observing the environment during the exploration. We use CLiFF-map to represent the human motion pattern.

A. Architecture

The learning system of HA-GHDQ consists of three main parts, i.e., the hierarchical reinforcement learner, the environment model, and the automated task planner. An overview of the system architecture is shown in Fig. 2. The planner takes the domain knowledge and the problem instance as inputs (e.g., reported in PDDL). The domain knowledge includes the object types, the predicates, and the action knowledge, while the problem instance contains the initial and goal states, the properties of all objects, and the cost of traversing between each connected location (initialized with distances). Both inputs are defined by the user. We use Π to represent the automated task planner. During the learning process, it computes an optimal task plan π with respect to the current knowledge by minimizing the cost.

The task plan π is passed to the hierarchical reinforcement learner. π provides a sequence of state-action transitions that successfully leads to the goal and guides the update of the action value function. The hierarchical reinforcement learner generates the action policies for the robot to interact with the environment model. The CLiFF-map is integrated into the environment model and serves for obtaining the cost of traversing through human flow. The environment model returns the corresponding rewards back to the hierarchical reinforcement learner for updating the value function, and also to the problem instance to update the cost terms.

B. Task Decomposition

To serve the HRL, the planning problem can be decomposed as follows: to assemble an e-bike, all parts need to be fetched in the factory and transported to the assembly zone. Both fetching and transporting a part can be seen as a simple navigation task consisting of a few *move* actions followed by a *pick* or *drop* action. Thus, the planning problem can be decomposed into three hierarchical levels as shown in Fig. 3. The first level refers to the options, i.e., the parts. The second level denotes the sub-options, i.e., fetch a part or transport it back, with a determined order. The last level consists of sequences of primitive actions for accomplishing the sub-options, which indicate the minimalistic tasks and can be solved by a simple RL approach. Finally, the *assemble* action concludes the task. In addition to $\boldsymbol{\Omega}$, we use $\boldsymbol{U}$ to denote the sub-option space. The policies over option, sub-option, and intra-option policy, i.e., π_Ω, π_U and π_ω, can be interpreted as the order of options, sub-options, and actions respectively.

C. Environment Model

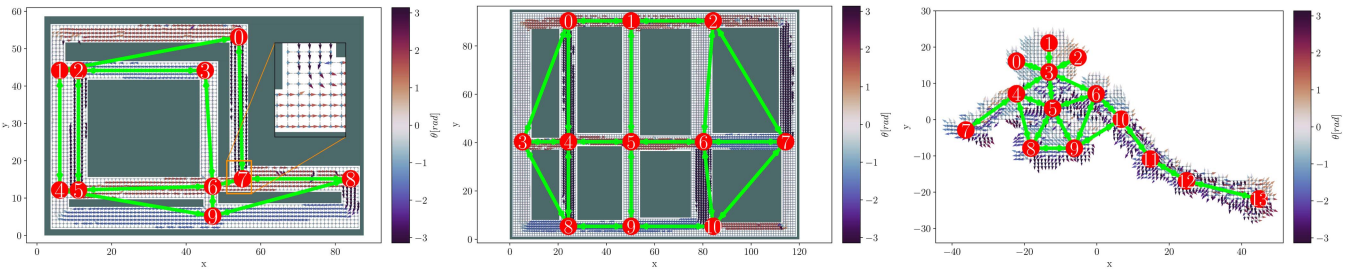
To integrate the CLiFF-map into the environment model, first, the human motion pattern needs to be extracted from a dataset of human trajectories. For simplifying the computation of cost to cross human flow, in each grid cell, we only use the weighted sum of the mixture models to represent the velocity ρ and heading angle θ of human flow, the motion probability $\bar{p}$ and observation ratio $\bar{q}$ are not multi-modal and can therefore be directly used.

For computing the cost of traversing the human flow, a grid graph is built based on the CLiFF-map, as shown in Fig. 4. Each node in the grid graph lies in the center of the corresponding grid cell (1 m resolution) obtained when building the CLiFF-map. The edges between each two grid nodes are bi-directional, given that the cost to cross human flow may be different in each direction. The weight of each edge is computed as the weighted sum of the step cost and the cost to cross human flow between the two grid cells. Moreover, the cost to cross human flow is proportional to the relative angle between agent and human flow direction θ, human velocity ρ, motion probability $\bar{p}$, and observation ratio $\bar{q}$. It has lower values if the robot moves in the same direction as the human flow, and higher values when it moves against the human flow direction. It is computed as follows:

$$\begin{aligned} w_{i,j} &= w_1 c_{step_{i,j}} + w_2 c_{flow_{i,j}} \\ &= w_1 c_{step_{i,j}} + w_2[(1-\cos(\theta_j - \theta_{agent})) \cdot \rho_j \cdot \bar{p}_j \cdot \bar{q}_j] \end{aligned} \tag{1}$$

where $w_{i,j}$, $c_{step_{i,j}}$ and $c_{flow_{i,j}}$ are the edge weight, the step cost, and the cost to cross human flow from grid node i to j, respectively, with $w_{i,j} \geq 0$.

The grid graph is stored in the environment model for obtaining the real cost when the agent interacts with the environment. If an agent moves from one grid node to another, the resulting cost will be the summed weight along all the traveled edges. The



(a) P-shape map [1] (b) Maze map [1] (c) ATC map [23]

Fig. 4. The grid graphs and the planning graphs on the simulation ((a) and (b)) and real-world human trajectory datasets (c). A zoomed view of the grid graph and the CLiFF-map is illustrated on the top right in (a). The planning graphs consist of the red nodes and the green bi-directional edges, with nodes 0 (a), 10 (b) and 9 (c) being the assembly zones, nodes 3 (a), 6 (b) and 2 the steering wheel zones, nodes 5 (a), 8 (b) and 7 (c) the saddle zones, nodes 6 (a), 10 (b) and 0 (c) the rear wheel zones, nodes 7 (a), 1 (b) and 13 (c) the battery zones, and nodes 8 (a), 3 (b) and 1 (c) the body frame zones.

cost from a source grid node *src* to a target grid node *tar* is given by $c_{src,tar} = \sum_{k=src,l=src+1}^{tar-1,tar} w_{k,l}$.

However, producing a long-term task plan based on the dense and large dimensional grid graph is computationally inefficient. Therefore, a sparse planning graph is built based on the grid graph by extracting the relevant landmarks from the map. The landmarks are manually identified (i.e., or can be learned offline), where their locations are relevant for the planning domain (e.g., a part is located at this place, or where the human flow merges or splits). The nearest grid node to a landmark is selected, and treated as a node to constitute the planning graph. The illustration of the planning graph can also be seen in Fig. 4. The robot can only move from one node to a neighbor node when it takes a *move* action. The planning graph consists of significantly fewer nodes and can be used to compute the task plan for the robot. The edges of the planning graph are also bi-directional, and the edge weight in each direction is obtained by first finding the corresponding grid nodes from the grid graph, and then calculating the cost from one grid node to another: $w_{p_{m,n}} = c_{\mathscr{G}(m),\mathscr{G}(n)}$, where $w_{p_{m,n}}$ is the edge weight from node m to node n in the planning graph, c refers to the previously defined cost computation, $\mathscr{G}$ is a mapping function which finds the corresponding grid node from the grid graph given a node from the planning graph.

D. Algorithms

Algorithm 1 describes the learning loop of HA-GHDQ. It takes the inputs of option space $\boldsymbol{\Omega}$, sub-option space $\boldsymbol{U}$, state space $\boldsymbol{S}$, action space $\boldsymbol{A}$, automated task planner Π, termination condition β, initial and goal states s_0 and s_G as well as the parameters: discount factor γ, learning rate α, counter threshold m and maximal reward $R_{\max}$. The action value function Q is optimistically initialized with the help of the automated task planner Π (Line 1), and then updated with policy iteration (Line 7). The learning loop is repeated until the reward converges, or the maximal number of training episodes is reached. The results of the algorithm are the trained Q function, and the policies over option, sub-option, and intra-option policy, namely π_Ω, π_U and π_ω.

Algorithm 2 presents the optimistic initialization. At Line 4, the task planner creates an initial optimistic plan π based on the current knowledge, i.e., the distances between each node in the sparse graph, without knowing the human motion pattern. Since the plan π only consists of state-action pairs, while the HRL needs the hierarchy of options, sub-options, and actions to update the value function, it is necessary to extract the policy over option π_Ω, policy over sub-option π_U as well as the intra-option policy π_ω from the plan π to serve the HRL, i.e., the order of transporting parts and the actions to accomplish the *fetch* and *transport* sub-options (Line 5). The agent uses the policies to initialize the transition, reward, and Q function (Lines 6 to 11).

Algorithm 1: HA-GHDQ.

Data: $\boldsymbol{\Omega}$, $\boldsymbol{U}$, $\boldsymbol{S}$, $\boldsymbol{A}$, Π, β, s_0, s_G, γ, α, m, R_{max}
Result: π_Ω, π_U, π_ω, Q
1 $Q \leftarrow$ InitWithGuide($\boldsymbol{\Omega}, \boldsymbol{U}, \boldsymbol{S}, \boldsymbol{A}, \Pi, s_0, s_G, \gamma, \alpha, R_{max}$)
2 $s \leftarrow s_0, R \leftarrow 0$
3 **for** *e in range(episodes)* **do**
4 $\quad \omega \leftarrow$ random-policy($\boldsymbol{\Omega}$)
5 $\quad u \leftarrow$ random-policy($\boldsymbol{U}(\omega)$)
6 $\quad$ **while** $s \neq s_G$ **do**
7 $\quad\quad Q \leftarrow$ PolicyUpdate($\boldsymbol{\Omega}, \boldsymbol{U}, \boldsymbol{S}, \boldsymbol{A}, \Pi, \beta, Q, s_0, s_G, \gamma, \alpha, m$)
8 $\quad$ **end**
9 **end**
10 $\forall \omega \in \boldsymbol{\Omega}, \forall u \in \boldsymbol{U}(\omega), \forall s \in \boldsymbol{S}, \forall a \in \boldsymbol{A}(\omega, u, s)$, $\pi_\Omega, \pi_U, \pi_\omega \leftarrow \text{argmax}_a Q(\omega, u, s, a)$

Algorithm 3 describes the policy update with hierarchical Dyna-Q. In each training step, the agent selects an action and executes it to interact with the environment, and obtains the reward r and the next state s' from the environment model. The reward function is defined as follows: for a *move* action from node m to node n, the agent receives $r = r_{move} = -w_{p_{m,n}}$ according to the edge weight; for the *pick* or *drop* action, the agent gets $r = r_{pick} = r_{drop} = \frac{1}{2} R_{\max}$; finally, for the *assemble* action, the agent obtains $r = r_{assemble} = R_{\max}$.

The reward r is used to update the Q function (Line 4) and the cost terms in the problem instance if the agent observes that the real cost differs from the cost term in the problem instance (Line 12). Once a state tuple $\langle \omega, u, s, a, s' \rangle$ has been visited for over m times, the transition and reward functions will be updated, indicating this state tuple is already known (Lines 8 to 10). Subsequently, the task planner creates a new plan based on the updated knowledge and updates the Q function with simulated experience (Lines 13 to 21). Once the current option or sub-option terminates, the next possible ones will be selected according to the option value function Q_Ω and the sub-option value function Q_U (Lines 22 to 27).

Algorithm 2: INITWITHGUIDE.

Data: $\boldsymbol{\Omega}$, $\boldsymbol{U}$, $\boldsymbol{S}$, $\boldsymbol{A}$, Π, s_0, s_G, γ, α, R_{max}
Result: Q
for $\forall\omega \in \boldsymbol{\Omega}, \forall u \in \boldsymbol{U}(\omega), \forall s \in \boldsymbol{S}, \forall a \in \boldsymbol{A}(\omega,u,s)$ **do**
  $Q(\omega,u,s,a) \leftarrow 0, R(\omega,u,s,a) \leftarrow 0, T(s'|\omega,u,s,a) \leftarrow 1$
end
$\pi \leftarrow \Pi(\boldsymbol{S},\boldsymbol{A},s_0,s_G)$
$\pi_\Omega, \pi_U, \pi_\omega \leftarrow$ extract_option$(\boldsymbol{\Omega},\boldsymbol{U},\pi)$
for $\forall\omega \in \pi_\Omega, \forall u \in \pi_U, \forall\langle s,a\rangle \in \pi_\omega$ **do**
  $R(\omega,u,s,a) \leftarrow R_{max}$
end
for $\forall\omega \in \boldsymbol{\Omega}, \forall u \in \boldsymbol{U}(\omega), \forall s \in \boldsymbol{S}, \forall a \in \boldsymbol{A}(\omega,u,s)$ **do**
  $Q(\omega,u,s,a) \leftarrow R(\omega,u,s,a) + \gamma \sum_{s' \in S} T(s'|\omega,u,s,a) Q(\omega,u,s',a')$
end

Algorithm 3: POLICYUPDATE.

Data: $\boldsymbol{\Omega}$, $\boldsymbol{U}$, $\boldsymbol{S}$, $\boldsymbol{A}$, Π, β, Q, s_0, s_G, γ, α, m
Result: Q
$\omega, u, s \leftarrow$ current option, sub-option and state
$a \leftarrow \varepsilon$-greedy-policy$_{\mathrm{A}}(Q(\omega,u,s))$
$s', r \leftarrow$ execute(a)
$Q(\omega,u,s,a) \leftarrow Q(\omega,u,s,a) + \alpha[r + \gamma \max_{a'} Q(\omega,u,s',a') - Q(\omega,u,s,a)]$
$C(\omega,u,s,a,s') \leftarrow C(\omega,u,s,a,s') + 1$
$R(\omega,u,s,a) \leftarrow R(\omega,u,s,a) + r$
if $\sum_{s'} C(\omega,u,s,a,s') > m$ **then**
  $T(s'|\omega,u,s,a) \leftarrow C(\omega,u,s,a,s') / \sum_{s''} C(\omega,u,s,a,s'')$
  $R(\omega,u,s,a) \leftarrow R(\omega,u,s,a) / \sum_{s''} C(\omega,u,s,a,s'')$
end
if *r differs from the cost terms in the problem instance* **then**
  Update problem instance with r
  $\pi' \leftarrow \Pi(\boldsymbol{S},\boldsymbol{A},s_0,s_G)$
  $\pi'_\Omega, \pi'_U, \pi'_\omega \leftarrow$ extract_option$(\boldsymbol{\Omega},\boldsymbol{U},\pi')$
  Add $\pi'_\Omega, \pi'_U, \pi'_\omega$ to plan history
end
for *n in range(look_ahead)* **do**
  $\omega, u, s, a \leftarrow$ random experience in plan history
  $s', r \leftarrow$ execute(a)
  Update $Q(\omega,u,s,a)$ with Bellman equation
end
if $\beta(u,s')$ *is True* **then**
  $u \leftarrow$ sequential-policy$_{\mathrm{U}}(Q_U(\omega))$
end
if $\beta(\omega,s')$ *is True* **then**
  $\omega \leftarrow$ random-policy$_\Omega(Q_\Omega))$
end
$s \leftarrow s'$

V. EVALUATION

We compare our approach with three baselines, two are non-HRL approaches, namely Dyna-Q [17] and Guided Dyna-Q (GDQ) [7], and one HRL approach, namely H-Dyna [16]. For evaluating their performance, the part of the hierarchical reinforcement learner shown in Fig. 2 is replaced by each baseline approach correspondingly. The main difference between GDQ and Dyna-Q as well as between HA-GHDQ and H-Dyna is that GDQ and HA-GHDQ use the automated task planner to guide the learning process of Dyna-Q and H-Dyna, respectively. As illustrated in Fig. 4, each time, the robot will start at the assembly zone, and pick up a part by navigating to the corresponding position and transporting it back to the assembly zone successively. In the following, each learning approach is called an "agent" for convenience.

In order to gain insights into the sample efficiency, quality of the results, and robustness against changes in the environment and planning configuration, we test the approaches in three experiments: a static environment, a switching goal situation, and a partial change of the human flow. We compute the following metrics: the cumulative reward, the plan length, the task completion rate (i.e., how many sub-tasks were completed), and the planning time. The task completion rate is computed as follows: completing a sub-option gives 9% of completion, since there are 5 parts (i.e., options) in total and each option contains two sub-options, completing all the options (i.e., all parts are transported to the assembly zone) returns $(9+9)\times 5 = 90\%$ of completion. Finally, performing the *assemble* action gives the last 10%.

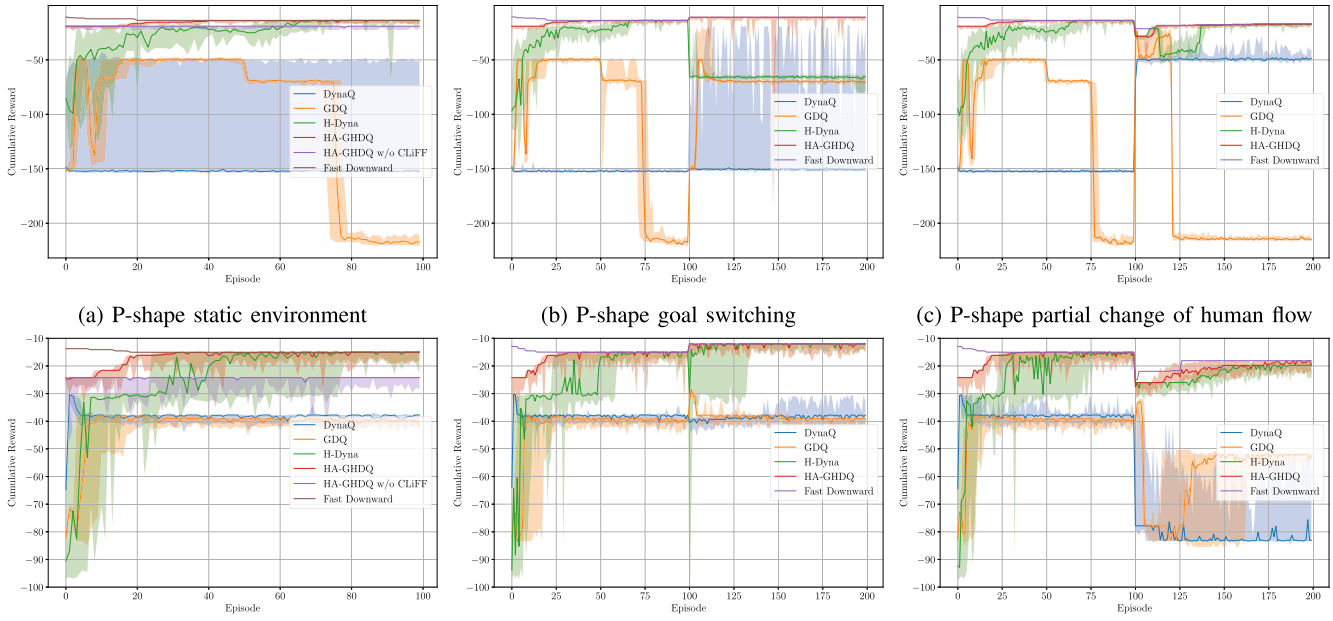
A. Implementation and Parameters

We use the simulation datasets of human trajectory designed in our previous work [1], i.e., the p-shape and the maze map, and the ATC dataset which records the real-world human trajectories in a shopping center [23], as shown in Fig. 4. The parameters are set as follows: the step cost $c_{step} = 1$, the maximal reward $r_{\max} = 4$, the learning rate $\alpha = 0.001$, the discount factor $\gamma = 0.99$, and the probability of random action $\epsilon = 0.015$. Each agent is trained with 20 runs, with 100 episodes per run for the p-shape and maze maps, and 200 episodes for the ATC map. Each episode allows a maximum of 180 steps for the p-shape map and 80 steps for the maze and ATC maps. The number of "replay" steps is set to 300. We focus on highlighting the superiority of HRL in long-term task planning rather than the implementation of HRL, thus, the HRL in our approach is simplified by omitting the option value function Q_Ω and the sub-option value function Q_U, i.e., the order of options and sub-options. Therefore, we simplify the running example by having the same start and end position for the robot to allow a random option order.

All approaches are implemented in Python 3.8.13. We run the experiments on a standard PC with an Intel Xeon W CPU at 3.40 GHz and 32 GB RAM. Furthermore, we use PDDL to encode the domain knowledge and problem instance and Fast Downward (FD) [24] as the automated task planner and A* blind heuristic as the search configuration. We validate the correctness and feasibility of the plan created by the FD planner with the plan validation tool VAL [25].

B. Static Environment

Fig. 5 shows a comparison of the cumulative rewards of each agent and the normalized cost of the optimal plan created by the FD planner, i.e., the optimal cost, in two simulation datasets. The slight drop in the planner cost in the first 20 episodes is caused by the update of the cost terms in the problem instance when the agent observes the cost to cross human flow. Both hierarchical agents have stably reached near-optimal cost, which implies a high consistency in finding the optimal solution. HA-GHDQ takes averagely 54.24 s in the p-shape map and 129.58 s in the maze map till convergence, and H-Dyna takes 18.92 s and 9.84 s in the two maps respectively. Despite HA-GHDQ taking much longer time to converge than H-Dyna due to the execution time of the FD planner, HA-GHDQ converges to near-optimal reward with significantly fewer episodes than H-Dyna in both maps, i.e.,



(a) P-shape static environment (b) P-shape goal switching (c) P-shape partial change of human flow

(d) Maze static environment (e) Maze goal switching (f) Maze partial change of human flow

Fig. 5. Comparison of cumulative rewards and normalized costs of the Fast Downward (FD) planner of all experiments on p-shape and maze maps. The shaded areas represent the 10% and 90% quantile of reward through all runs.

TABLE I
COMPARISON OF TASK COMPLETION RATE AND PLAN LENGTH (I.E., NUMBER OF STEPS IN EACH EPISODE) OF ALL EXPERIMENTS. THE NUMBERS ARE DISPLAYED IN $median \pm std$ AND ARE CALCULATED OVER ALL RUNS AND EPISODES

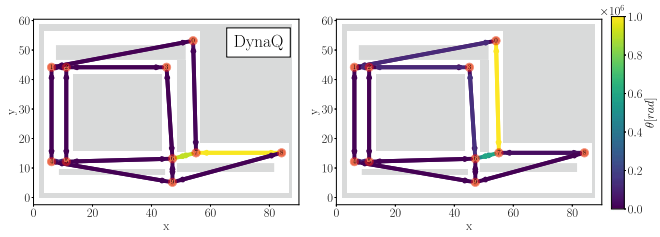
Agents / Planner	Metrics	Experiments						
		Static environment			Goal switching (after)		Partial change of human flow (after)	
		P-shape	Maze	ATC	P-shape	Maze	P-shape	Maze
Dyna-Q	Completion rate [%]	9±9.81	36±5.63	9±0.75	9±1.06	36±10.38	9±10.66	18±8.60
	Plan length	180±0	80±0	80±0	180±0	80±0	180±0	80±0
GDQ	Completion rate [%]	27±16.53	18±10.99	0±3.50	18±7.18	18±5.55	0±8.28	0±5.46
	Plan length	180±1.68	80±0	80±0	180±2.04	80±0	180±0	80±0
H-Dyna	Completion rate [%]	45±116.88	100±37.75	18±37.25	54±34.88	100±2.55	100±6.03	100±0.22
	Plan length	180±63.22	47±17.30	80±14.08	180±24.30	38±12.02	39±58.15	43±3.89
HA-GHDQ	Completion rate [%]	**100±0**	**100±0**	**100±0**	**100±10.20**	**100±0**	**100±0**	**100±0**
	Plan length	**37±5.33**	**44±4.45**	**43±2.53**	**39±22.84**	**38±2.09**	**37±9.70**	**43±2.50**
Fast Downward	Plan length	**37**	**44**	**43**	**39**	**38**	**37**	**43**

40 vs. 70 episodes on average in the p-shape map (75% more than HA-GHDQ) and 32 vs. 71 episodes on average in the maze map (122% more than HA-GHDQ). Table I also presents that H-Dyna is only able to complete the task goal in the maze map, while HA-GHDQ consistently reaches 100% of completion, and is able to achieve the optimal plan length with a notably lower standard deviation.

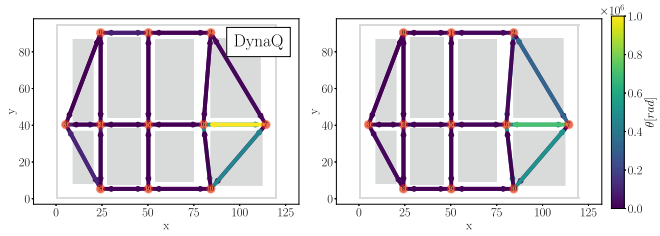
Additionally we test the HA-GHDQ agent without learning the CLiFF-map derived costs, which result in lower rewards than the optimal values. Those are close to HA-GHDQ's initial rewards as shown in Fig. 5(a) and (d), since the costs of traversing between each node are only initialized with distances (see Section IV-A).

In contrast, the non-hierarchical agents GDQ and Dyna-Q converge to notably lower rewards, and none of them completes the task goal. The reason for the poor performance can be inferred from Fig. 6, the brighter edge color implies a significantly higher frequency of traversing, which is caused by a convergence in a local minimum. For the ATC dataset we run the agents in the static environment experiment: since the human trajectories are recorded in the real-world, simply changing a part of the trajectories will lead to non-existing human behaviors. The rewards shown in Fig. 7 display similar results, where only the two hierarchical agents reach the optimal cost. Although HA-GHDQ takes averagely 473.14 s to converge and H-Dyna takes 23.09 s, HA-GHDQ uses significantly fewer episodes than H-Dyna, i.e., 21 vs. 171 episodes (719% more than HA-GHDQ), and has a dominantly higher median completion rate as shown in Table I.

Moreover, the trained HA-GHDQ agent can generate a plan up to nearly 70% faster (in the ATC map) than H-Dyna, and over 200 times faster than the FD planner in the p-shape and the maze



(a) P-shape



(b) Maze

Fig. 6. Comparison of exploration heat-map of Dyna-Q and GDQ agents of partial change of human flow experiment. The edge color indicates the total frequency of traversing along the edge in both directions.

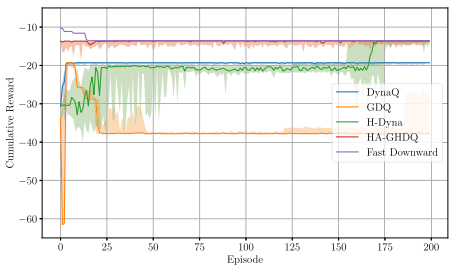


Fig. 7. Reward of static environment experiment on ATC map. The shaded areas represent the 10% and 90% quantile of reward through all runs.

TABLE II
COMPARISON OF PLANNING TIME IN $[ms]$ BETWEEN TRAINED AGENTS AND FD PLANNER OF ALL MAPS. THE RESULTS OF DYNA-Q AND GDQ ARE NEGLECTED SINCE THEY NEVER FULLY COMPLETE THE TASK. THE NUMBERS ARE SHOWN IN $median \pm std$

Agents / Planner	Maps		
	P-shape	Maze	ATC
H-Dyna	18.64±7.90	9.29±12.80	9.16±2.05
HA-GHDQ	**18.56±4.68**	**8.92±2.23**	**5.39±0.62**
Fast Downward	4030.00±49.96	2150±16.05	13110.00±456.74

maps, as well as over 2000 time faster than FD in the ATC map as shown in Table II.

C. Goal Switching

We evaluate the adaptability of the agents against changes in the planning configuration. We switch the location of the steering wheel zone and the assembly zone after completing the static environment experiment, and train for additional 100 episodes without resetting the learned Q functions. The other configurations and the CLiFF-map remain unchanged.

As shown in Fig. 5(b) and (e), for the first 100 episodes, the rewards of all agents developed similar to those in the previous experiments. After the switch, the reward of H-Dyna drops immediately and then slowly converges again to a new highest reward in the maze map, but fails to reach the optimal cost in the p-shape map, and only has 54% of median completion rate, while the reward of HA-GHDQ has barely dropped and converges again to the new optimal cost in both maps. Table I shows that H-Dyna can still complete the task after the configuration change in the maze map, but the lower standard deviations in completion rate and plan length of HA-GHDQ indicate a high capability of completing the task consistently, and high robustness against the configuration change. In contrast, GDQ and Dyna-Q still perform the worst with respect to all the metrics.

D. Partial Change of Human Flow

In order to evaluate the robustness of the agents against the environmental change, we flip the human flow direction of the CLiFF-map in the circle of nodes $2-3-6-5-2$ in the p-shape map and nodes $2-6-7-2$ in the maze map by $180°$ for additional 100 episodes. Other values in the CLiFF-map as well as the planning configuration remain the same.

After the change, the reward of HA-GHDQ drops lightly but reaches the new optimal cost again in both maps, while the reward of H-Dyna in the p-shape map rises slower, and does not reach the optimal cost in the maze map, as reported in Fig. 5(f). According to the task completion rate and plan length given in Table I, HA-GHDQ still performs the best after the change of human flow, indicating high robustness and adaptability to environmental change.

E. Observations

1) On the Need of MoDs: Without embedding the costs derived from the MoDs, the agents only consider the traveling distance, and it leads to lower rewards, i.e., higher overall costs, as shown in Fig. 5(a) and (d). By using the MoDs to represent the human motion patterns, the agents learn to minimize the overall cost by avoiding traversing against the human flow to achieve human flow awareness.

2) On the Need of HRL: Observed from all the experiments above, we conclude that with the growing complexity of the task planning problems in a given environment, it is more likely that the agent will take the same action multiple times, but in fact, each time the action serves a different purpose, i.e., under a different option. Thus, the same action will have different Q values regarding different options. Non-hierarchical RL is by no means proper in this case, and will possibly lead to confusion in selecting actions.

3) On the Combination of Learning and Planning: Although the computation time of the FD planner increases HA-GHDQ's training time, the FD planner helps to optimistically initialize the agent, and improves sample efficiency by notably reducing the number of steps in each episode, which can be implied by comparing the plan length between H-Dyna and HA-GHDQ in Table I. Moreover, the usage of the FD planner allows HA-GHDQ to achieve a higher completion rate and robustness against changes in environment and planning configuration, as well as shorter planning time than H-Dyna. We argue that plan quality, robustness, and planning time are significantly more important for benchmarking robot operation than training time.

While the learning agents can automatically perceive the environmental changes when interacting with the environment and obtaining rewards, automated task planners such as FD are less adaptable to these changes, they need to be combined with extra functionalities, e.g., execution monitoring and recovery. Besides, their domain knowledge and problem instances are usually specified by human experts manually. Moreover, automated task planners tend to fail with the growing complexity of task planning problems and longer planning horizons [26].

4) On the Limitations: Our approach can be applied to other mobile robot navigation tasks, but the computational efficiency of the training phase varies according to the efficiency of the automated task planning, which grows with the complexity of the planning problem. Our approach does not consider replanning in failure cases, where the robot stops at a place that is neither at any node nor edge.

VI. Conclusion

To handle the complexity of human-robot-cohabited environments and long planning horizons, we propose Human-Flow-Aware Guided Hierarchical Dyna-Q (HA-GHDQ), which presents a novel combination of HRL with automated task planning that plans considering learned costs derived from MoDs' representations. Human awareness is enabled by considering the latter, which encodes long-term human behaviors in crowded and dynamic environments. The evaluation shows that our algorithm outperforms all the other baselines with respect to sample efficiency, robustness to the changes in planning configuration and environment, plan quality, and planning time. For future work, we plan to use function approximators for representing the learned hierarchical Q function, and train also with different problem definitions. We also plan to validate our approach on a robot in a real-world environment, and open source the code.

Acknowledgment

The authors thank Tomasz P. Kucner for the support on the usage of the CLiFF-map.

References

[1] L. Palmieri, T. P. Kucner, M. Magnusson, A. J. Lilienthal, and K. O. Arras, "Kinodynamic motion planning on Gaussian mixture fields," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 6176–6181.

[2] R. Triebel et al., "SPENCER: A socially aware service robot for passenger guidance and help in busy airports," in *Field and Service Robotics*. Berlin, Germany: Springer, 2016, pp. 607–622.

[3] I. Georgievski, I. H. J. Jeyakumar, and S. Kale, "Designing a system based on robotic assistance for privacy awareness in smart environments," in *Proc. IEEE 12th Annu. Ubiquitous Comput., Electron. Mobile Commun. Conf.*, 2021, pp. 0427–0432.

[4] E. Kaldeli, E. U. Warriach, A. Lazovik, and M. Aiello, "Coordinating the web of services for a smart home," *ACM Trans. Web*, vol. 7, no. 2, pp. 1–40, 2013.

[5] E. Karpas and D. Magazzeni, "Automated planning for robotics," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 3, pp. 417–439, 2020.

[6] M. Ghallab et al., "PDDL | the planning domain definition language," Tech. Rep. CVC TR-98-003/DCS TR-1165, 1998.

[7] Y. Hayamizu, S. Amiri, K. Chandan, K. Takadama, and S. Zhang, "Guiding robot exploration in reinforcement learning via automated planning," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2021, vol. 31, pp. 625–633.

[8] D. Gordon, D. Fox, and A. Farhadi, "What should I do now? Marrying reinforcement learning and symbolic planning," 2019, *arXiv:1901.01492*.

[9] Y. Jiang, F. Yang, S. Zhang, and P. Stone, "Task-motion planning with reinforcement learning for adaptable mobile service robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 7529–7534.

[10] P. T. Singamaneni, A. Favier, and R. Alami, "Human-aware navigation planner for diverse human-robot interaction contexts," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021, pp. 5817–5824.

[11] S. Pateria, B. Subagdja, A.-H. Tan, and C. Quek, "Hierarchical reinforcement learning: A comprehensive survey," *ACM Comput. Surv.*, vol. 54, no. 5, pp. 1–35, 2021.

[12] T. P. Kucner, A. J. Lilienthal, M. Magnusson, L. Palmieri, and C. S. Swaminathan, *Probabilistic Mapping of Spatial Motion Patterns for Mobile Robots*. Berlin, Germany: Springer, 2020.

[13] F. Surma, T. P. Kucner, and M. Mansouri, "Multiple robots avoid humans to get the jobs done: An approach to human-aware task allocation," in *Proc. IEEE Eur. Conf. Mobile Robots*, 2021, pp. 1–6.

[14] R. Parr and S. Russell, "Reinforcement learning with hierarchies of machines," in *Proc. 10th Int. Conf. Neural Inf. Process. Syst.*, 1997, pp. 1043–1049.

[15] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artif. Intell.*, vol. 112, no. 1/2, pp. 181–211, 1999.

[16] S. P. Singh, "Reinforcement learning with a hierarchy of abstract models," in *Proc. Nat. Conf. Artif. Intell.*, 1992, Art. no. 202.

[17] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM Sigart Bull.*, vol. 2, no. 4, pp. 160–163, 1991.

[18] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith, "Using reward machines for high-level task specification and decomposition in reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2107–2116.

[19] L. Illanes, X. Yan, R. T. Icarte, and S. A. McIlraith, "Symbolic plans as high-level instructions for reinforcement learning," in *Proc. Int. Conf. Automated Plan. Scheduling*, 2020, vol. 30, pp. 540–550.

[20] H. Khambhaita and R. Alami, "Assessing the social criteria for human-robot collaborative navigation: A comparison of human-aware navigation planners," in *Proc. IEEE 26th Int. Symp. Robot Hum. Interactive Commun.*, 2017, pp. 1140–1145.

[21] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6015–6022.

[22] S. Matsuzaki and Y. Hasegawa, "Learning crowd-aware robot navigation from challenging environments via distributed deep reinforcement learning," in *Proc. Int. Conf. Robot. Automat.*, 2022, pp. 4730–4736.

[23] D. Brščić, T. Kanda, T. Ikeda, and T. Miyashita, "Person tracking in large public spaces using 3-D range sensors," *IEEE Trans. Human-Mach. Syst.*, vol. 43, no. 6, pp. 522–534, Nov. 2013.

[24] M. Helmert, "The fast downward planning system," *J. Artif. Intell. Res.*, vol. 26, pp. 191–246, 2006.

[25] R. Howey, D. Long, and M. Fox, "VAL: Automatic plan validation, continuous effects and mixed initiative planning using PDDL," in *Proc. IEEE 16th Int. Conf. Tools With Artif. Intell.*, 2004, pp. 294–301.

[26] C. Agia et al., "Taskography: Evaluating robot task planning over large 3D scene graphs," in *Proc. Conf. Robot Learn.*, 2022, pp. 46–58.