



Article

Hierarchical Task Assignment and Path Finding with Limited Communication for Robot Swarms

Dario Albani ^{1,2,3,*} , Wolfgang Hönig ^{4,5} , Daniele Nardi ³, Nora Ayanian ⁴ and Vito Trianni ² 

¹ Autonomous Robotics Research Centre (ARRC), Technology Innovation Institute, Abu Dhabi, United Arab Emirates

² Institute of Cognitive Sciences and Technologies (ISTC), Italian National Research Council (CNR), 00185 Rome, Italy; vito.trianni@istc.cnr.it

³ Department of Computer, Control and Management Engineering, Sapienza University of Rome, 00185 Rome, Italy; nardi@diag.uniroma1.it

⁴ Department of Computer Science, Viterbi School of Engineering, University of Southern California, Los Angeles, CA 90089, USA; whoenig@usc.edu (W.H.); ayanian@usc.edu (N.A.)

⁵ Bitcraze AB, 21222 Malmö, Sweden

* Correspondence: dario.albani@tii.ae

Abstract: Complex service robotics scenarios entail unpredictable task appearance both in space and time. This requires robots to continuously relocate and imposes a trade-off between motion costs and efficiency in task execution. In such scenarios, multi-robot systems and even swarms of robots can be exploited to service different areas in parallel. An efficient deployment needs to continuously determine the best allocation according to the actual service needs, while also taking relocation costs into account when such allocation must be modified. For large scale problems, centrally predicting optimal allocations and movement paths for each robot quickly becomes infeasible. Instead, decentralized solutions are needed that allow the robotic system to self-organize and adaptively respond to the task demands. In this paper, we propose a distributed and asynchronous approach to simultaneous task assignment and path planning for robot swarms, which combines a bio-inspired collective decision-making process for the allocation of robots to areas to be serviced, and a search-based path planning approach for the actual routing of robots towards tasks to be executed. Task allocation exploits a hierarchical representation of the workspace, supporting the robot deployment to the areas that mostly require service. We investigate four realistic environments of increasing complexity, where each task requires a robot to reach a location and work for a specific amount of time. The proposed approach improves over two different baseline algorithms in specific settings with statistical significance, while showing consistently good results overall. Moreover, the proposed solution is robust to limited communication and robot failures.

Keywords: swarm robotics; decision-making; task allocation; path finding



Citation: Albani, D.; Hönig, W.; Nardi, D.; Ayanian, N.; Trianni, V. Hierarchical Task Assignment and Path Finding with Limited Communication for Robot Swarms. *Appl. Sci.* **2021**, *11*, 3115. <https://doi.org/10.3390/app11073115>

Academic Editor: Álvaro Gutiérrez

Received: 28 February 2021

Accepted: 26 March 2021

Published: 31 March 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Several service scenarios are characterized by new tasks appearing at any time and in any place, necessitating persistent operation [1]. As an example, consider a large warehouse where orders arrive erratically and diverse items need to be collected from different locations. In such context, besides retrieving goods, other services must be continuously executed on pallets, such as tidying, sorting objects, and controlling and refilling stocks [2,3]. Similarly, in a greenhouse, continuous and localized operation is necessary for pruning plants, variable-rate spraying, and selective harvesting. Finally, in a facility or office environment, cleaning or delivery tasks may be continuously and unpredictably requested by users located in different areas [4]. Automation in such context cannot always rely on centralized, fixed infrastructures, which may be costly or impractical. Instead, wireless sensor networks, IoT, and mobile devices may be exploited to raise the need for tasks to be serviced by a swarm of specialized robots [5,6].

Addressing a persistent and unpredictable service demand with a multi-robot system can provide higher efficiency, thanks to parallel operation, and larger flexibility by modulating the operational costs in terms of robots employed, as well as by deploying robots only where needed. At the same time, the robotic system must prove scalable, fault-tolerant and robust, to ensure continuous and efficient operation despite the number of units deployed, the occurrence of failures, and the variability in the task demands. Concrete solutions can be obtained by addressing two concurrent problems. On the one hand, it is necessary to find a suitable task assignment for each robot, so that costs are minimized in terms of relocation, robot idle time, and task waiting time [3,7,8]. On the other hand, a collision-free trajectory for the robot must be planned in order to safely reach the destination without interfering with other robots' plans [3,9]. This class of problems is often split in Multi-Robot Task Allocation (MRTA) and Multi-Agent Path Finding (MAPF). A common naïve approach consists in solving MRTA and MAPF separately. However, in cluttered environments, such as indoor spaces, those two problems are tightly coupled: the former provides a set of origin-destination pairs to the latter, which, in turns, returns back actual costs and timings associated to the movements planned by the different robots. For both MRTA and MAPF, many centralized and decentralized approaches exist in the literature (see Section 2). Centralized solutions often rely on perfect intra-robot communication and suffer the well-known issues of scaling with group size—especially when the task demand can rapidly change over time—and of fault-tolerance due to the existence of a single point of failure. Decentralized approaches, on the other hand, potentially feature high fault-tolerance and scalability, but few have been shown to operate in environments with limited communication.

Here, we propose a decentralized algorithm for both task allocation and motion planning that is robust to limited communication and robot failures, and scalable to different environments and group size. Our method is referred to as Hierarchical Task Assignment and Path Finding (HTAPF) because the task allocation problem is solved exploiting a hierarchical representation of the locations where tasks must be executed. More precisely, by partitioning the work-area in a quad-tree, our MRTA problem is efficiently solved through the deployment of robots in areas of interest corresponding to the tree leafs where tasks to be executed are available. To this end, we propose a non-trivial extension of a collective decision-making process designed for robot swarms [10,11] to deal with a hierarchy of sequential decisions, allowing a robot swarm to prioritize those areas with highest demand, while minimizing movement costs. This task allocation approach is tightly coupled with a search-based path planning approach, which plans the robot movements and estimates the expected costs [12]. Thanks to this coupling, HTAPF is able to dynamically adjust the robots' density in different areas, granting consistently good performance across different problem instances.

The main contributions of this paper are the following: (i) introduction of a decentralized hierarchical approach to the task assignment problem; (ii) tight coupling of the task assignment and path finding processes based on a novel utility function; (iii) validation of the proposed approach in realistic settings; (iv) statistical analysis and comparison of the performances of the system against two different baselines.

In the following, we first discuss the literature about coupled MRTA and MAPF problems (see Section 2). Next, we provide a formal description of the analyzed problem together with the proposed approach in all its constituent parts (see Section 3). Section 4.1 presents the results obtained from the experimental evaluation, and Section 5 concludes by discussing the benefits and limitations of the proposed approach.

2. Related Work

Centralized approaches to coupled MRTA and MAPF have been proposed that, given a fixed road-map, compute the shortest path first and then utilize integer linear programming (ILP) to compute the optimal task assignment [13]. Other approaches use search-based methods that can find optimal and bounded sub-optimal schedules with respect to the

sum-of-cost [14] or optimal schedules with respect to make-span [15]. In obstacle-free spaces, the concurrent assignment and planning of trajectories (CAPT) finds optimal motions plans for multiple robots by minimizing the squared velocity. This method has been also proposed in a decentralized version, but with no optimality guarantees [16]. Other decentralized solutions for the combined problem include controller synthesis to solve the formation change problem in environments with obstacles and communication constraints [17]. However, this solution scales exponentially with the number of robots. In Reference [18], a decentralized control approach is presented featuring good performance for large robot teams, but—unlike our approach—it does not consider obstacles. Additionally, this approach considers pairwise task-swaps and, therefore, might not optimize the objective over the whole team. Other approaches are based on the estimation of the robot density [19]. The swarm distribution is estimated locally, followed by the distributed solution of an optimal transport problem. However, also this method does not consider obstacles posing constraints on motion and communication.

Some traditional approaches to decentralized task assignment use auctions [20]. A great part of these methods rely on temporarily centralizing the decision-making on a auctioneer that collects offers and defines the winning robot, to which the task gets assigned. In swarm robotics contexts, similar techniques are often unpractical due to the high communication load for large groups, and the limited communication abilities of the individual robots. Instead, in swarm robotics, threshold-based approaches [21] or stochastic task switching [22] are often preferred. These methods are inspired by biological systems where a flexible and stochastic allocation of tasks is observed (e.g., in social insects, individuals engage in tasks on the basis of the urgency of accomplishing it [23,24]). Robot swarms employ similar strategies, which allow to achieve coordination in an implicit way, relying on probabilistic rules that can be modulated by the individual ability to evaluate the utility in engaging in a given task. These methods can be improved by enabling interaction among individuals deciding collectively which tasks to execute or in which area to move, and various task assignment dynamics have been observed from winner take all to a balanced deployment [11,25]. These latter studies are inspired by the decision-making abilities observed in ants [26,27] and honeybees [28]. The latter, when searching for a site where to establish a new nest, engage in repeated interactions trying to recruit other scouts for high-quality sites, at the same time discouraging other individuals to recruit for alternative sites. Theoretical studies revealed that the mechanisms involved in this collective process can lead to optimal decisions in a relatively short time [28], also presenting striking similarities with the neural dynamics observed in human decision-making [29,30].

Previous work deployed a decision-making algorithm based on the behavior of honeybees and tailored to collective decisions in decentralized systems [10], which proved useful for the design of artificial systems in a variety of contexts, from robot swarms [11,25,31,32] to cognitive radio networks [33]. Here, we extend this framework to a hierarchy of sequential decisions, largely increasing the complexity of the resulting collective dynamics. In contrast to previous studies, we account for robot motion and collision costs, robot failures, and limited communication. For path planning, we use a fully distributed version of the priority-based planning paradigm [12] as described in Section 3.2.3. Priorities are assigned implicitly by the order in which data is received from nearby robots. No token passing procedure or specific schedule is in place ensuring robust execution also with limited probabilistic communication and robot failures. Other decentralized path planning methods—not considered here—include push-rotate-update [34] and a method where the path is planned by a sensor network [35].

3. Hierarchical Task Assignment and Path Finding

We consider scenarios with several tasks to be serviced, e.g., an area needs cleaning or human workers request supplies. We assume that there is no central authority coordinating the task execution or sharing the list of available tasks to all the robots. Additionally, the task distribution is not known in advance, and tasks may appear at any time in any place. Here,

we first provide a formal description of the problem, and then discuss our approach to address both MRTA and MAPF in a fully decentralized way.

3.1. Problem Description

We consider the class of problems characterized by tasks that can be executed by a single robot at a time (SR), and robots that can execute only a single task at the time (ST). We also consider instantaneous assignment (IA), as the arrival of further tasks is not available and unpredictable [36,37].

Without loss of generality, the work area is defined as a square partitioned in a 4-connected grid of size $N \times N$, $N \in \mathbb{N}$ (see Figure 1, left), which can be configured to represent spaces of different complexity, from an office to a warehouse. A grid cell $c_i \in \mathcal{C}$ is identified by its coordinates $x_i, y_i \in \{1, \dots, N\}$, and might represent free space where robots can move and tasks might appear, or an obstacle (e.g., a wall). Each free cell can be occupied by at most one robot and can contain at most one task at any time.

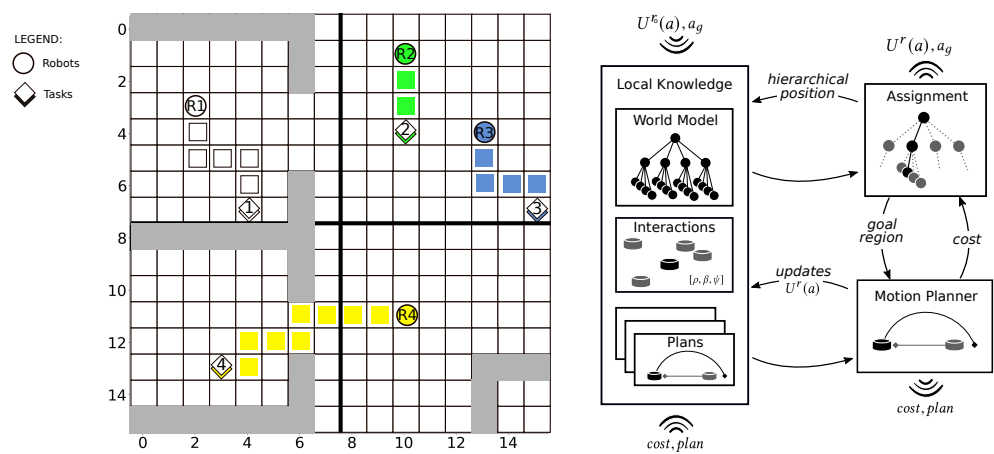


Figure 1. Left: example outcome of Hierarchical Task Assignment and Path Finding (HTAPF) with four robots and four tasks in a 16×16 grid world, here divided in 4 sub-areas. Robots move to the closest cell with a Manhattan neighborhood, and share their plan with other robots within the communication range. Conflicts are resolved by sequential planning, exploiting the knowledge of others' plans for task assignment and collision-free path planning. In the image, each robot is identified by its unique color and number, i.e., r_1 white, r_2 green, r_3 blue, r_4 yellow. Grey regions are walls. Right: Interaction scheme among the different components of HTAPF. The local knowledge of robot r stores the world model, the estimated area utilities obtained upon interaction with other robots, and the received plans from other robots. Both the area assignment process and the motion planner exploit and update the local knowledge and interact with each other with local information passing. Information received from other robots about area utilities (i.e., $U^0(a)$ from robot r_0 about area a and the goal region a_g) and individual motion plans are exploited to update the local knowledge.

A task $T_i \in \mathcal{T}$ is a tuple $T_i = (x_i, y_i, t_i^s, t_i^e, t_i^w)$, which requires any robot to move to the cell at location (x_i, y_i) and work at that location. Task T_i appears at time-step t_i^s and is accomplished by a robot at time-step t_i^e (where $t_i^e = \infty$ as long as the task has not been finished). The work time t_i^w is the time required from a robot to stay at the task location and actually execute the task (e.g., cleaning, delivery, or maintenance). The task set \mathcal{T} is the disjoint union of the completed tasks $\mathcal{T}^c = \{T_i | T_i \in \mathcal{T}, t_i^e \neq \infty\}$ and the unfinished tasks $\mathcal{T}^u = \{T_i | T_i \in \mathcal{T}, t_i^e = \infty\}$. At any time t , we can define the set of available tasks $\mathcal{T}^a(t) = \{T_i | T_i \in \mathcal{T}^u, t_i^s < t\}$. In this study, the primary objective is to serve the highest number of tasks ($\max |\mathcal{T}^c|$).

We consider R identical holonomic robots starting in randomly assigned free cells. A robot $r_i \in \mathcal{R}$ is described by tuple $r_i = (i, x_i, y_i, s_i, a_i)$ identified by its unique id i , an associated position in the environment given by its coordinates (x_i, y_i) a discrete state s_i indicating the current behavior (whether exploring or exploiting available knowledge,

see below) and an area a_i where the robot is currently deployed. At each time-step, a robot can wait at its current position, move to an adjacent free cell, or work on a task located in the cell it is occupying. We assume limited sensing capabilities: robots can only sense adjacent neighbors, and use their sensing abilities to avoid moving in a cell that is already occupied. Additionally, robots can communicate with each other wirelessly. Communication is simulated by means of a realistic model for WiFi networks [38–40], whereby the received power (in dBm) from robot r_j when robot r_i is transmitting can be computed as:

$$P_r(i, j) = P_0 - 10\eta \log(d(i, j)) - nW(i, j) \cdot W + \mathcal{N}(0, \sigma), \quad (1)$$

where P_0 is the signal strength at 1 m, η is the path loss, and $d(i, j)$ computes the distance in meters between the two robots. Moreover, $nW(i, j)$ computes the number of walls in the direct line between the two robots, W is the wall attenuation, and σ the noise variance. Data is exchanged in packets (or frames), and the frame error rate is defined as:

$$\text{FER}(P_r(i, j)) = \text{FER}_S \cdot e^{\gamma(S - (P_r(i, j) - N_b) - N_{th})}, \quad (2)$$

where S is the receiver sensitivity, FER_S and γ are hardware specific constants, N_b is the background noise, and N_{th} the thermal noise. In this work, we exploit the receiver sensitivity S as a parameter to determine local and realistic communication abilities.

The above communication model is relevant also for discovering tasks. Indeed, we consider tasks as being issued by mobile devices equipped with the same WiFi communication system as the robots (e.g., tablets or smartphones). Hence, robots might have limited perception of the available tasks and the existence of a task is not known by all the robots at the same time. We assume, unless otherwise stated, that robots do not share task-related information.

3.2. HTAPF: Principles and Methods

We now illustrate the details of HTAPF, our novel distributed approach that couples MRTA and MAPF. The overall approach includes three main components: (i) the *local knowledge* available to each robot, (ii) the *area assignment* leveraging interaction with other robots, and (iii) the *motion planning and task assignment*, which is performed by exploiting the knowledge of other robots' plans (see Figure 1, right).

The local knowledge is available to each robot and consists of a world model, area utility estimations obtained via interaction with other robots, and motion plans of nearby robots. The world model has a map of the environment—the only information known a priori—and a hierarchical description of the work area, that is divided in sub-areas (see Section 3.2.1). As time goes by, the world model is updated including information about the available tasks and about the presence of other robots. Robots exchange two types of information. First, they share their locally-estimated perceived utility of working in the area they are currently located, which is exploited to support the area/task assignment process. Second, they exchange their motion plans, which are useful for path finding.

The area assignment process generates goals for the motion planner in terms of a goal area where the robot wants to service tasks. To determine the robot destination, the area assignment exploits the world model, the information coming from other robots, and knowledge from the motion planner about the costs of reaching a target area. In other words, robots can choose to work in an area only if they are assigned to it, and their motion plan is, therefore, influenced by the area to which they belong. For instance, in Figure 1 (left), all robots can move and work within their current area but r_4 , the yellow robot, which has no task available in its area and needs to get assigned to a different one before selecting a task (in this case, T_4).

Task assignment in the target area and motion planning are performed concurrently. The motion planner is fully distributed and does not require synchronization among robots. Given a goal area, it assigns tasks and generates collision free paths according to the information previously received and stored in the local knowledge, including the plans

of other robots. The current plan is also used internally to enhance the area assignment process and is also broadcast to other robots to support concurrent planning. For instance, in Figure 1, the green robot r_2 first plans its path toward task T_2 , and r_3 , the blue robot, later decides to move to task T_3 even if farther away to avoid interference with r_2 .

From this high-level description, it is evident that there is a very tight coupling between the different components, which constantly share information to increase performance both at the level of the individual robot and of the group (also see Algorithm 1). In the following, we provide a detailed description, starting from the world model and then moving to the main contributions of this study, that is, the task assignment and path finding, giving particular attention to the coupling between the two.

Algorithm 1 HTAPF

```

1: procedure STEP( $a_i^n, D, P_a, P_d$ )
2:   for  $\tau$  in range  $1 \dots D$  do
3:      $\gamma_m^{n+1}, \rho_m^{n+1}, \alpha_i^n, \beta_{i,i'}^n, \psi_i^n \leftarrow 0$  ▷ Reset transition probabilities
4:      $r_o \leftarrow \text{SelectNeighbour}()$  ▷ Random choice of a known robot
5:      $s_i \leftarrow \text{NextState}(P_a, P_d)$  ▷ Selection of next transition type
6:     if  $s_i = s_a$  then
7:        $\alpha_i^n, \beta_{i,i'}^n, \psi_i^n \leftarrow \text{AscendingTransitions}(a_i^n, r_o)$  ▷ Equations (6)–(8)
8:     else
9:        $\gamma_i^r, \rho_i^r \leftarrow \text{DescendingTransitions}(a_i^n, r_o)$  ▷ Equations (4) and (5)
10:       $a_g \leftarrow \text{RandomTransition}(\gamma_i^r, \rho_i^r, \alpha_i^n, \beta_{i,i'}^n, \psi_i^n)$  ▷ New hierarchical area
11:       $\text{UpdateTree}(a_g)$  ▷ Update all ancestors
12:       $\text{cost}, \text{plan} \leftarrow \text{PlanMotion}(a_i^n, a_g)$  ▷ Collision Free Path
13:       $\text{Broadcast}(a_g, U^r(a_g), \text{cost}, \text{plan})$ 
14:       $\text{UpdateUtility}(a_g, \text{cost})$  ▷  $U^r(a_g)$  in Equation (3)
15:       $\text{ExecuteNextAction}(\text{plan})$ 

```

3.2.1. World Model

As previously stated, we employ a world model that consists of a hierarchical description of the discretized space, which is represented by a generic k -d tree. Such a hierarchical description is general enough to be automatically constructed, without requiring knowledge about the structure of the work area in terms of rooms, corridors, and connecting spaces. In the following, we consider $k = 4$, that is, a quad-tree where each leaf node relates to a small area where task may require execution, and the root node corresponds to a region representing the whole $N \times N$ environment. Leaf and non-root nodes are assigned following the classical quad-tree generation as illustrated in Figure 2 (left and center panels). Every node in the tree is associated with an area a_j^n characterized by the level in the hierarchy n and an index $j \in \{1, 2, 3, 4\}$, so that the area is fully described by the tuple $a_j^n = (x_j^n, y_j^n, l^n)$, where (x_j^n, y_j^n) are the coordinates of the top-left corner of the considered region and $l^n = N/2^n$ is its length (i.e., nodes closer to the root have greater length). The root node a^0 is identified by the tuple $(0, 0, N)$, while a single cell c_j can be identified as $a_j^m = (x_j^m, y_j^m, 1)$, where $m = \log_2 N$.

At the beginning, a robot r is provided with the size of the environment N and the depth of the desired quadtree $M \leq \log_2 N$, which is sufficient to build its own world model. At any time t , each area a_j^n of any size can contain a set of tasks $\mathcal{T}_{a_j^n}^a(t)$ that are available and need to be executed. The corresponding node in the world model of robot r is, therefore, updated with the local knowledge available, and a utility is computed that reflects the expected number of tasks that r can execute in the corresponding area a_j^n :

$$U^r(a_j^n, t) = \sum_{T \in \mathcal{T}_{a_j^n}^a(t)} \frac{1 - C^r(T)}{\sum_{r_o \neq r} 1 - H_{FW}(r_o, T)}, \quad (3)$$

where the term $C^r(T) \rightarrow [0, 1]$ is the normalized cost for robot r to reach task T , as estimated by the motion planner, computed as the number of actions needed to move from the robots' current location to its target location, normalized by the maximum lower bound cost for the given environment. The term $H_{FW}(r_o, T) \rightarrow [0, 1]$ is a heuristic function that estimates the normalized lower bound cost for any other robot r_o to reach task T or its associated area, computed using the Floyd–Warshall algorithm (also see Section 3.2.3). If, for any reason, Equation (3) is not defined, we set $U^r(a_j^n) = 0$. In other words, the utility of an area a_j^n is higher the more tasks are present, the lower the cost for reaching them (as estimated by the motion planner), and the lower the number of other robots that can potentially execute the same tasks. In case of non-perfect communication, r might not have knowledge of the full range of tasks and other robots, in which case the utility function only considers the set of known robots and tasks.

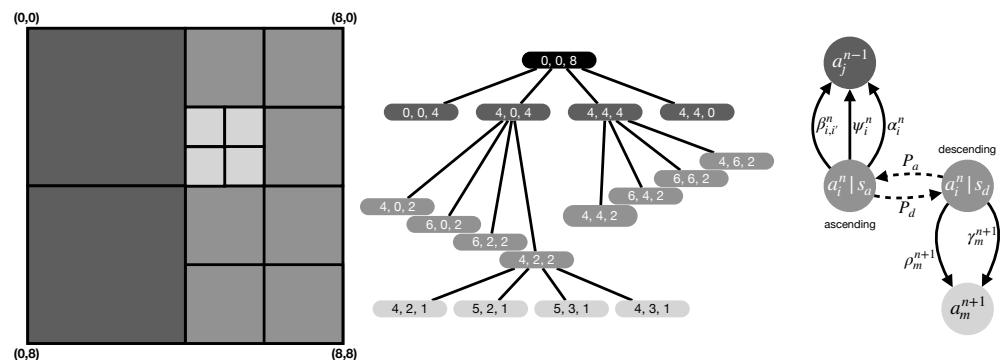


Figure 2. Left: Graphical representation of the considered hierarchical model. In this figure, the environment is discretized in a 8×8 matrix (i.e., $N = 8$) and the corresponding quad-tree is built upon it, where only certain areas are partitioned, for the sake of brevity. Center: The quad-tree has a maximum depth of $\log_2(N) = 3$. Each node of the quadtree contains the identifying tuple (x, y, l) , with the x and y coordinates and size l of the corresponding area. For instance, a robot residing in area $(4, 2, 2)$ has the possibility to descend to any of the four corresponding leaf-nodes, such as the cell $(5, 3, 1)$, exploiting its current knowledge of the sub-tree, or to ascend the quad-tree to area $(4, 0, 4)$ for wider exploration of the environment. Right: probabilistic transitions between possible states, where a robot committed to a_i^n can either move up to the parent node (area a_i^{n-1}) when in the ascending state s_a , or move down to one of the children nodes (area a_m^{n+1}) when in the descending state s_d . Robots first decide whether to switch state $s_a \rightarrow s_d$ ($s_d \rightarrow s_a$) with probability P_d (P_a). Then, transitions can take place that allow a robot to move in the quad-tree, either spontaneous (commitment γ_m^{n+1} , abandonment α_i^n) or interactive (recruitment ρ_m^{n+1} , cross-inhibition $\beta_{i,i'}^n$ and self-inhibition ψ_i^n).

3.2.2. Decentralized Area Assignment

The world model is constantly updated upon receipt of information about new tasks or about the presence of other robots in a certain area. In this way, each robot can compute the utility of each area from its own perspective, which is then used to select a goal area. The decision ultimately corresponds to the assignment of the robot to a node of the hierarchical world model. Here, the selection of a non-leaf node corresponds to the robot deciding to move to and remain in a certain area, while the assignment to a leaf node corresponds to the robot deciding to move to the corresponding area and possibly execute one of the available tasks. To determine the assignment to the nodes of the hierarchical world model, we exploit a decentralized algorithm inspired by the collective decision-making abilities of honeybees [10,28], which was used in different contexts [25,32,33] and also extended for MRTA [11]. Here, we improve over previous studies by extending the algorithm to a hierarchical representation of the world, and by considering the constraints imposed by both the need for collision-free movements and limited communication.

At any time, a robot r_i is considered committed to a node in the quad-tree, i.e., an area $a_i = a_j^n$ in the environment wherein its movements are constrained. Initially, robots start committed to the root node $a_i = a^0$ representing the whole environment. Robots committed

to non-leaf nodes in the quad-tree are not assigned to any specific task and are free to explore the corresponding area performing a random walk [41]. Robots committed to leaf nodes are considered allocated for task execution, whenever tasks become available. At each decision step, a robot can either descend the quad-tree towards leaf nodes or move up the hierarchy towards the root node (see Figure 2). When descending the quad-tree, a robot *exploits* its current knowledge about the children nodes in the quad-tree to choose the area with higher utility (3) for task execution. When ascending the quad-tree, the robot gets committed to the parent node, hence accessing a larger area to *explore* for servicing tasks elsewhere. Switching between the ascending state s_a and the descending state s_d is performed according to a probabilistic threshold-based decision, which can be tuned to balance the exploration-exploitation trade-off (see Figure 2, right). Specifically, a robot changes its state s_i from ascending (s_a) to descending (s_d) with probability P_d per time-step, while it switches from descending to ascending with probability P_a per time-step. Overall, the probability that a robot at any time is found in the ascending (descending) state is $\frac{P_a}{P_a+P_d}$ ($\frac{P_d}{P_a+P_d}$). In this work, we use equal probabilities $P_a = P_d = 0.5$ to obtain a fair exploration-exploitation trade-off.

In both the descending and ascending states, a robot decides whether or not to change the current area on the basis of its local knowledge. The proposed decentralized algorithm exploits both the world model and the knowledge from other robots to determine if neighboring areas are worth being considered. The idea is that areas with high utility should be preferred over areas of low utility, following a proven decision-making design pattern [10,28]. At the same time, overcrowding should be avoided, hence balancing exploitation among different areas. To this end, robots interact with each other to determine the best possible allocation. Note that, to move towards areas at the same hierarchical level, a transition through the parent node is required.

The selection of a destination node in the quad-tree is performed according to four stochastic processes: (i) *commitment* to an area at a lower hierarchical level, (ii) *recruitment* by a peer robot committed to an area at a lower hierarchical level, (iii) *abandonment* of the current area to move up the hierarchy, and (iv) *inhibition* determined by a peer robot—either committed to the same (self-inhibition) or to a same-level area (cross-inhibition)—leading to a move up in the hierarchy (see Figure 2, right panel).

Hence, commitment and recruitment represent the possible descending transitions available when the robot is in s_d , allowing a robot to move down the hierarchy towards the leaf nodes. Abandonment and inhibition represent the possible ascending transitions—available when the robot is in s_a —allowing a robot to move up the hierarchy towards the root node. Specifically, self-inhibition avoids overcrowding, pushing robots to explore other areas when the current one has too many robots working around, while cross-inhibition favors the selection of better areas of the same hierarchical levels by promoting exploration at the higher level [10,28]. We refer to commitment and abandonment as *spontaneous processes*, as they are determined by individual knowledge, while recruitment and inhibition are *interactive processes* that take place upon interaction with a different robot. All transitions between nodes are stochastic, and the transition probability is computed on the basis of the utility associated to the quad-tree nodes, as defined in Equation (3) (see Figure 2, right). Note that, at any time, there may be a non-null probability of remaining in the current area, depending on the probabilities of the available transitions.

Descending Transitions

The robot r located in area a_i^n and in state s_d can change node by means of commitment or recruitment (see Figure 2, right). Commitment represents the spontaneous decision of r to move to one of the four children nodes a_m^{n+1} according to the perceived utility. For each area a_m^{n+1} , the transition probability is computed proportionally to its utility as follows:

$$\gamma_m^{n+1} = k \cdot U^r(a_m^{n+1}), \quad (4)$$

where k is a control parameter for spontaneous processes. Conversely, recruitment allows a robot r to get recruited by another robot. At any time, a robot r_o is randomly chosen among the robots known by r , and—if it belongs to the sub-tree of a child node a_m^{n+1} —a transition probability is computed according to the utility shared by r_o , based on the assumption that U^{r_o} provides a better estimation for area a_m^{n+1} since r_o is located in that region. Moreover, thanks to the coupling with the motion planner, U^{r_o} embeds the motion costs as estimated by r_o , which serve as an indication of the perceived ease of motion within that area. Additionally, recruitment provides a means to tune the assignment process to the density of robots in a given area, as the probability of selecting a robot assigned to a certain area is proportional to the robot density of that area. Hence, the recruitment transition probability is computed as follows:

$$\rho_m^{n+1} = h \cdot U^{r_o}(a_m^{n+1}), \quad (5)$$

where h is a control parameter for interactive processes. Note that, at any time, only a single neighboring robot r_o is considered, and recruitment takes place only if it belongs to a sub-tree of the children nodes of a_i^n . In all other cases, no recruitment is considered. In this way, the overall recruitment rate of robots to a given area is proportional to the population of robots already committed to that area [10].

Ascending Transitions

When in state s_a and in area a_i^n , robot r may decide to move to the parent node a_j^{n-1} through abandonment, self-inhibition or cross-inhibition (see Figure 2, right). Spontaneous abandonment is the more probable the lower is the utility of the current area:

$$\alpha_i^n = k(1 - U^r(a_i^n)). \quad (6)$$

On the other hand, a robot r residing in area a_i^n can be inhibited by another robot r_o belonging to the sub-tree either from the same area (self-inhibition) or from a different area $a_{i'}^n$ at the same hierarchical level n (i.e., a_i^n and $a_{i'}^n$ share the same parent node, that is, they are sibling nodes). Self-inhibition is used to cope with overcrowding, where several robots share the same region, hence impairing robot motion. It is defined as:

$$\psi_i^n = h \cdot U^{r_o}(a_i^n) \cdot \mathcal{H}(R(a_i^n) - \theta_\psi \mathcal{C}(a_i^n)), \quad (7)$$

where $R(a_i^n)$ represents the number of robots in area a_i , and \mathcal{H} is the Heaviside step function, which enables the process only if the population in the area exceeds a fraction $\theta_\psi = 3/4$ of its capacity $\mathcal{C}(a_i^n)$, defined as the number of cells that do not contain an obstacle. Overall, self-inhibition balances recruitment to an area when the population density is too high (also see Reference [11]). Conversely, Cross-inhibition forces a robot i to abandon a region with low utility in favor of a region i' with higher utility. It is used to focus the assignment to areas of high utility, allowing the robots to abandon their current area (i.e., getting assigned to the parent node) to explore one of the sibling areas. The purpose of this process is to balance poor commitments that might arise from the probabilistic nature of the system and outdated/limited knowledge. Hence, given a robot r_o residing in the sub-tree of area $a_{i'}^n$, $i' \neq i$, the probability of cross-inhibition is proportional to the utility $U^{r_o}(a_{i'}^n)$, as defined in the following:

$$\beta_{i,i'}^n = h \cdot U^{r_o}(a_{i'}^n) \cdot \mathcal{H}(\theta_\beta \mathcal{C}(a_{i'}^n) - R(a_{i'}^n)), \quad i \neq i', \quad (8)$$

where the Heaviside function \mathcal{H} enables the process only if the population in area i' does not exceed one fourth of the capacity, i.e., $\theta_\beta = 0.25$, again to limit potential overcrowding. Similar to recruitment, also in this case inhibition happens only if r_o belongs to areas at the same hierarchical level or below, otherwise the transition probability is null.

Overall Algorithm and Control Parameters

A pseudo-code of the steps followed by the robot during the area assignment process is provided in Algorithm 1. The robot moves in the hierarchical tree one node at the time based on ascending and descending transition, respectively, selected according to two a-priori fixed probabilities P_a and P_d (i.e., probabilistic selection of the robot state s_i and hence the next transition type, ascending or descending). Next, transitions probabilities are computed according to the currently selected neighboring robot and the considered areas (see Algorithm 1, lines 7 and 9) and are, respectively, associated to the locally computed interactive and spontaneous processes values as in Equation (4) to (8). The assignment process randomly extract a new hierarchical area (line 10) on the basis of computed transition probabilities and propagate this decision up, to all its ancestors in its local quad-tree. We assume the assignment process to be faster than robot motion, so that D decisions are performed while a robot is moving between two adjacent cells. In this study, we heuristically set D to be equal to the depth of the quad-tree M . Increasing or decreasing D can improve or diminish the tree exploration. Last, a new motion plan is made (line 12) only if the assignment process changes the current area after D steps, hence requiring to determine a new destination (either a chosen task or a randomly determined cell).

As originally proposed by Reina et al. [28], we have weighted interactive and spontaneous processes by two free parameters, respectively, h and k , used to define the ratio h/k between interactive and spontaneous processes. As shown by Albani et al. [11], in the simple non-hierarchical case, changing this ratio allows switching between a utility-proportional deployment where robots are allocated to areas proportionally to their utility when the ratio is low, to a utility-responsive collective decision where the optimal number of robots is allocated to the area that currently has the highest utility (higher values for h/k). Indeed, when $h = 0$, there is no recruitment or inhibition, and only commitment or abandonment are considered, which are driven directly by knowledge of the area utility. Relying more on interactive transitions allows to make a collective decision towards areas of higher utility, while balancing the deployment to avoid overcrowding, thanks to self-inhibition. In this study, we consider $h/k = 0.25$ ($h = 0.2$, $k = 0.8$), which favors spontaneous transitions while still exploiting interactions to collectively choose the best area where to service tasks.

3.2.3. Decentralized Task Assignment and Motion Planning

Task assignment and motion planning are performed concurrently by exploiting a priority-based scheme that assumes robots are ordered and plan sequentially, treating the plans of previous robots as known dynamic obstacles. Such an approach is in general not complete, but scales linearly with the number of robots. We implement this approach in a distributed way, similar to previous work [12], as outlined below.

At the beginning, we pre-compute the all-pair shortest paths in a pre-processing step using the Floyd–Warshall algorithm. This can be computed in a decentralized way by each robot or pre-computed and given to each robot as part of the environment description. We use the all-pair shortest paths in two ways. First, we normalize the all-pair shortest paths and use the resulting values for our heuristic function $H_{FW}(\cdot, \cdot)$, which informs our utility function (see Equation (3)). Second, the knowledge of the all-pair shortest paths is an admissible heuristic that is used within the path planning algorithm to speed up the computations.

To this end, as the path planning algorithm, we use Safe Interval Path Planning (SIPP) [42], a variant of A* that can plan efficiently in the space-time domain if the trajectories of dynamic obstacles are known a-priori. We make an adjustment to the algorithm to better cooperate with the assignment algorithm. We let the decision-making algorithm reason on goal regions rather than single cells: when a goal region is selected as the next assignment by the decision-making algorithm, we consider any cell in the goal region a_g as a possible goal, therefore solving a task assignment and path planning problem simultaneously. Specifically, each cell containing a task within a leaf node of the quad-tree

is evaluated and the one with lowest cost is selected as destination. This allows the assignment process to reason about the expected number of tasks in the goal region rather than having to consider low-level congestion information of the path planner. We note that there is no explicit synchronization and that our approach is fully distributed and asynchronous.

We then broadcast our current plan to neighboring robots, that update their local knowledge accordingly using received costs to update their local utility (also see Algorithm 1). Our approach does not rely on a fixed priority (e.g., Reference [12]), or implicit synchronization using tokens (e.g., Reference [43]). However, a disadvantage is that safety can only be guaranteed with the assumed sensing capabilities of neighboring robots. Reactive obstacle avoidance prevents collisions that might be caused by our probabilistic communication model or dynamic ordering of the robots: a movement is performed only if the target cell is free, otherwise re-planning is triggered.

4. Experimental Evaluation

We evaluate our approach empirically by exploiting a custom 2D simulation environment written in C++ implementing a grid world (Section 4.1). On this setup, we compare results for the proposed algorithm against two baselines: (i) a greedy task selection with task swapping, and (ii) an auction-based mechanism relying on the Contract Net Protocol (CNP) framework [44] (see Section 4.2). To ensure a fair comparison, all algorithms rely on the same motion planner and have access to the same information. The experimental setup is detailed in Section 4.3, and the obtained results are analyzed in Section 4.4.

4.1. Service Robotics Simulation

Figure 3 shows the four different environments used in the experiments, featuring different size, i.e., different number of cells: (i) a small two rooms environment, where the total area ($N = 16$) is split in half by a wall with a 2-cells opening in the center (*split16*); (ii) an empty environment of size $N = 32$ (*empty32*); (iii) a facility-like environment of size $N = 32$, with several rooms connected by hallways and one big room that serves as a warehouse (*facility*); and (iv) a rough reproduction of the floor of the Automatic Coordination of Teams Lab (Robotics and Autonomous Systems Center, University of South California) of size $N = 64$ (*lab*). For the purpose of the experiments, we consider each cell c_j in the grid world to be either free or occupied. A free cell represents empty space where a robot can move and a task appear. For the *empty32* environment, task locations are uniformly distributed. In the *split16*, *facility* and *lab* scenarios, potential task locations are specified such that they do not block narrow passages or corridors and only appear inside rooms (see Figure 3). An occupied cell is either an empty cell already taken by a robot—and does not allow the presence of a second robot—or represents a wall or any obstacle not allowing the presence of robots and tasks. Walls and obstacles have an impact on the communication range, following the model introduced in Equations (1) and (2). Figure 4 shows how the environment configuration impacts on the communication range for the *facility* and *lab* environments.

As already introduced in Section 3.2.1, we consider R identical holonomic robots. At each time-step the set of robots is randomly shuffled to remove any bias introduced by ordered sequential decisions. This allows to avoid ties since we use the reshuffled sequential order of robots to prioritize the decisions. At each step, a robot is asked to perform one of the available actions: wait at its current position, move to an adjacent free cell, or work on a task located in the cell it is occupying. Robot actions are performed after the decision phase that constitutes the main contribution of this work and is summarized in Algorithm 1.

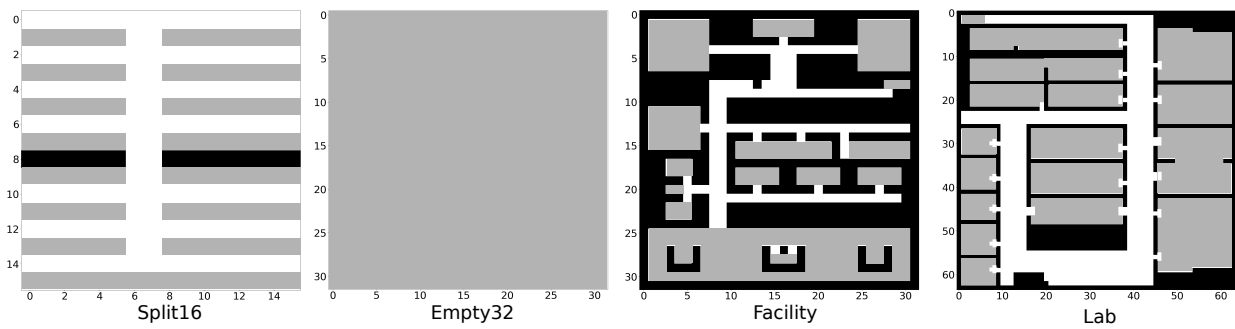


Figure 3. The four environments used in the experimental setup. From **left to right**: the `split16` environment of size 16×16 , the `empty32` environment of size 32×32 , the `facility` environment of size 32×32 , and last the `lab` environment of size 64×64 . Black represent walls, white and grey areas are free cells where robots can move, but tasks can appear only in the gray shadowed areas.

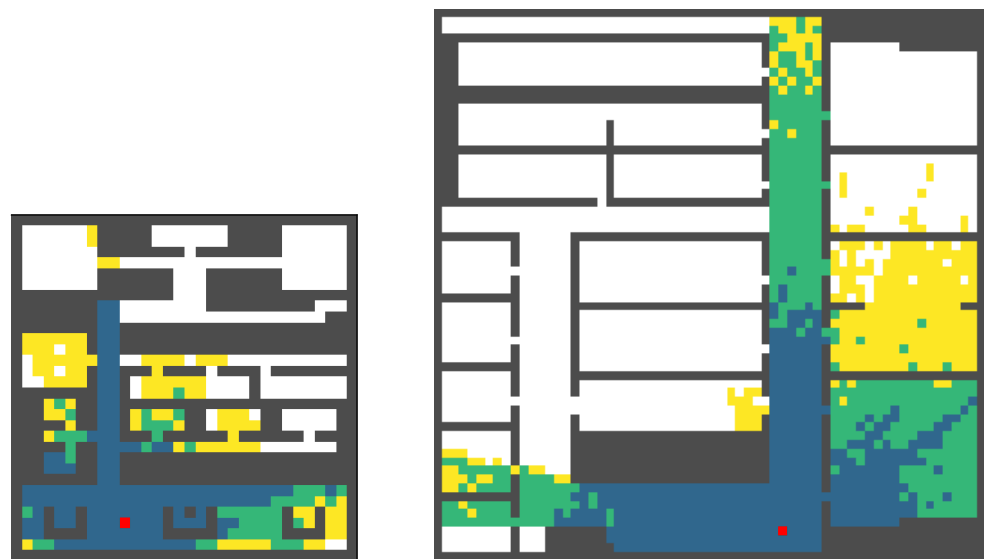


Figure 4. Example of communication range reduction in the `facility` (**left**) and the `lab` (**right**) environments. With respect to the position marked in red, the plot highlights areas within communication range (with probability 99%) for $S = -80$ (blue), for $S = -100$ (green), and for $S = -120$ (yellow).

4.2. Greedy and Auction-Based Strategies

For MRTA, a greedy approach is suitable because “anecdotal evidence suggests that the greedy algorithm works extremely well” [36]. Indeed, with high robot density and low task generation rate, the greedy approach performs close to the optimal solution due to its intrinsic reactive property, provided that conflicts in task assignment can be easily resolved. Similarly, with few robots but many tasks, the robotic system is often fully occupied all the time, resulting in the maximum possible efficiency. For our baseline, each robot r chooses the closest available task T_i . We propose a variation of the trivial greedy approach by including a task swapping procedure. If a robot r_p committed to task T_i receives information related to the commitment of a second robot r_q to the same task, a conflict resolution procedure begins. Robot r_p compares its utility against r_q 's utility and drops its current assignment only if $U^{r_p}(T_i) < U^{r_q}(T_i) \implies H_{FW}(r_p, T_i) < H_{FW}(r_q, T_i)$. Such greedy variant does not require the concept of areas (see Section 3.2.1); thus, the utility reduces to the heuristic h .

Our proposed auction-based baseline relies on the Contract Net Protocol scheme (CNP) [44], upon which several other systems have been built (e.g., Reference [45]). Our implementation models self-interested robots without the possibility of subcontracts and transfers [37]. During every selection step, if not assigned with any task, a robot starts the announcement phase and becomes the manager for the closest task distance-wise. During

the submission phase, any other non-committed robot within the communication range answers with its current bid, equals to the heuristic H_{FW} as defined in above. The selection and contract phases proceed as normal giving priority to lower bids, i.e., robots with a shorter path to the task, and we break ties by following a FIFO selection. Note that, due to the announcement phase, task-related information might be propagated farther than the original task range, potentially increasing the number of robots involved in the transaction. In obstacle-free areas and, in particular, in absence of environmental bottlenecks one can imagine the CNP baseline to act as a improved variation of the greedy approach, but in complex scenarios it has several differences. The assignment process is synchronous and considers all the robots with related utilities at once, thus locally centralizing the system for a small time-frame and generating a bigger communication overhead than the other two proposed approaches.

4.3. Experimental Setup

To model our service robot domain, tasks are generated within two macro-areas at a time, chosen from 16 areas that cut the environment in identically-sized regions. Chosen areas generate tasks at a fixed task generation rate, set to 1 task per time-step in all environments, unless the area is completely full. Additionally, the enabled macro-areas change during the simulation eight times, resulting in a dynamic demand across the environment. Tasks do not disappear after a region change and can still be served by the robots in the area. We believe this to be an effective and realistic approximation of the service robot problem, in which a region—once served—does not require further attention for a certain amount of time.

Unless otherwise specified, the settings for the experiments are as follows. The work time t_i^w for all tasks is constant and fixed to 5 steps, i.e., the number of steps a robot has to remain over a task to consider it accomplished. The number of robots involved in the simulations is related to the number of free-cells in the considered environment: 25 for *split16* and *facility*, 50 for *empty32*, and 150 for *lab*. We consider nodes associated to areas of size 2 (i.e., $a_i^m = (x_i, y_i, 2)$, with $m = \log_2 N - 1$) as leaves. The number of decisions per time-step (i.e., movements in the hierarchical tree) is set to $D = M = \log_2(N) - 1$, that is, the depth of the quad-tree, and the probability to switch from the ascending to the descending state, and vice versa, are set to $P_a = P_d = 0.5$. Each simulation was executed for 300 time-steps. All the parameter values have been determined on the basis of previous work or empirical evaluations and are not specifically tuned based on the environment to be tested (for a discussion about this choice, see Section 5).

4.4. Results

The evaluation of performance of the proposed HTAPF approach is compared against the greedy and CNP approaches in different experimental settings. For each setting, we performed 50 independent simulations with different random seeds and random initial robot positions. In each simulation, we measure the number of completed tasks looking for the approach that maximizes the demand satisfaction. For each tested environment and experimental condition, a one-way non-parametric ANOVA (Kruskal–Wallis) test was conducted to examine the differences in the completed tasks according to the different strategies considered (greedy, CNP, and HTAPF). Moreover, the post-hoc Dunn's test was employed to examine the significance of all possible pairwise comparisons ($p < 0.05$).

Figure 5 provides the results obtained in all environments in ideal conditions, that is, when communication among robots is perfect and there is no failure. The former aspect implies that every robot can receive without error the information about (i) tasks available and (ii) motion plans from other robots. This condition is clearly unrealistic for practical scenarios but is useful as a baseline on which to evaluate the impact of limitations, like local communication and possible failures.

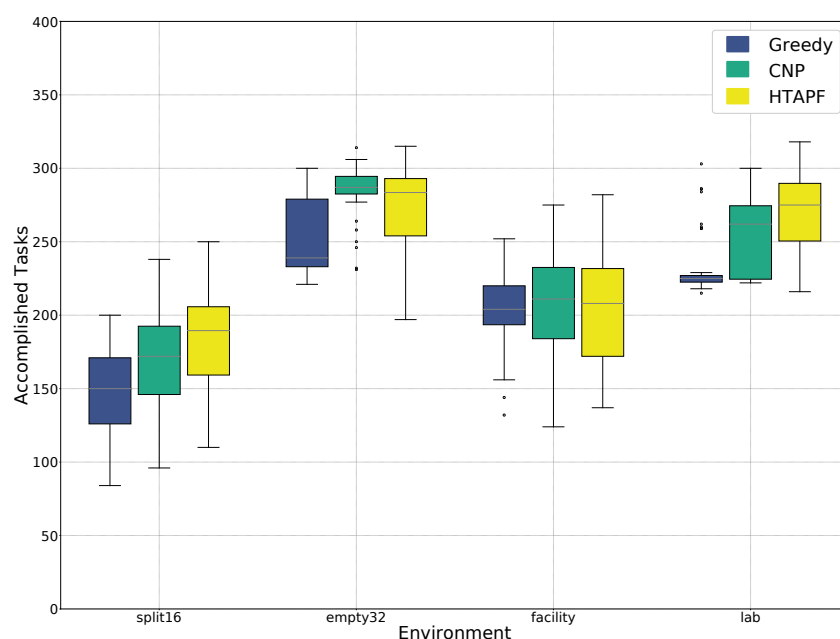


Figure 5. Number of accomplished tasks by the different approaches (greedy, Contract Net Protocol (CNP), and HTAPF) across the different environments, obtained with unlimited communication and no robot failures. For each condition, 50 independent simulations are performed and the data displayed by a box-and-whiskers plot. Each box represents the inter-quartile range of the data, and the median is displayed as a line cutting the box in two parts. Whiskers extend to 1.5 times the top and bottom quartiles. Dots represent outliers.

Generally speaking, we observe good results from HTAPF, which performs on par or better than the greedy and CNP approaches (see Figure 5 and Table 1 for the outcome of statistical tests). Specifically, in *split16* HTAPF performs significantly better than both greedy and CNP. Here, the bottleneck constituted by the small opening between the two rooms may produce congestion when the task assignment to different areas is not properly managed, which is something that neither CNP nor the greedy approach properly do. In *empty32*, instead, CNP is properly assigning tasks to the robots and performs slightly better than HTAPF, while the greedy approach presents a significantly lower performance, due to the inability to quickly adapt to changes in the areas where tasks are spawned. It is not surprising that CNP performs well in this environment, considering that there is no limitation in motion from walls or obstacles, and, therefore, the bidding estimates closely correspond to the actual costs incurred, as motion planning has ample freedom to select a collision-free route. This is not the case for the *facility* environment, where the motion constraints in relocating from one area to another have a similar impact on all approaches due to the narrow corridors that in some cases force the passage of a single robot at the time. Finally, the most challenging environment in terms of size and structure is *lab*, where HTAPF again outperforms all other approaches.

Note that, thanks to the perfect communication, CNP is close to a centralized approach for task assignment, as all robots can participate to the bidding process. In addition, the greedy approach can take advantage of the perfect information available, making task switching very efficient. Hence, the competitor strategies for HTAPF are working in the best possible conditions for their characteristics. HTAPF also exploits all the available information well, but the stochastic task assignment maintains some randomness that best suits for scenarios with limited knowledge, as we show later. Additionally, the communication requirements for HTAPF are minimal, while CNP imposes a considerable communication load. Hence, the advantage presented by HTAPF in these baseline experiments is commendable.

Table 1. Results of the statistical tests on the simulations performed with unlimited communication and no robot failures.

| | | split16 | | | empty32 | | |
|------------|----------------|------------------------------------|----------------------|----------------------|-------------------------------------|----------------------|-----------------------|
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| | median | 150 | 172 | 189.5 | 239 | 287 | 283.5 |
| | Kruskal–Wallis | $H = 47.7, p = 1.1 \times 10^{-4}$ | | | $H = 43.8, p = 7.0 \times 10^{-4}$ | | |
| p -value | greedy | – | 9.1×10^{-4} | 6.7×10^{-8} | – | 5.8×10^{-9} | 2.1×10^{-4} |
| | CNP | – | – | 3.6×10^{-2} | – | – | 3.6×10^{-2} |
| | | facility | | | lab | | |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| | median | 204 | 211 | 208 | 225 | 262 | 275 |
| | Kruskal–Wallis | $H = 4.7, p = 0.31$ | | | $H = 62.0, p = 1.1 \times 10^{-12}$ | | |
| p -value | greedy | – | × | × | – | 4.3×10^{-5} | 1.4×10^{-12} |
| | CNP | – | – | × | – | – | 2.6×10^{-3} |

A more realistic setting is when communication is limited in range, due to limited emission power and imperfect receiver sensitivity (We use the following parameters for the communication model for Equations (1) and (2): $P_0 = -20$, $\eta = 5.6$, $W = 10$, $\sigma = 3.1$, $FER_S = 0.08$, $\gamma = 1$, $N_b = -100$, and $N_{th} = -100$). We study the influence of limited communication by using three different sensitivities, $S \in \{-120, -100, -80\}$ which substantially limit the range at which broadcast messages can be received (see Figure 4 for some examples). Note that both robots and tasks rely on the same communication model, meaning that tasks are known to a robot only when within range. Figure 6 shows the results for all environments and all approaches considered, while Table 2 reports the results of the statistical tests. Additionally, we also considered failures in robots happening at different rates (0.05 and 0.1 expected failures per time-step, corresponding to an average of 15 and 30 failing robots per simulation, respectively), which turn robots into static obstacles that need to be avoided, therefore influencing the overall task execution abilities of the system (see Figure 7 and Table 3 for the results of the statistical tests).

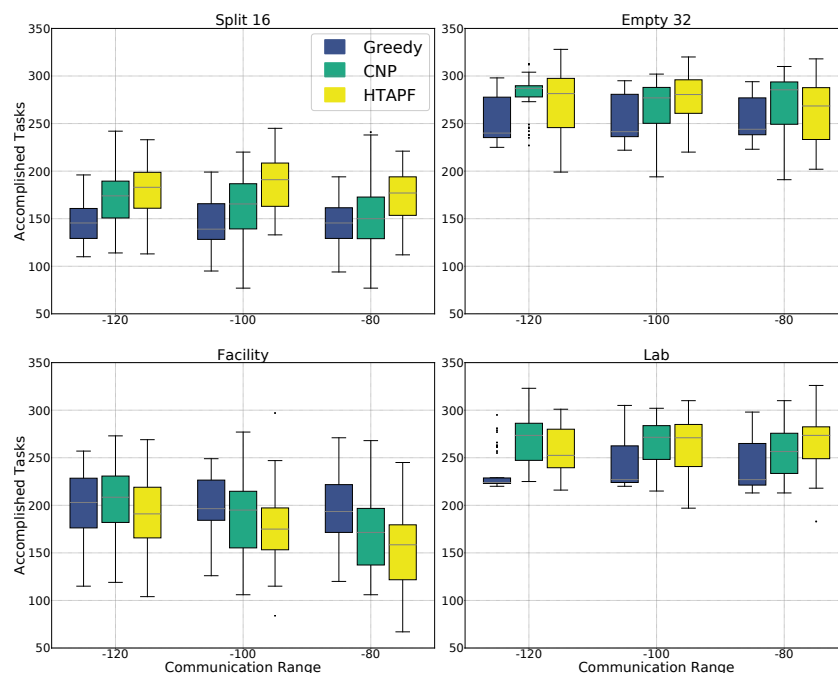
**Figure 6.** Number of accomplished tasks by the different approaches (greedy, CNP, and HTAPF) across the different environments, obtained with limited communication and no robot failures.

Table 2. Results of the statistical tests on the simulations performed with different limited communication range.

| | | split16 | | | empty32 | | |
|------------|----------------|--------------------------------------|----------------------|----------------------|-------------------------------------|-----------------------|----------------------|
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| $S = -120$ | median | 145.5 | 174 | 183 | 240 | 287 | 281.5 |
| | Kruskal–Wallis | $H = 58.5, p = 5.9 \times 10^{-12}$ | | | $H = 34.4, p = 6.0 \times 10^{-7}$ | | |
| | greedy | – | 1.7×10^{-4} | 1.9×10^{-7} | – | 8.8×10^{-7} | 4.6×10^{-5} |
| | CNP | – | – | 1.5×10^{-1} | – | – | 4.0×10^{-1} |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| $S = -100$ | median | 139 | 165.5 | 191 | 241.5 | 277 | 280.5 |
| | Kruskal–Wallis | $H = 72.86, p = 5.6 \times 10^{-15}$ | | | $H = 21.67, p = 2.3 \times 10^{-4}$ | | |
| | greedy | – | 1.2×10^{-2} | 3.7×10^{-9} | – | 6.6×10^{-3} | 2.1×10^{-5} |
| | CNP | – | – | 7.3×10^{-4} | – | – | 1.2×10^{-1} |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| $S = -80$ | median | 145.5 | 150 | 177 | 244 | 285.5 | 268.5 |
| | Kruskal–Wallis | $H = 35.5, p = 3.7 \times 10^{-7}$ | | | $H = 13.7, p = 8.3 \times 10^{-3}$ | | |
| | greedy | – | 2.6×10^{-1} | 3.8×10^{-5} | – | 1.1×10^{-3} | 2.5×10^{-1} |
| | CNP | – | – | 2.7×10^{-3} | – | – | 3.3×10^{-2} |
| | | facility | | | lab | | |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| $S = -120$ | median | 203 | 208.5 | 191 | 224 | 273.5 | 252.5 |
| | Kruskal–Wallis | $H = 8.2, p = 8.3 \times 10^{-2}$ | | | $H = 60.5, p = 2.3 \times 10^{-12}$ | | |
| | greedy | – | × | × | – | 3.2×10^{-12} | 3.7×10^{-8} |
| | CNP | – | – | × | – | – | 1.4×10^{-1} |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| $S = -100$ | median | 196.5 | 195 | 175 | 227 | 271.5 | 271 |
| | Kruskal–Wallis | $H = 17.2, p = 1.8 \times 10^{-3}$ | | | $H = 43.1, p = 1.0 \times 10^{-8}$ | | |
| | greedy | – | 1.2×10^{-1} | 8.5×10^{-4} | – | 7.4×10^{-8} | 1.5×10^{-7} |
| | CNP | – | – | 7.4×10^{-2} | – | – | 8.9×10^{-1} |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| $S = -80$ | median | 194 | 171.5 | 158.5 | 227 | 256.5 | 273.5 |
| | Kruskal–Wallis | $H = 38.4, p = 9.1 \times 10^{-8}$ | | | $H = 35.8, p = 3.2 \times 10^{-7}$ | | |
| | greedy | – | 3.8×10^{-3} | 2.0×10^{-6} | – | 2.0×10^{-2} | 6.0×10^{-5} |
| | CNP | – | – | 6.4×10^{-2} | – | – | 2.8×10^{-2} |

Generally speaking, the value of HTAPF is confirmed practically everywhere. In the split16 environment, HTAPF proves robust against communication limitations, and always performs significantly better than both greedy and CNP, apart when a sensitivity of $S = -120$ is considered where no significant difference was found between HTAPF and CNP (see Figure 6, top-left panel, and Table 2). The small dimensions and the limited number of walls entail that limitations in communication are not too evident, substantially confirming the baselines results. In addition, in case of failures, HTAPF presents a robust behavior, only marginally affected by the unavailable robots, but still outperforming the competitor approaches (Figure 7 top-left and Table 3).

In the empty32 environment, limitations in communication have practically no impact, and we observe again that CNP and HTAPF are performing on par and significantly better

than the greedy approach, although a higher variability for CNP is observed when the communication sensitivity is worse ($S = -100$ and $S = -80$). In the latter case, CNP performs slightly better, thanks to the ease of motion granted to the robots (see Figure 6, top-right, and Table 2). When robot failures are introduced, instead, HTAPF performs better than the other approaches for the highest failure rate, as it proves to be more robust in re-allocating the residual robots when needed. Indeed, there is no significant drop in performance with increasing failure rates (see Figure 7, top-right, and Table 3).

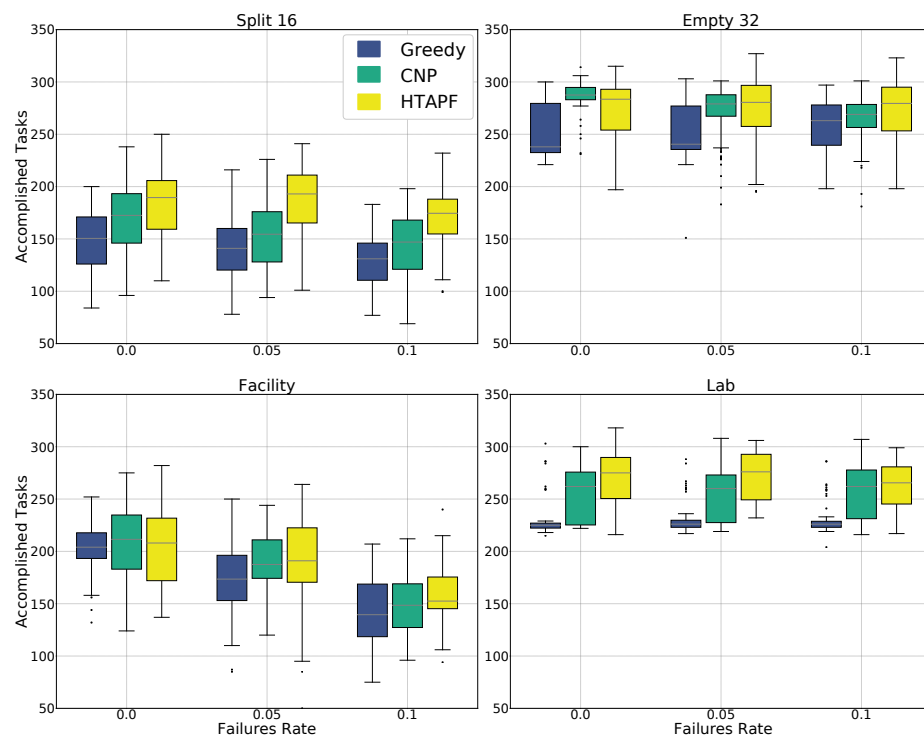


Figure 7. Number of accomplished tasks by the different approaches (greedy, CNP, and HTAPF) across the different environments, obtained with unlimited communication and variable robot failures rate.

The facility environment is the one that presents the harshest communication limitations, with thick walls preventing a proper exchange of information. In such conditions, the performance of CNP and HTAPF gets worse and the greedy approach proves more robust as it does not rely much on communication. Significant differences are observed when $S \geq -100$, where the greedy approach outperforms both CNP and HTAPF (see Figure 6, bottom-left, and Table 2). In case of failures, all approaches show a similar drop in performance, due to failed robots blocking corridors or doorways, and no algorithm is more robust than the others (see Figure 7, bottom-left, and Table 3).

Finally, the lab environment confirms the robustness of HTAPF both with respect to limited communication and robot failures. In the former case (Figure 6 bottom-right and Table 2), HTAPF always outperforms the greedy approach and is significantly better than CNP when $S = -80$. In the latter case, HTAPF is significantly better than both approaches but for the maximum failure rate, where the advantage is marginal (Figure 7 bottom-right and Table 3). It is worth noting that the performance of HTAPF is rather constant despite the complexity of the environment and the introduced limitations and failures, which suggests that the algorithm properly exploits the available robots, flexibly deploying them to the areas that mostly need work to be performed, quickly adapting to variations in the service demands.

Table 3. Results of the statistical tests on the simulations performed with different rates of robot failures.

| | | split16 | | | empty32 | | |
|-----------|----------------|------------------------------------|----------------------|----------------------|------------------------------------|----------------------|-----------------------|
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| rate 0.05 | median | 141 | 154.5 | 193 | 240.5 | 279.5 | 280.5 |
| | Kruskal–Wallis | $H = 30.9, p = 1.9 \times 10^{-7}$ | | | $H = 5.4, p = 6.6 \times 10^{-2}$ | | |
| | greedy | – | 2.2×10^{-1} | 6.4×10^{-9} | – | × | × |
| | CNP | – | – | 4.5×10^{-6} | – | – | × |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| rate 0.1 | median | 131 | 147 | 174.5 | 263 | 269 | 279.5 |
| | Kruskal–Wallis | $H = 22.5, p = 1.2 \times 10^{-5}$ | | | $H = 6.25, p = 4.4 \times 10^{-2}$ | | |
| | greedy | – | 3.7×10^{-2} | 3.1×10^{-9} | – | 6.4×10^{-1} | 5.3×10^{-3} |
| | CNP | – | – | 1.2×10^{-4} | – | – | 2.0×10^{-2} |
| | | facility | | | lab | | |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| rate 0.05 | median | 173.5 | 187.5 | 191 | 226 | 260 | 276 |
| | Kruskal–Wallis | $H = 0.2, p = 9.1 \times 10^{-1}$ | | | $H = 11.3, p = 3.4 \times 10^{-3}$ | | |
| | greedy | – | × | × | – | 5.0×10^{-6} | 2.0×10^{-15} |
| | CNP | – | – | × | – | – | 7.7×10^{-4} |
| | | greedy | CNP | HTAPF | greedy | CNP | HTAPF |
| rate 0.1 | median | 139.5 | 148.5 | 152.5 | 226 | 262 | 265.5 |
| | Kruskal–Wallis | $H = 2.2, p = 3.4 \times 10^{-1}$ | | | $H = 2.3, p = 3.2 \times 10^{-1}$ | | |
| | greedy | – | × | × | – | × | × |
| | CNP | – | – | × | – | – | × |

5. Conclusions

We presented HTAPF, a novel fully decentralized algorithm that couples a bio-inspired task allocation algorithm with a distributed motion planner for persistent robotic service tasks. The task allocation component significantly extends our prior work by supporting multiple hierarchical decision levels and a utility function that directly considers the (estimated) cost of robot motions. The motion planning component is based on a distributed prioritized planning scheme with goal assignment within a specified region. Both approaches are tightly coupled because the motion planner's cost estimates are directly considered for area/task assignment. Tests on different environments show that the proposed solution performs well in all tested cases, validated by statistical tests and compared to two baseline algorithms. We also demonstrate that HTAPF is robust with respect to limited communication and induced robot failures.

Although the HTAPF algorithm is characterized by several free parameters, no specific tuning has been performed for the experiments presented here. Proposed values are chosen according to previous work and reasonable heuristics, and have been kept fixed across different problem instances to demonstrate the wide applicability of the proposed approach. While improvements in performance are possible with a detailed tuning of all the parameters for specific experimental settings, these results demonstrate that the system properly generalizes to different conditions, different number of robots and different introduced limitations. Indeed, we have shown that performance is satisfactory also in those cases where the nature of the environment (e.g., robot failures or communication limitation) is not known in advance. We believe that the proposed solution is a good fit for many realistic service task scenarios or situations where robots with limited power, reliability, or communication are taken into account. Examples of the latter are unstructured

environments that lack a central entity, a distributed network that covers the entire area, or scenarios where the system can be deployed without specific infrastructures, such as search and rescue scenarios and delivery systems for large industrial settings.

Future work includes the study of the macroscopic dynamics of the system and the adaptive tuning of the tree exploration. A deep analysis of a macroscopic model representing the decision dynamics over the quad-tree can provide a better understanding of the dynamics characterizing the assignment process, leading to the identification of parameterizations that provide optimal performance and desired system properties, possibly including different parameters for different hierarchical levels in the quad-tree. Adaptive solutions to the tuning of some key parameters are important to dynamically respond to problem requirements. For instance, the possibility of online tuning the probability of descending the quad-tree can provide an important means to optimize the exploration-exploitation trade-off.

Author Contributions: Conceptualization D.A., W.H., V.T., N.A. and D.N.; methodology D.A., W.H. and V.T.; software D.A., W.H. and V.T.; writing–review and editing, D.A., W.H., V.T., N.A. and D.N.; funding acquisition V.T. and N.A. All authors have read and agreed to the published version of the manuscript.

Funding: V.T. and D.A. acknowledge support from the Office of Naval Research Global (ONRG) for the project “Collective Decisions in Dynamic Environments” (Award N62909-18-1-2093). V.T. acknowledges support from the project “TAILOR” (H2020-ICT-48 GA: 952215). N.A. and W.H. acknowledge support from NSF under grant IIS-1724392.

Institutional Review Board Statement: Not Applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Garcia, E.; Jimenez, M.A.; De Santos, P.G.; Armada, M. The evolution of robotics research. *IEEE Robot. Autom. Mag.* **2007**, *14*, 90–103. doi:10.1109/MRA.2007.339608.
2. D’Andrea, R. Guest Editorial: A Revolution in the Warehouse: A Retrospective on Kiva Systems and the Grand Challenges Ahead. *IEEE Trans. Autom. Sci. Eng.* **2012**, *9*, 638–639.
3. Hönig, W.; Kiesel, S.; Tinka, A.; Durham, J.W.; Ayanian, N. Persistent and Robust Execution of MAPF Schedules in Warehouses. *IEEE Robot. Autom. Lett.* **2019**, *4*, 1125–1131. doi:10.1109/LRA.2019.2894217.
4. van Henten, E.; Bac, C.; Hemming, J.; Edan, Y. Robotics in protected cultivation. *IFAC Proc. Vol.* **2013**, *46*, 170–177.
5. Ferrer, A.J.; Marquès, J.M.; Jorba, J. Towards the Decentralised Cloud: Survey on Approaches and Challenges for Mobile, Ad Hoc, and Edge Computing. *ACM Comput. Surv.* **2019**, *51*, 1–36. doi:10.1145/3243929.
6. Dorigo, M.; Theraulaz, G.; Trianni, V. Reflections on the future of swarm robotics. *Sci. Robot.* **2020**, *5*, eabe4385.
7. Farinelli, A.; Iocchi, L.; Nardi, D. Distributed on-line dynamic task assignment for multi-robot patrolling. *Auton. Robot.* **2017**, *41*, 1321–1345.
8. de Lope, J.; Maravall, D.; Quiñonez, Y. Self-organizing techniques to improve the decentralized multi-task distribution in multi-robot systems. *Neurocomputing* **2015**, *163*, 47–55.
9. Sharon, G.; Stern, R.; Felner, A.; Sturtevant, N.R. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.* **2015**, *219*, 40–66.
10. Reina, A.; Valentini, G.; Fernández-Oto, C.; Dorigo, M.; Trianni, V. A Design Pattern for Decentralised Decision Making. *PLoS ONE* **2015**, *10*, e0140950-18.
11. Albani, D.; Manoni, T.; Nardi, D.; Trianni, V. Dynamic UAV Swarm Deployment for Non-Uniform Coverage. In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Stockholm, Sweden, 10–15 July 2018; pp. 523–531.
12. Velagapudi, P.; Sycara, K.P.; Scerri, P. Decentralized prioritized planning in large multirobot teams. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4603–4609.
13. Sabattini, L.; Digani, V.; Secchi, C.; Fantuzzi, C. Optimized simultaneous conflict-free task assignment and path planning for multi-AGV systems. In Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Canada, 24–28 September 2017; pp. 1083–1088.

14. Hönig, W.; Kiesel, S.; Tinka, A.; Durham, J.W.; Ayanian, N. Conflict-Based Search with Optimal Task Assignment. In Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Stockholm, Sweden, 10–15 July 2018; pp. 757–765.
15. Ma, H.; Koenig, S. Optimal Target Assignment and Path Finding for Teams of Agents. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), Singapore, 9–13 May 2016; pp. 1144–1152.
16. Turpin, M.; Michael, N.; Kumar, V. CAPT: Concurrent assignment and planning of trajectories for multiple robots. *Int. J. Robot. Res.* **2014**, *33*, 98–112.
17. Ayanian, N.; Kumar, V. Decentralized Feedback Controllers for Multiagent Teams in Environments With Obstacles. *IEEE Trans. Robot.* **2010**, *26*, 878–887.
18. Panagou, D.; Turpin, M.; Kumar, V. Decentralized goal assignment and trajectory generation in multi-robot networks: A multiple Lyapunov functions approach. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 6757–6762.
19. Bandyopadhyay, S.; Chung, S.; Hadaegh, F.Y. Probabilistic swarm guidance using optimal transport. In Proceedings of the 2014 IEEE Conference on Control Applications (CCA), Juan Les Antibes, France, 8–10 October 2014; pp. 498–505.
20. Dias, M.B.; Stentz, A. Opportunistic optimization for market-based multirobot control. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, Lausanne, Switzerland, 30 September–4 October 2002; pp. 2714–2720.
21. Castello, E.; Yamamoto, T.; Libera, F.D.; Liu, W.; Winfield, A.F.T.; Nakamura, Y.; Ishiguro, H. Adaptive foraging for simulated and real robotic swarms: the dynamical response threshold approach. *Swarm Intell.* **2016**, *10*, 1–31.
22. Napp, N.; Klavins, E. A compositional framework for programming stochastically interacting robots. *Int. J. Robot. Res.* **2011**, *30*, 713–729.
23. Theraulaz, G.; Bonabeau, E.; Deneubourg, J.N. Response threshold reinforcements and division of labour in insect societies. *Proc. R. Soc. London Ser. B Biol. Sci.* **1998**, *265*, 327–332.
24. Beshers, S.N.; Fewell, J.H. Models of Division of Labor in Social Insects. *Annu. Rev. Entomol.* **2001**, *46*, 413–440.
25. Miletitch, R.; Dorigo, M.; Trianni, V. Balancing exploitation of renewable resources by a robot swarm. *Swarm Intell.* **2018**, *86*, 307–326.
26. Stroeymeyt, N.; Robinson, E.J.H.; Hogan, P.M.; Marshall, J.A.R.; Giurfa, M.; Franks, N.R. Experience-dependent flexibility in collective decision making by house-hunting ants. *Behav. Ecol.* **2011**, *22*, 535–542.
27. Franks, N.R.; Richardson, T.O.; Stroeymeyt, N.; Kirby, R.W.; Amos, W.M.D.; Hogan, P.M.; Marshall, J.A.R.; Schlegel, T. Speed-cohesion trade-offs in collective decision making in ants and the concept of precision in animal behaviour. *Anim. Behav.* **2013**, *85*, 1233–1244.
28. Reina, A.; Marshall, J.; Trianni, V.; Bose, T. Model of the best-of-N nest-site selection process in honeybees. *Phys. Rev. E* **2017**, *95*, 052411.
29. Marshall, J.A.R.; Bogacz, R.; Dornhaus, A.; Planqué, R.; Kovacs, T.; Franks, N.R. On optimal decision-making in brains and social insect colonies. *J. R. Soc. Interface* **2009**, *6*, 1065–1074.
30. Reina, A.; Bose, T.; Trianni, V.; Marshall, J.A.R. Psychophysical Laws and the Superorganism. *Sci. Rep.* **2018**, *8*, 4387–4388.
31. Reina, A.; Bose, T.; Trianni, V.; Marshall, J.A.R. Effects of Spatiality on Value-Sensitive Decisions Made by Robot Swarms. In *Distributed Autonomous Robotic Systems (DARS)*; Springer: Berlin/Heidelberg, Germany, 2018; pp. 461–473.
32. Fleming, C.; Adams, J.A. Recruitment-Based Robotic Colony Allocation. In *Distributed Autonomous Robotic Systems (DARS)*; Springer: Berlin/Heidelberg, Germany, 2019; pp. 79–94.
33. Caleffi, M.; Trianni, V.; Cacciapuoti, A.S. Self-Organizing Strategy Design for Heterogeneous Coexistence in the Sub-6 GHz. *IEEE Trans. Wirel. Commun.* **2018**, *17*, 7128–7143.
34. De Wilde, B.; Ter Mors, A.W.; Witteveen, C. Push and rotate: a complete multi-agent pathfinding algorithm. *J. Artif. Intell. Res.* **2014**, *51*, 443–492.
35. Luna, R.; Bekris, K.E. Network-guided multi-robot path planning in discrete representations. In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 18–22 October 2010; pp. 4596–4602.
36. Gerkey, B.P.; Matarić, M.J. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. J. Robot. Res.* **2004**, *23*, 939–954.
37. Khamis, A.; Hussein, A.; Elmogy, A. Multi-robot task allocation: A review of the state-of-the-art. *Coop. Robot. Sens. Netw.* **2015**, *2015*, 31–51.
38. Faria, D.B. *Modeling Signal Attenuation in IEEE 802.11 Wireless LANs*; Technical Report TR-KP06-0118; Stanford University: Stanford, CA, USA, 2005; Volume 1.
39. Beuran, R.; Nakata, J.; Okada, T.; Nguyen, L.T.; Tan, Y.; Shinoda, Y. A Multi-Purpose Wireless Network Emulator: QOMET. In Proceedings of the 22nd International Conference on Advanced Information Networking and Applications-Workshops (AINA Workshops 2008), Gino-wan, Japan, 25–28 March 2008; pp. 223–228.
40. Wang, S.; Krishnamachari, B.; Ayanian, N. The optimism principle: A unified framework for optimal robotic network deployment in an unknown obstructed environment. In Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 28 September–2 October 2015; pp. 2578–2584.
41. Dimidov, C.; Oriolo, G.; Trianni, V. Random Walks in Swarm Robotics: An Experiment with Kilobots. In Proceedings of the International Conference on Swarm Intelligence (ANTS), Brussels, Belgium, 7–9 September 2016; Volume 9882, pp. 185–196.

42. Phillips, M.; Likhachev, M. SIPP: Safe interval path planning for dynamic environments. In Proceedings of the 2011 IEEE International Conference on Robotics and Automation, Shanghai, China, 9–13 May 2011; pp. 5628–5635.
43. Ma, H.; Li, J.; Kumar, T.K.S.; Koenig, S. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS), São Paulo, Brazil, 8–12 May 2017; pp. 837–845.
44. Smith, R.G. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Trans. Comput.* **1980**, *29*, 1104–1113.
45. Sandholm, T. An implementation of the contract net protocol based on marginal cost calculations. In Proceedings of the AAAI, Washington, DC, USA, 11–15 July 1993; Volume 93, pp. 256–262.