Autonomous Robots
https://doi.org/10.1007/s10514-018-09820-5

Dynamic task allocation in an uncertain environment with heterogeneous multi-agents

Hebah ElGibreen[1] · Kamal Youcef-Toumi[2]

Received: 10 November 2017 / Accepted: 20 December 2018

Abstract

Dynamic task allocation (DTA) is a key feature in collaborative robotics. It affects organizations' profits and allows agents to perform more tasks when efficiently designed. Although some work has been done on DTA, allocating tasks dynamically in an uncertain environment between heterogeneous multi-agents has rarely been investigated. The solutions proposed so far have inefficiently managed uncertainty, and none of them has utilized the semantics of heterogeneous agents' capabilities. Studies measuring the performance of these techniques on real robots are also scarce. Therefore, this paper proposes an online DTA method, which introduces new functionalities that can be applied in a real environment. In particular, an uncertain incremental cost function is developed with a distributed semantic negotiation strategy that reflects heterogeneous capabilities without needing to communicate them. The proposed method is tested in a dynamic environment and experiments on heterogeneous real/virtual robots are conducted with different numbers of agents. Different statistical and visualization tools are used to analyze the results, including bar graphs for the waiting time metrics, histograms for the waiting time frequency, scatter plots for the result distribution and variance, and critical difference diagrams for ANOVA–Tukey results. The results indicate that the proposed DTA balances allocation quality and reliability, allowing the agents to serve targets equally without neglecting certain targets at the expense of the total performance. Evidently, updating the cost incrementally allows agents to update their allocation and choose better routes to finish the task earlier. Understanding the capability also gives priority to the capable agents that complete the task faster.

Keywords Dynamic task allocation · Heterogeneous multi-agent · Ontology · Uncertainty theory · Robotics

1 Introduction

In dynamic task allocation (DTA), tasks change during the lifetime of an agent. Subtasks can be introduced and removed while agents collaboratively perform a certain task. In a distributed collaborative environment, agents divide a task into subtasks and assign themselves to complete the job. Hence, the main requirement of any DTA algorithm is to correctly select agents to optimize the task's execution.

A DTA problem can be represented as a graph $G = \{V, E, \xi\}$, where $V = \{1,\ldots,n\} \in R$ are the vertices that represent the tasks, $E = \{e_{i,j} \mid i,j \in V\} \in R^2$ are the edges that encode connections between tasks, and $\xi = \{\xi_{i,j} \mid i,j \in V\} \in R$ are the set of cost values for edge $e_{i,j}$. In a distributed multi-agent environment, agents coordinate to allocate themselves to different vertices to collaboratively perform the tasks at each node. Agents are defined as $G = \{A, C\}$, where $A = \{1,\ldots,m\} \in R$ is the set of agent's ID and $C = \{c_{m,n} \mid m \in A \; \& \; n \in V\} \in R$ is the set of cost of allocating agent m to task n (Farinelli et al. 2017). The ultimate goal in DTA is to provide the agents with a mechanism to collaboratively select the least expensive subtasks and find the optimal path to execute them, which maximizes the total performance. Hence, the optimal solution is to minimize the total assignment cost (Eq. 1), where x_{mn} is a flag variable with zero value if agent m is not assigned to task n and a value of one if it is assigned, and c_{mn} is the allocation cost.

✉ Hebah ElGibreen
hjibreen@ksu.edu.sa

Kamal Youcef-Toumi
youcef@mit.edu

1 Information Technology Department, King Saud University, Riyadh 11415, Saudi Arabia

2 Mechanical Engineering Department, Massachusetts Institute of Technology (MIT), Cambridge, MA 02139, USA

$$\text{Min} \sum_{m \in A} \sum_{n \in V} x_{mn} c_{mn} \quad (1)$$

In general, task allocation is an NP-hard problem (Parker 2013), and introducing dynamics into the allocation makes the problem even harder. In a real environment, not only the task, but also the environment, is dynamic, which increases the computational complexity and uncertainty (Liu and Shell 2012a, b). In real-life environments, dynamics are always introduced owing to natural reasons. For example, agents in a rescue mission could have a predefined map of the environment, but the environment could change for natural reasons, such as if certain obstacles are demolished and new obstacles are found, depending on the type of disaster. Moreover, for the same reasons, tasks are also dynamic in real-life situations, because it is difficult to predefine all possible situations and assign agents to work on these tasks only. Returning to the example of a rescue mission, agents collaborate to patrol several areas and find survivors, whereby each area is considered a sub-task. In real situations, new areas are discovered while patrolling and, hence, new sub-tasks need to be assigned. It is also possible that certain areas are blocked and, hence, temporarily removed from the allocation until further information is found. The same concept also applies to factories, where agents collaborate to perform tasks such as boxing products or assembling devices.

The uncertainty caused by the dynamics introduces inconsistencies, whereby changes may cause the assignment plan to be invalid at some point, and thus it must be updated continuously. In general, owing to the changes occurring in dynamic environments, the cost of performing a task is uncertain and must be updated depending on the latest change. For example, in a rescue mission, the distance from an agent to the target area could be different from one time to another, depending on the current obstacles found in the environment. In addition, agents' characteristics and components can also increase the uncertainty when measuring the performance of an agent to allocate it to the most suitable task. This includes agents' reliability in term of localization, perception, planning, and communication delay. Another challenge reported in the literature is the reliability of DTA in dynamic and changing environments (Zhang et al. 2015). Current solutions require building a prediction model that is function-dependent and can become obsolete in time. Other solutions used heuristics to introduce flexibility, but this affected their efficiency and consistency.

An adequate task assignment, in which agents can handle as many tasks as possible in a timely manner, is a key feature to maximize any organization's profits. To reduce the computation and communication overhead, a decentralized multi-agent structure can be applied so that each agent can work independently in parallel to utilize the resources as efficiently as possible. However, this decentralization creates another problem in terms of coordination, especially if the agents are heterogeneous (Farinelli et al. 2017). Reducing the communication cost causes agents to lose important information that needs to be shared. Heterogeneous agents do not have common encoding, which makes it difficult for them to understand each other (Ure et al. 2015). Some solutions attempt to isolate agents from each other, which is inapplicable in a collaborative environment. Other attempts have tried to unify agent models, which is not an efficient solution, especially if scalability is important. Hence, allocating tasks is not an easy problem, and the challenges introduced by the environment and agents must be reflected in the selection decision.

This paper contributes to the literature of DTA by proposing a new distributed and online DTA method. The new method is called a Semantic-based Allocation with an Uncertain and Dynamic Environment (SAUDE). A novel negotiation strategy based on semantic ontology is introduced to solve the problem of heterogeneous multi-agent negotiation. This strategy uses an assessment matrix that allows agents to reason based on their physical capabilities, in addition to the current and new workload. Each agent only needs to understand itself and then coordinate with the other agents to choose the most suitable task. To deal with the problem of uncertainty, an incremental cost function is also adopted in SAUDE. This function is based on uncertainty theory (Liu 2015), and has been developed to add flexibility while preserving consistent planning. SAUDE is fully distributed and online, allowing agents to deal with dynamic tasks and continuously plan their next moves without setting a predefined plan. This function allows agents to consider new changes from the current observation while avoiding memory overload and excessive computation.

When a new task is available and as the agents are performing their own tasks, SAUDE allows each agent to determine online the next subtask to do. Each agent elects the best subtask that it can do based on its ontology. Then, it assesses its negotiation matrix to determine if there is another agent that is more capable. If another agent is found, the best agent allocates itself to execute the task. Then, the allocated agent incrementally updates the task cost based on the last observation. The updated cost is attached to the acknowledgement message that announces the task is complete. This way, all agents update their information without the need to add extra communication that could overload the network. To reduce the time, these steps are done in parallel for all agents, and the allocation is determined while the task is being executed.

This study also contributes to the literature by conducting several experiments on real and virtual robots. In these experiments, robots with positive and negative capabilities (velocity, acceleration, mass) collaborate to patrol areas and protect these areas from intruders. Dynamic obstacles and walls are introduced to change the environment from the original map given to the agents. The experiments are divided into three main parts: virtual, real, and real with virtual robots. In the virtual experiments, the performance is analyzed in a simulation with different numbers of agents that have similar hardware definitions but different physical capabilities. In the real experiments, the performance is analyzed using real robots called Turtlebots. These robots have similar hardware components but different physical capabilities, such as speed and acceleration. In the third set of experiments, however, both virtual and real robots collaborate to perform the task. These experiments measure the performance when the agents are heterogeneous in terms of not only their capabilities, but also their hardware, where the virtual robots use lasers to collect their observations and the real robots use cameras.

In all the experiments, each trial is repeated three times with every method to accurately measure the significance between the algorithms. This is because there are many variations introduced by the real and dynamic environment that could affect the performance of one experiment more than another. Measuring the performance over multiple trials can uncover common features in all tests and accurately analyze how well SAUDE is behaving compared to the other methods. During the experiments, several scenarios are considered to study the effect on a different number of agents. The performance of SAUDE is also compared with three of the latest distributed and online DTA methods (DTAG, DTAS, and DTAP). The target waiting time is used as a performance metric to determine how fast a task is completed. The results are analyzed using different visualization and statistical measures. In particular, the average, standard deviation, and maximum waiting time are visualized using a bar graph. The frequency of the waiting time throughout the experiment is visualized using a histogram. The distribution of the results is represented in a temporal scatter plot that shows how frequently and for how long the agents visited the targets during the experiments. Finally, each trial is repeated three times, and an analysis of variance (ANOVA) statistical test with a Tukey pairwise test is applied to the results to measure the significance between the tested methods.

From the results, it was seen that updating the cost incrementally changed the agents' allocation depending on the current environment and reduced the time of allocation. Allowing agents to rank themselves based on the semantics of their physical capabilities significantly improved their coordination and resulted in better scaling of their allocation with the increased number of agents. When testing SAUDE on agents with different physical capabilities, it was found that unlike the other tested methods, SAUDE can perform well when there is a small number of agents and also scales better with an increasing number of agents. This consistency was possible because of the proposed negotiation strategy that allowed agents to utilize their resources and coordinate depending on the current circumstances. Moreover, when applying SAUDE to agents with different physical capabilities and hardware components, its improvement was even more significant. When real and virtual agents worked together to patrol the given map, the variance of the targets' waiting time was very low, indicating that SAUDE allows the agents to serve all targets equally fast. When introducing dynamics into the environment and changing the obstacles, SAUDE adapted quickly and allocated better agents to targets that had become hard to reach. This adaptation was possible because of the incremental update of the cost, whereby the cost of new roads was reflected in the allocation decision.

The remainder of this paper is organized as follows: First, the background needed to understand the proposed approach is described. Then, the literature on DTA is discussed to identify the current gaps and open research areas. Next, the proposed method is presented, and its details are explained. After that, the experimental details are analyzed with the results. Finally, the paper is concluded, and future work is proposed.

2 Background

In this section, the important information and definitions needed to understand the proposed method are presented.

2.1 Multi-agent dynamic task allocation

In general, a multi-agent system consists of two or more agents collaborating to perform a certain task. An agent structure is illustrated in Fig. 1, which shows that sensors and actuators are used to collect observations and apply decisions in the environment. From the collected observations, an agent can plan its next move using a path planning method and avoid collisions using its plan. In general, when agents collaborate to perform a task, the task is divided into a number of subtasks that are assigned to different agents. While an agent plans its move to perform its current subtask, it can also decide on the next one after finishing the current plan. The selected subtask is negotiated with the other agents to collaborate and improve the efficiency of their work. Collaboration in task allocation allows agents to avoid redundancy and perform the best subtask with respect to the current circumstances. Hence, efficient task allocation can help in finishing the entire task faster, which is important to increase the profits of any organization.

Introducing a multi-agent structure can improve the task allocation in general (Khamis et al. 2015). The benefits include improving the reliability of the system, reducing the task completion time, and increasing the allocation scalability over complex tasks. However, tasks are not always known in advance, and dynamics in allocation should also

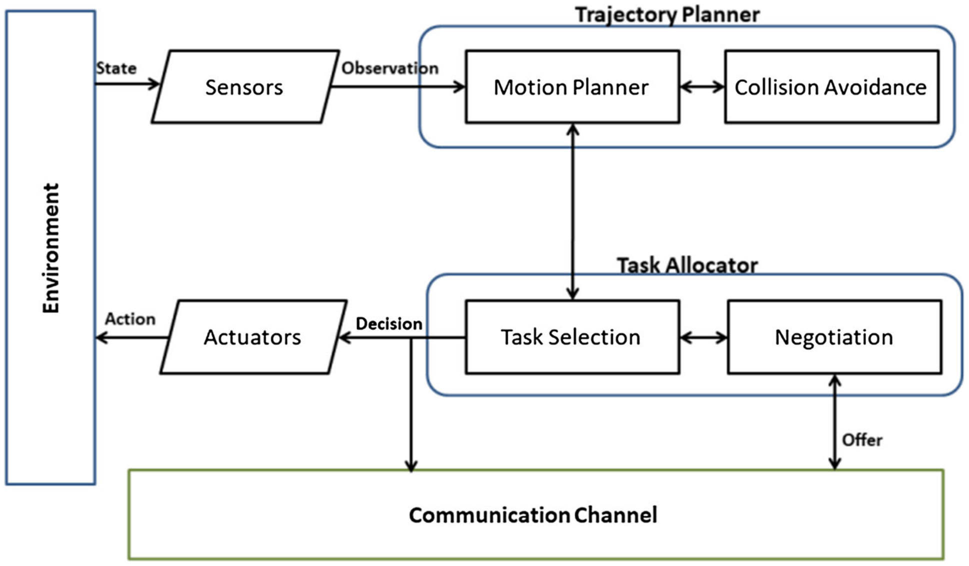


Fig. 1 General agent architecture

be considered. These dynamics increase the complexity of coordination among different agents, regardless of whether they have a common model or not.

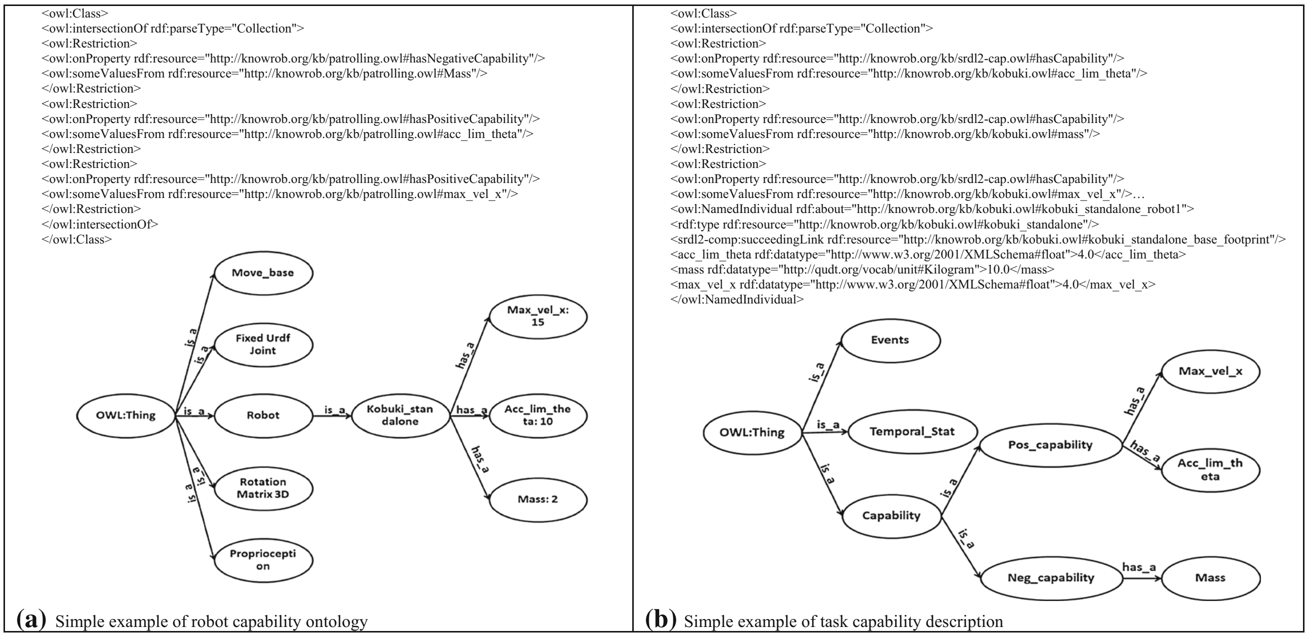
DTA is used when the tasks are not known in advance or when they change while the agents are collaborating. This means that the agents cannot come up with a predefined plan and commit to a specific task (Wei et al. 2016). This dynamic introduces difficulties for the agents, and they must continuously coordinate and adapt to changes that happen during their collaboration. Hence, the structure discussed in Fig. 1 is not suitable when dealing with the complexity of this problem, and further improvements to the structure are required. In particular, the agents' heterogeneity, uncertainty, and environmental dynamics must be reflected in the agents' structure to efficiently allocate the tasks to the most appropriate agent. Although the problem of DTA can be managed using online allocation as agents are completing their tasks (Farinelli et al. 2017), the current online methods are not suitable to handle the complexity introduced by real robots and dynamic environments.

2.2 Ontology

An ontology is a formal representation of knowledge that classifies the knowledge into concepts and connects it to represent a relationship (Hsieh et al. 2014). An ontology must contain at least three elements: the domain concept (classes), facts about the real world (instances), and the relationship between concepts. Hence, each concept is described by its attributes, values, and relationship. The language used to build ontologies is known as the web ontology language (OWL). It is used to define the vocabulary that describes a certain domain (Hristoskova et al. 2013). To visualize an ontology, the model must contain two parts: conceptualization and specification (Obitko and Marik 2002). The conceptualization shows the classes of the representation, and the specification is a formal description of these classes. This vocabulary can be used as a knowledge dictionary of a certain topic.

Because ontologies can be used to describe the agents' knowledge in a formal representation, one example of building this knowledge representation is using the unified robot description format (URDF) and physical models that represent their capabilities. One example of such a representation is illustrated in Fig. 2. In this example, the OWL file is represented in XML format, which can be visualized as a connected graph with classes (concepts) defining the capabilities and specifications defining the values of these capabilities. The connections among the concepts show the hierarchy of the physical capabilities and their relationships.

The main advantages of using ontologies is that they facilitate knowledge reuse, sharing, and communication (Bimba et al. 2016). They allow agents to understand the semantics of their capabilities and the effect of their values. Using URDF to build an ontology avoids the need for humans to validate this ontology. Agents can autonomously build and validate their ontology using their manufactured model. However, relating the capability to the task itself requires a predefined task description. One way to do so is by using the programming code of the task to build the task's description automatically. Another way is to use autonomous ontology tools that collect related task descriptions from the web. Finally, one of the mostly widely used methods to build ontologies is to use experts to validate the task description before using it. The task description can then be used by the agents to relate their capabilities and define what is a positive or negative capability with respect to the task.



```
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Restriction>
<owl:onProperty rdf:resource="http://knowrob.org/kb/patrolling.owl#hasNegativeCapability"/>
<owl:someValuesFrom rdf:resource="http://knowrob.org/kb/patrolling.owl#Mass"/>
</owl:Restriction>
<owl:Restriction>
<owl:onProperty rdf:resource="http://knowrob.org/kb/patrolling.owl#hasPositiveCapability"/>
<owl:someValuesFrom rdf:resource="http://knowrob.org/kb/patrolling.owl#acc_lim_theta"/>
</owl:Restriction>
<owl:Restriction>
<owl:onProperty rdf:resource="http://knowrob.org/kb/patrolling.owl#hasPositiveCapability"/>
<owl:someValuesFrom rdf:resource="http://knowrob.org/kb/patrolling.owl#max_vel_x"/>
</owl:Restriction>
</owl:intersectionOf>
</owl:Class>
```

(a) Simple example of robot capability ontology

```
<owl:Class>
<owl:intersectionOf rdf:parseType="Collection">
<owl:Restriction>
<owl:onProperty rdf:resource="http://knowrob.org/kb/srdl2-cap.owl#hasCapability"/>
<owl:someValuesFrom rdf:resource="http://knowrob.org/kb/kobuki.owl#acc_lim_theta"/>
</owl:Restriction>
<owl:Restriction>
<owl:onProperty rdf:resource="http://knowrob.org/kb/srdl2-cap.owl#hasCapability"/>
<owl:someValuesFrom rdf:resource="http://knowrob.org/kb/kobuki.owl#mass"/>
</owl:Restriction>
<owl:Restriction>
<owl:onProperty rdf:resource="http://knowrob.org/kb/srdl2-cap.owl#hasCapability"/>
<owl:someValuesFrom rdf:resource="http://knowrob.org/kb/kobuki.owl#max_vel_x"/>…
<owl:NamedIndividual rdf:about="http://knowrob.org/kb/kobuki.owl#kobuki_standalone_robot1">
<rdf:type rdf:resource="http://knowrob.org/kb/kobuki.owl#kobuki_standalone"/>
<srdl2-comp:succeedingLink rdf:resource="http://knowrob.org/kb/kobuki.owl#kobuki_standalone_base_footprint"/>
<acc_lim_theta rdf:datatype="http://www.w3.org/2001/XMLSchema#float">4.0</acc_lim_theta>
<mass rdf:datatype="http://qudt.org/vocab/unit#Kilogram">10.0</mass>
<max_vel_x rdf:datatype="http://www.w3.org/2001/XMLSchema#float">4.0</max_vel_x>
</owl:NamedIndividual>
```

(b) Simple example of task capability description

Fig. 2 Ontology sample: **a** sample of an agent ontology collected from its URDF and visualized as graph beneath it; **b** sample of an OWL file for patrolling task and its graph visualization beneath it

2.3 Uncertainty theory

When allocating tasks in a real environment, uncertainty could affect the quality of the agents and, hence, how fast they can perform a task. This uncertainty is introduced owing to changes in the environment and noise caused by the agents' sensors, actuators, motors, or communication failures. Uncertainty affects the agents' utilities and performance, which increase the importance of considering its effects in the task allocation. Most of the classical methods used to coordinate among agents to optimize the overall performance do not consider such uncertainty or try to resolve it through the mean or variance of the cost (Liu and Shell 2012a, b). However, when the task cost is not deterministic, these solutions are not efficient enough and information regarding the uncertainty must consider the distribution of the information instead of reducing it to a single value.

Some attempts have been made to consider the cost distribution in task allocation. One example is the Hungarian algorithm proposed by Liu and Shell (2012a, b). However, their algorithm was used to divide a map among agents and allocate them statically. The allocation was not dynamic and required a pre-processing estimate before the allocation, which increased the commutation complexity. Thus, new branches of mathematics must be investigated to reflect the uncertainty in dynamic allocation. In particular, uncertainty theory can be used to describe the nondeterministic cost of tasks during the task allocation.

An uncertainty space (B. Liu 2015) is represented as (Γ, L, M), where L is a σ-algebra over a non-empty set Γ and M{Λ} is an uncertainty measure that indicates the degree of belief that Λ element will occur. In this space, an uncertain variable is a measurable function ξ that results in a real number from the uncertainty values. For ξ ~ L(a, b), the uncertainty distribution is measured by M{ξ ≤ x} for any real number x (Eq. 2).

$$\mathrm{M}\{\xi \le x\} = \Phi(x) = \begin{cases} \mathbf{0}; & \boldsymbol{if\ x \le a} \\ \frac{(\boldsymbol{x-a})}{(\boldsymbol{b-a})}; & \boldsymbol{if\ a} < x < b \\ \mathbf{1}; & \boldsymbol{if\ x \ge b} \end{cases} \tag{2}$$

Discovering the uncertainty distribution with respect to an unknown real value x is not possible in a dynamic environment. However, using the operational property of the uncertainty theory, the distribution can be inferred from the inverse uncertainty distribution $\Phi^{-1}(\alpha)$ of the uncertain variable ξ that has a regular uncertainty distribution $\Phi(x)$, where α∈[0–1] is a confidence constant. For ξ ~ L(a, b), the inverse uncertainty distribution is defined by (Eq. 3).

$$\Phi^{-1}(\alpha) = (1-\alpha)\mathrm{a} + \alpha\mathrm{b} \tag{3}$$

As described in the introduction, a DTA problem is presented as a graph, where tasks are connected by edges. Agents coordinate to allocate themselves to different vertices to collaboratively perform their tasks. The vertices' cost is nondeterministic and highly affected by uncertainty. Hence, finding the optimal path between two tasks is very important. Recently, the inverse uncertainty distribution was used to discover the shortest path in uncertain networks. Gao (2011) tried to discover the deterministic cost that is set based on an uncertain distance. The solution used the uncertainty theory to discover the distribution of the shortest path length in an uncertain network. However, the approach started by constructing a deterministic network to compute the shortest path from all possible confidence values. This work was further improved (Sheng and Gao 2016) to deal with dynamics in the environment by introducing random distance. Nevertheless, both solutions enumerated all possible paths between two nodes, and the latest one required knowledge of the probability distribution of the random variable in advance. This increased the computational complexity and reduced the algorithm's scalability. It was found that even though the uncertainty theory can introduce flexibility and consistency into the shortest path problem when dealing with uncertainty, new strategies need to be adopted to scale it in real environments.

3 Related work

Because of the importance of DTA, several attempts have been made to solve this problem. The proposed solutions can be divided into two types: exact solutions and heuristics (Liu et al. 2015). For the heuristic types, evolution-based methods are usually applied, especially ant colony (AC) algorithms. An AC-based algorithm (Liu et al. 2015) was proposed to use a greedy heuristic to assign dynamic tasks. The basic idea is to consider each task assignment given to an agent as a bee and start with an initial population using a greedy heuristic. This solution is then improved using a greedy local search, in which a fitness value is calculated to be remembered by the bees. This solution is scalable over large problems in terms of speed, but its accuracy is still questionable because it can be stuck in local minima. Another attempt to use AC was proposed (Su et al. 2017), where each agent is treated as an ant that communicates its current observation with others before it makes a decision. AC methods were more acceptable in DTA than other evolutionary algorithms, such as genetic algorithms (GAs) and particle swarm optimization, but the accuracy of the allocation is always affected by the heuristics, and it is possible to select a local optimal solution instead of the global optimal.

An improved particle swarm optimization (IPSO), ant colony algorithm (ACA), and GA (Huang and Zhuo 2017), were proposed for task assignment in unmanned combat aerial vehicles (UCAVs). The task assignment was formed based on the UCAV flight characteristics over the battlefield, and the three evolutionary algorithms were applied for planning and task assignment. Over the battlefield, agents select the best algorithm for planning depending on the current environment. Based on the simulation results, these algorithms traded between accuracy and speed and could not guarantee the best when performing both. This problem is a general open research area in evolutionary algorithms.

When it comes to exact solutions, methods under this type of DTA can converge better. One of the most widely known approaches used in DTA problems is the market-based approach, and auctioning in particular. The basic idea is that each agent bids on the task, and the winner is allocated to this task. It is possible to have a centralized station as an auctioneer or to consider each agent as an auctioneer to introduce distribution. Usually, one bidder wins the bid, and thus tasks are only assigned to one agent at a time, but there were some attempts to relax this condition to overcome possible failures. A consensus-based auction algorithm (CBAA) and consensus-based bundle algorithm (CBBA) (Choi et al. 2009) were proposed based on the concept of auctions and bundles. These algorithms use consensus routines for local communication and market-based strategies as the decision mechanism. First, the auction approach is used to allocate a task to an agent, and then, a conflict-based protocol is applied to resolve any conflicts with other agents. Although the performance of this solution guaranteed the convergence to a conflict-free assignment, it only guaranteed 50% of the optimality and assumed task consistency. Heterogeneity and environmental uncertainty were neglected in this work.

A new DTA strategy (García et al. 2013) was proposed for a heterogeneous multi-agent system. The main objective of this study was to address the scalability and dynamicity problem found in auction-based algorithms. This was done by adding a behavior-based architecture layer over the allocation algorithm to reuse past decisions for better coordination. It was found that a simple auction-based allocation can offer a linear computation of the cost when increasing the number of agents. This confirmed its scalability, but the performance was still an issue, especially in a multi-agent environment. Liu and Shell (2012a, b) proposed another scalable task assignment approach that works in a top-down manner. During the assignment process, partitioning and coarsening operations were applied to the utility matrix to allocate tasks. Centralized and distributed approaches were applied at different levels to increase the algorithm's scalability. Tasks were allocated either by using a previous plan or by refining the plan recursively. Although the allocation approach is scalable, it sacrificed the results' quality. The algorithm traded the solution efficiency for quality.

Another DTA approach was proposed (Pippin et al. 2013) for observing agents' collaboration in a patrolling task and observing the agents' behavior to detect poor performers and reassign the tasks to better ones. The approach starts by allocating each agent to an area and using a centralized monitor that measures how long each agent takes to finish its task. When a predefined threshold is reached and an agent did not finish the task, this task is reassigned to another agent. This work considered the possibility that agents could have difficulties in finishing their tasks, and it was tested with real robots. However, the proposed approach is not fully distributed owing to the centralized monitor used to find poor agents. Additionally, finding the best predefined threshold to trigger a poor agent is challenging and user-defined. The capabilities of the agents are not reflected and only the time of finishing the task is considered. Moreover, the environment and tasks are not dynamic. The proposed approach was applied to real robots without analyzing the performance results. They only showed how the robots changed their assignment without assessing the quality of the re-assignment.

Another improvement on the market-based approach was proposed (Wicke et al. 2015), in which a bounty hunter and bail bondsmen methods were used instead of auctioning. In this method, auctioneers are replaced with bondsman and bidders are replaced with bounty hunters. Agents compete to complete the task, and the one that finishes it first is rewarded. Uncompleted tasks will have a bounty value that increases over time to make it more desirable for agents to select. This algorithm allows more than one agent to do the same task, even though one of them will be aborted eventually. This wastes more resources, regardless of its capability to handle noise in the system.

A probabilistic task allocation strategy (Portugal and Rocha 2016) was proposed based on distributed intelligence and Bayesian decision rules. Using a reward-based learning technique, the agents decide their future moves. As in the case of SAUDE, the strategy supports heterogeneous agents by avoiding the use of predefined routes. However, their method does not reflect the differences in capabilities in the allocation decision. Hence, each agent will decide its patrolling path differently depending on its capability, but the allocation strategy is the same, regardless of the heterogeneity. The proposed strategy was tested with real robots, but it was found that the effect of adding more heterogeneous agents on the global performance was minor. This indicates that, indeed, their strategy did not consider the new capabilities of the newly added agents and, hence, the performance did not improve significantly.

The DTA problem was also studied in the search and retrieval domain (Wei et al. 2016), where each agent needed to collaborate to search for a target and bring it home. The solution was inspired by the market-based approach and introduced a prediction capability that allowed agents to predict what other agents would do without negotiation. Even though this prediction capability improved the speed, it affected the allocation accuracy. It traded the solution optimality for reduced completion time. It should be noted that the performance using real robots was not analyzed. The effect of using continuous versus punctual coordination in auction-based protocols was also studied (Lozenguez et al. 2016). In punctual coordination, tasks are distributed during the task's execution, whereas in continuous coordination, one task is assigned at a time, which is performed by one agent. These approaches were tested on simulated and real robots, showing that continuous coordination is more suitable when communication can be lost frequently, but punctual coordination is better suited when initial knowledge of the environment and task is introduced. However, it was found that the efficiency of the proposed approach affected the quality of the algorithm, especially in punctual coordination, because it requires exponential computation. It was stated that more work is needed to produce a more efficient allocation while improving the selection convergence for the optimal solution.

Another attempt was presented by Farinelli et al. (2017), who proposed two market-based DTA algorithms: greedy (DTAG) and sequential DTA (DTAP, DTAS). As in the case of SAUDE, these algorithms coordinate online and can be applied in a heterogeneous setup. The proposed algorithms are auction-based algorithms and select tasks based on their waiting time (idleness). In DTAG, the local instantaneous idleness is considered to make the decisions. In DTAS and DTAP, however, agents announce their desired tasks first and then collect bids from other agents to decide whether an agent can perform a task or leave it for another agent. The main difference between DTAP and DTAS is in the way the cost is computed during the bid collection. DTAP computes the cost between the central node of the current task and the new task. DTAS computes the path cost between current and new task in addition to the other tasks in the waiting list. Therefore, DTAS was not reported in the main paper but rather in the supplementary resources provided in the simulation tool website. The performance of DTAP and DTAG was reported using realistic simulations, and it was found that online coordination can critically improve performance.

DTAP, DTAS, and DTAG were implemented to support heterogonous agents by avoiding the use of predefined routes and rather computing the routes while performing the task. However, the differences among the agents' capabilities were not reflected in the allocation. The allocation methods proposed only considered the cost of performing a task without considering the agents' different characteristics. Moreover, even though there was an attempt to apply the algorithms to real robots, the performance was not collected or analyzed. The environment and costs were assumed to be predefined and fixed, and the agents' capabilities were neglected. It was confirmed by the authors that online DTA is still an open research area and needs improvement. Their solution is still lacking for complex operations and did not handle agents with different abilities.

To improve agents' coordination in a heterogeneous collaborative system, the differences among agents' capabilities must be considered. Iocchi et al. (2003) proposed a coordination approach in a heterogeneous multi-agent system.

The approach proposed broadcasting the utility function to communicate agents' capabilities before performing a task. These capabilities included several variables related to the new task, such as obstacles, trajectory, distance, and movement direction. Although heterogeneity among robots was considered, it was proposed that robots with similar capabilities could communicate with each other, indicating the unification of these capabilities. Moreover, the approach was tested empirically with the Azzurra Robot Soccer Team to confirm its effectiveness in the RoboCup competition. However, overcoming failure and robustness was still a concern that needed further investigation.

Wurm et al. (2013) also addressed the problem of heterogeneous agent coordination when agents must perform different actions using heuristics. Decisions were made based on the action cost of each agent using symbolic planning systems. Although this solution reduced the time needed for planning, the empirical results showed that non-optimal results were discovered more often while improving the solution. This caused a scalability issue whereby the complexity increased with the number of agents. Moreover, the semantics of these actions were not considered. Another attempt was reported (Drenjanac et al. 2015) where the authors used a shared knowledge interaction modeling framework for task allocation. This framework uses two ontologies: the resource description and coordination requirement for interaction. The resulting framework was used to evaluate the task allocation performance but not in the allocation process itself. Moreover, tasks were considered static and known in advance to enable building the model. Humans were also used to evaluate the model and within the allocation process. Hence, the allocation was not dynamic and guided by humans. Zhang et al. (2015) used the agents' capabilities to improve coordination between heterogeneous agents in DTA. However, only the number of capabilities was considered, regardless of the meaning of these capabilities or whether they were good or bad capabilities with respect to the task. A fuzzy-based algorithm was developed to optimize the assignment plan in the package loading–unloading task. The results were function-dependent, and the membership function had to be tuned by the user.

It can be concluded from the review of the literature that methods of the exact type have been proposed to find the optimal allocation given a certain objective. These methods tend to take time to find the solution. Finding the optimal solution is difficult, which causes them to find sub-optimal solutions instead (solutions near the optimal) and, thus, they do not scale well. Methods of the heuristic type, however, are faster and more effective, but these solutions tend to be affected by the heuristics, and it is possible to select a locally optimal solution (optimal with respect its neighbors only) instead of the global optimal solution. There is always a trade-off between the computation overhead and solution optimality.

When agents have different capabilities and are heterogeneous, the current solutions neglect the semantics of these capabilities with respect to the current task. Moreover, more work needs to be done with respect to DTA in real robots. Although the literature on DTA is available, analyses of real robots' performance are still limited. The dynamic and uncertain situations introduced by a real environment are neglected, and more studies need to be done in this area. Realistic experiments must be conducted, and DTA methods must be analyzed within this real environment. Solutions must be proposed to find the optimal allocation while scaling the method over a large and dynamic environment. The next section introduces an online allocation mechanism in a distributed multi-agent architecture that updates the plan using the uncertainty theory and semantic negotiation.

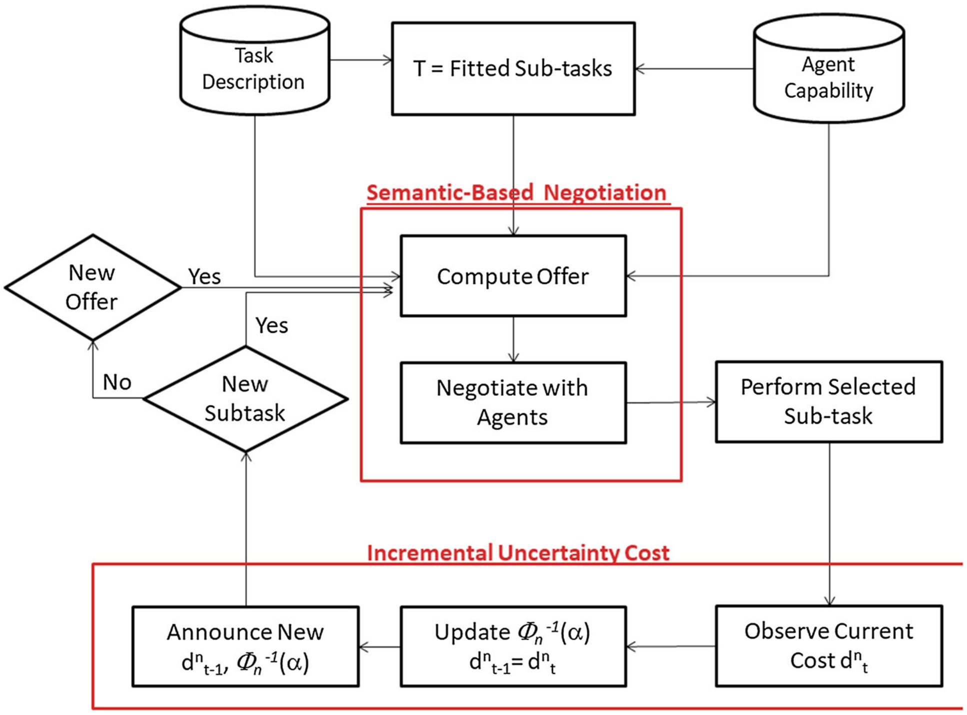
4 Semantic-based allocation with uncertain and dynamic environment: SAUDE

In this paper, DTA is considered when a task must be done by more than one agent. The task is divided into subtasks and agents collaborate to finish the whole task. For an agent to allocate itself to the most suitable subtask, considering its cost is insufficient. Understanding the semantics of agents' capabilities and knowing each agent's value is critical to improving the allocation performance. Nevertheless, when the agents are heterogeneous, understanding the semantics is very difficult, and unifying the agents' models is inapplicable in a dynamic environment. To optimize DTA in a dynamic environment and allow heterogeneous agents to collaboratively perform their tasks, a novel distributed and online DTA method is proposed in this section. This method is called Semantic-based Allocation with an Uncertain and Dynamic Environment (SAUDE), and is inspired by the negotiation capability introduced in the auctioning approach, where each agent converges its plan to the best allocation strategy.

Each agent starts by building its own ontology to represent the semantics of its physical capabilities. In SAUDE, agents build their ontology from their URDF and manufacturing model to represent their types of physical capabilities. The task description is then used to identify the agents' positive and negative capabilities, and the capabilities' values are taken from their URDF. In the past, several tools have been used to build an ontology offline and online. In particular, KnowRob tool (Tenorth 2011; UNIHB 2016) was used in SAUDE to build the ontology, and the knowledge was stored in OWL files to be processed using Protégé (Research 2017). Each agent can then use its own ontology to understand what types of positive and negative capabilities it has and what the agent's values are with respect to the current task.

As illustrated in Fig. 3, SAUDE allows each agent to select the appropriate task based on its capabilities. After negotiat-

Fig. 3 SAUDE method (one agent view)


Task Description
T = Fitted Sub-tasks
Agent Capability
Semantic-Based Negotiation
New Offer
Yes
Compute Offer
Yes
Negotiate with Agents
Perform Selected Sub-task
No
New Subtask
Incremental Uncertainty Cost
Announce New d^n_{t-1}, $\Phi_n^{-1}(\alpha)$
Update $\Phi_n^{-1}(\alpha)$ $d^n_{t-1} = d^n_t$
Observe Current Cost d^n_t

ing with the other agents, the selected task is performed by the most suitable agent. After that, each agent continuously updates and communicates the cost distribution based on its current observation. Therefore, the most important components of SAUDE can be divided into three parts: semantic negotiation, incremental update for the uncertain cost, and communication. To understand how the agents negotiate and update the uncertain cost, these components are explained in detail in the following sections.

4.1 Semantic negotiation

To avoid the need to communicate the capabilities while considering its semantics, each agent only needs to understand its own capabilities and compute its offer using a negotiation matrix that can be used during the negotiation. The negotiation strategy in SAUDE works as illustrated in Fig. 4. After building the ontology, each agent can access the values of its positive and negative capabilities with respect to the task. Positive capabilities are those that have a good impact on the task when increased, whereas negative capabilities are those that have a bad impact on the task when increased. For example, increasing the speed in a searching task is a good behavior for a robot to have (positive capability), whereas increasing the robot weight reduces the robot performance in searching tasks (negative capability). Robots can understand whether their capability value is positive or negative using the ontology built from their URDF and manufacturing model.

During the collaboration, when an agent selects a task to negotiate, this selection is announced to the other agents with the announcer offer. The other agents assess the task using the matrix defined in (Eq. 4), where $\psi_{n'}^{-1}(\alpha)$ and $\psi_{n}^{-1}(\alpha)$ are the costs of performing the new and current tasks, respectively. The cost distribution is incrementally updated to consider changes in the environment using the incremental uncertain cost function, which is explained in Sect. 4.2. $\varsigma\boldsymbol{p},\boldsymbol{m}$ and $\varsigma\boldsymbol{n},\boldsymbol{m}$ are the weighted sums of the positive (Eq. 5) and negative capabilities (Eq. 6). These weights give the percentage of how good and bad the agent is with respect to its own capabilities. A percentage scale is used instead of the true values, because the value ranges may be different from one agent to another. Scaling the capabilities allows the use of the weight percentage, regardless of the agents' heterogeneity. Note that these values are collected from the agent ontology built from its URDF, as explained at the beginning of Sect. 4.

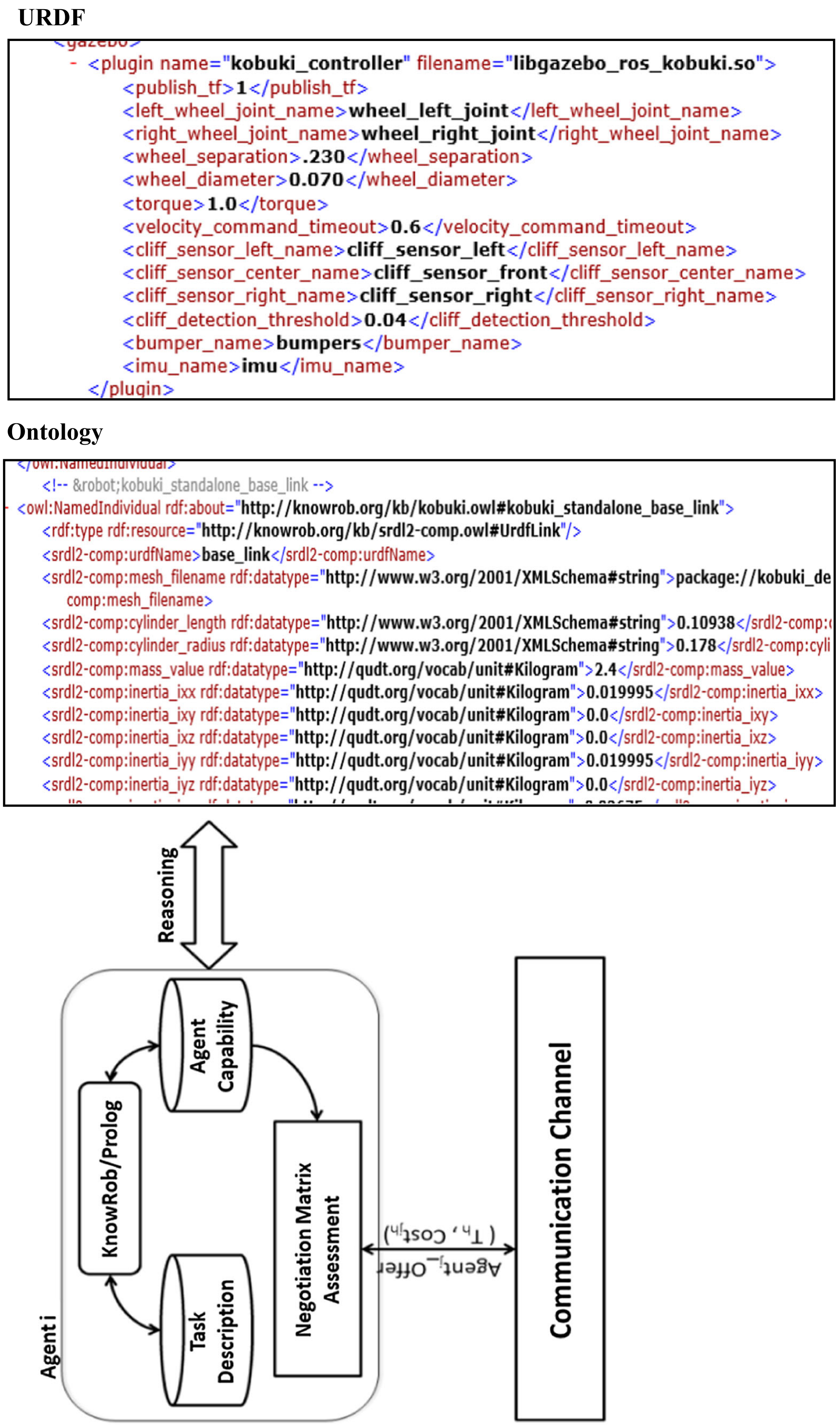
$$c_{mn'} = \frac{\varsigma\boldsymbol{n},\boldsymbol{m}}{\varsigma\mathbf{p},\mathbf{m}.\mathbf{Wm}} + \left[\psi_{n'}^{-1}(\alpha) + \psi_{n}^{-1}(\alpha)\right] \tag{4}$$

$$\varsigma\boldsymbol{p},\boldsymbol{m} = \frac{\sum_{\forall j \in pos.Capability} \overline{C_{mj}}}{\#\, of\ PosCapability} \tag{5}$$

$$\varsigma\boldsymbol{n},\boldsymbol{m} = \frac{\sum_{\forall j \in neg.Capability} \overline{C_{mj}}}{\#\, of\ NegCapability} \tag{6}$$

$$\mathbf{W_m} = \frac{No.\, of\ done\ tasks}{No.\, of\ tasks\ assigned} \tag{7}$$

$\mathbf{W_m}$ is the workload of agent m, which is computed using (Eq. 7) to consider how many tasks are waiting to be done



URDF

```
  <gazebo>
  - <plugin name="kobuki_controller" filename="libgazebo_ros_kobuki.so">
      <publish_tf>1</publish_tf>
      <left_wheel_joint_name>wheel_left_joint</left_wheel_joint_name>
      <right_wheel_joint_name>wheel_right_joint</right_wheel_joint_name>
      <wheel_separation>.230</wheel_separation>
      <wheel_diameter>0.070</wheel_diameter>
      <torque>1.0</torque>
      <velocity_command_timeout>0.6</velocity_command_timeout>
      <cliff_sensor_left_name>cliff_sensor_left</cliff_sensor_left_name>
      <cliff_sensor_center_name>cliff_sensor_front</cliff_sensor_center_name>
      <cliff_sensor_right_name>cliff_sensor_right</cliff_sensor_right_name>
      <cliff_detection_threshold>0.04</cliff_detection_threshold>
      <bumper_name>bumpers</bumper_name>
      <imu_name>imu</imu_name>
    </plugin>
```

Ontology

```
  </owl:NamedIndividual>
    <!-- &robot;kobuki_standalone_base_link -->
- <owl:NamedIndividual rdf:about="http://knowrob.org/kb/kobuki.owl#kobuki_standalone_base_link">
    <rdf:type rdf:resource="http://knowrob.org/kb/srdl2-comp.owl#UrdfLink"/>
    <srdl2-comp:urdfName>base_link</srdl2-comp:urdfName>
    <srdl2-comp:mesh_filename rdf:datatype="http://www.w3.org/2001/XMLSchema#string">package://kobuki_de
      comp:mesh_filename>
    <srdl2-comp:cylinder_length rdf:datatype="http://www.w3.org/2001/XMLSchema#string">0.10938</srdl2-comp:
    <srdl2-comp:cylinder_radius rdf:datatype="http://www.w3.org/2001/XMLSchema#string">0.178</srdl2-comp:cyli
    <srdl2-comp:mass_value rdf:datatype="http://qudt.org/vocab/unit#Kilogram">2.4</srdl2-comp:mass_value>
    <srdl2-comp:inertia_ixx rdf:datatype="http://qudt.org/vocab/unit#Kilogram">0.019995</srdl2-comp:inertia_ixx>
    <srdl2-comp:inertia_ixy rdf:datatype="http://qudt.org/vocab/unit#Kilogram">0.0</srdl2-comp:inertia_ixy>
    <srdl2-comp:inertia_ixz rdf:datatype="http://qudt.org/vocab/unit#Kilogram">0.0</srdl2-comp:inertia_ixz>
    <srdl2-comp:inertia_iyy rdf:datatype="http://qudt.org/vocab/unit#Kilogram">0.019995</srdl2-comp:inertia_iyy>
    <srdl2-comp:inertia_iyz rdf:datatype="http://qudt.org/vocab/unit#Kilogram">0.0</srdl2-comp:inertia_iyz>
```

Fig. 4 Semantic negotiation strategy

by the agent. Using this assessment matrix allows each agent to assess the task using its own positive ($\varsigma \boldsymbol{p}, \boldsymbol{m}$) and negative ($\varsigma \boldsymbol{n}, \boldsymbol{m}$) capabilities, the costs of performing the new ($\boldsymbol{\psi}_{\boldsymbol{n}'}^{-1}(\boldsymbol{\alpha})$) and current tasks ($\boldsymbol{\psi}_{\boldsymbol{n}}^{-1}(\boldsymbol{\alpha})$), as well as its current workload ($\mathbf{W_m}$). This way, it is possible to discover if there is a more capable agent that can perform the task while balancing the workload among all agents.

If an agent assesses itself and finds that it is better than the announcer, it sends a message with its offer to the announcer. After waiting for a predefined time, the announcer considers everyone's replies, sends an acknowledgement to the best-suited agent to take the task, and informs everyone else not to take it. However, if another agent finds itself better and did not get an acknowledgement from the announcer, it assumes that the acknowledgement was lost and allocates itself to the task. Hence, it is possible that because of uncertainty or possible communication failure, more than one agent temporary be allocated to the same task until further communication is acquired. However, because continuous coordination is used in SAUDE, lost information is exposed when another agent starts to coordinate its task. This allows agents to introduce flexibility in emergency situations while trying to preserve their consistency. Moreover, if the announcer obtains no answer, then the announcer allocates itself to the selected task. This strategy allows agents to take over tasks that were assigned to a poor agent if they can do a better job.

4.2 Incremental uncertain cost

During the negotiation process, agents need to assess the cost of performing the task in addition to using their assessment matrix. Unfortunately, the current literature uses the deterministic cost, which is inapplicable when the environment is dynamic. The cost distribution changes depending on the current obstacles, and such uncertainty must be reflected in the cost function. Therefore, it was decided to use the inverse uncertain cost distribution and incrementally update it based on the current observation.[1] In particular, the uncertainty theory is used to build this function, where the inverse distribution is used instead of exact values.

SAUDE computes the inverse uncertain distribution for the cost while updating it incrementally using the last observation only. This avoids the need for a predefined probability model or to store all the observations. To reduce the communication and computation costs, the distribution is defined as a linear distribution L(a, b), such that only the last image of the uncertain distribution is stored. To reduce the communication further, the cost information can be carried with an acknowledgement message when an agent finishes its job. To assign task n' to an agent that just finished task n, the uncertain cost between these two tasks can be predicted using the uncertainty theory, assuming the following for uncertain space = {V, E, ξ}, where V is the tasks, E is the edges, and ξ is the cost function of the edges such that:

- $\xi_{ij} \geq 0$
- $\xi_{ij} = 0$ iff $\xi_i = \xi_j$
- ξ_{ij} independent $\forall$ (i,j) $\in$ A
- There is one source and one target $\forall\ \xi_{ij}$

The optimal solution with respect to the uncertainty theory can be defined using (Eq. 8). The goal is to find the uncertain distribution and compute the shortest path with respect to the current time and $\xi_n \sim L(d^n_{t-1}, d^n_t)$. In SAUDE, to avoid excessive computation, only one confidence value (α) is used to compute the cost. A confidence value of 90% was chosen to compute the possible distribution of a distance that has a probability of 0.9. This has better flexibility than a confidence value of 100%, while preserving a high probability. Moreover, there is no need to compute all possible combinations of the confidence values as in the current shortest-path methods.

$$\text{Min} \sum_{m\in A} \sum\nolimits_{n\in V} x_{mn} \Phi(\alpha)^{-1}_{mn} \tag{8}$$

With respect to the shortest-path problem, let ξ_1, ξ_2,..., ξ_n be independent uncertain variables with a regular uncertainty distribution Φ_1(x), Φ_2(x),..., Φ_n(x) and the inverse uncertainty distribution for $\xi = f(\xi_1, \xi_2,.., \xi_n)$, where $f : \Re^n \rightarrow \Re$ is continuous and strictly increasing function defined by (Eq. 9). The results of this equation give an intuition of the predicted cost with a probability equal of α.

$$\psi^{-1}(\alpha) = f(\Phi_1^{-1}(\alpha), \Phi_2^{-1}(\alpha) \ldots, \Phi_n^{-1}(\alpha)) \tag{9}$$

While performing a task, agents observe the real cost of doing this task and use this value to update the inverse distribution with respect to the current time. As illustrated in Fig. 5, after completing a task, the agent incrementally updates the inverse distribution of the cost based on the last observation. The updated distribution is attached to the acknowledgement message that announces finishing the task. This way, all agents update their distributions without the need to add extra communication channels that could overload the network.

Later, the updated cost is used in the assessment matrix when agents need to negotiate selecting a certain task. Each agent continues to look for new tasks to perform and coordinate with other agents to perform all the tasks. These steps are done in parallel for all agents, and the allocation is determined while the tasks are being executed. Hence, the approach is considered fully distributed and online.

[1] In this context, the observation is the real cost observed by the robot while performing the task in its current environment.

Fig. 5 Incremental cost update

```
Input: current node n, last distance observation $d^{n}_{t-1}$, new distance observation $d^{n}_{t}$
Output: updated cost

1. Update the inverse uncertain distribution $\Phi_n^{-1}(\alpha)$ with respect to ($d^{n}_{t-1}$, $d^{n}_{t}$) using (Eq. 3).
2. Update the last distance observation ($d^{n}_{t-1} = d^{n}_{t}$)
3. Announce completing task n along with the updated distribution variables ($d^{n}_{t-1}$, $d^{n}_{t}$, $\Phi_n^{-1}(\alpha)$).
4. Using the Dijkstra algorithm, find the shortest path to the next task n'.
5. Negotiate the selected task and allocate it to the best agent.
6. Finish the new task and collect new distance observations ($d^{n'}_{t}$) at the arrival time.
7. Go to (1), assuming n = n' for current time t.
```

4.3 Communication

In order to negotiate and update the uncertain cost distribution, agents must coordinate with each other and exchange certain information. SAUDE is fully distributed, such that each agent locally keeps information about the environment and only needs to exchange offers and new observations with the other agents. As illustrated in Fig. 6, each agent starts by calling the initialization function [A] to collect the task information and initialize the task waiting time. Each agent computes the average rate of its positive and negative capabilities built previously from its own ontology, as described in Sect. 4.1. Then, each agent starts to allocate itself to the most suitable task and perform it accordingly.

While performing a task, two outcomes are possible: an agent either completes the task or is interrupted. When an agent is interrupted, function [C] is called, and the agent releases its resources and starts another task. This method avoids task preemption, whereby tasks are not interrupted unless another agent takes over the task. This occurs when an agent takes much more time than it should and another agent that is currently available is capable of finishing the task faster. This allows the system to avoid task starvation when an agent dies while performing its task. It also avoids letting two agents to perform the same task at the same time, which exploits the system resources.

When an agent completes its current task, function [B] is called to announce its completion and observation to the other agents. If the agent started the task without being interrupted in the previous task, the observed cost distribution is communicated to be updated as discussed in Sect. 4.2. However, if the agent started the task after being interrupted, this observation is not used. This is because the observed cost is affected by the interrupted task and, thus, it does not reflect the actual cost. This allows agents to only communicate accurate observations and avoid noisy data. The agent starts to select the following task and update its task list afterwards. Finally, the global waiting time is also communicated with a monitor node used to measure the system performance in the experiment, as discussed in Sect. 5.

In order for an agent to select the next task, using function [D], the most rewarding task is selected. A utility function is used to select the task, depending on the task performed. In Sect. 5.3, the utility function used in a patrolling task is explained in detail. After choosing the most rewarding task, the agent must negotiate with the other agents to ensure that it has the best offer. Hence, it computes its offer using (Eq. 4) and start its semantic negotiation as described in Sect. 4.1. Then, it announces the selected task with its offer, and the other agents compute their offers and send them back if they are better. The agent with the best offer (lowest cost) is allocated to the task. If the announcer is not allocated, then it negotiates the next-most rewarding task, until it is allocated to perform a task.

From the above discussion, it can be concluded that agents need to communicate in three cases, as described in function [E]. The first case is when a task is completed. In this case, the updated waiting time and observed cost distribution is shared to be stored locally. This is because the system is fully distributed, such that each agent must have a complete view of the system update. The second case is when an agent negotiates a selected task. The selected task is announced because the other agents' offers must be collected before the allocation. The last case is when an agent replies to a negotiation request and sends back its offer stating that it has better offer than the announcer.

Agents' communication could be interrupted or delayed owing to networking problems. The effect of this interruption is reduced in the proposed system, because it is fully distributed and performs continuous coordination. Note that there are more implementation details that consider communication delays to avoid deadlocks, but these were omitted from the pseudo code for simplicity. In general, the system was implemented to support possible delays in any communication and agents repeat their messages at least three times when they do not receive an answer. If an observation message is lost during communication, this information is

Fig. 6 Coordination pseudo code (one agent view)

```
A] Initialize() //Method called locally to initialize each agent's parameters
    My_Tasks = All_Tasks()          //Get the tasks and their initial costs
    Initialize_Global_waiting(My_Tasks, BIG_NUMBER)  //Initialize tasks as waiting forever
    Initialize_Global_Cost(My_Tasks, Cost(My_Tasks))    //Initialize cost distribution before observation
    My_PC = Get_Positive_Capabity() //Get positive capability using the ontology described in Section 4.1
    My_NC = Get_Negative_Capabity() //Get negative capability using the ontology described in Section 4.1
    Previous_Task = ø
    Current_Task = Select_Task (Previous_Task)        //Agent selects the first task to do
    Next_Task = Select_Task(Current_Task)             //Plan the second task to do in advance

B] Task_Complete(Current_Task)  //Method called locally when an agent finishes a task
    If (Interrupted)        //If agent was interrupted while doing the previous task
        If (Next_Task == Current_Task) Then
            //Avoid repeating the task that could happen because of interruption
            Next_Task = Select_Task(Current_Task)
        End if
        Interrupted = False
    Else
    //Update task cost distribution based on agent observation using the algorithm described in Fig. 5
     Update_Global_Cost(Current_Task, My_PC, My_NC)
    End if
    Previous_Task  = Current_Task
    Current_Task = Next_Task
    Next_Task = Select_Task(Current_Task)
    My_Tasks = My_Tasks + Previous_Task
    //Inform all agents that this task is completed, its updated cost, and the updated waiting time
    Communicate (Task_complete, Current_Task)
    //Update the global waiting time and communicate it to the result node and other agents
    Update_global_waiting (Current_Task)

C] Task_Not_Complete(Current_Task) //Method called when an agent  is interrupted
    Previous_Task  = Current_Task
    Interrupted = True
    Current_Task = Next_Task
    Next_Task = Select_Task(Current_Task)
    My_Tasks = My_Tasks + Previous_Task

D] Select_Task(Previous_Task)  //Method called by an agent to decide on the next task to do
    New_Task = Task_with_Max_utility(My_Tasks) //Choose the task with highest utility (example Eq. 10)
    Owner_Offer = compute_Negotiation_Matrix(New_Task) //Compute offer cmn' using Eq. 4
    //Initialize all allocation costs before negotiating them for the current task
    Initialize_Negotiation_Matrix(Negotiation_Offers, BIG_NUMBER)
    Communicate(Task_request, IP, New_Task, Owner_Offer)  //Send negotiation request to all agents
    Task_Found = False
    While not Task_Found
        //From Negotiation_Offers return the index of the best offer
        Best_Agent = return_agent_IP_with_highest_Offer(Negotiation_Offers)
        //If the agent with lowest negotiation cost is the same agent who selected the ask then allocate it
        If (Best_Agent = IP)
            My_Tasks = My_Tasks – New_Task
            Return New_Task
        Else
            New_Task = Task_with_Max_utility(My_Tasks)
            Owner_Offer = compute_Negotiation_Matrix(New_Task)
            Initialize_Negotiation_Matrix(BIG_NUMBER)
            //Send the request to all agents to negotiate
            Communicate (Task_request, IP, New_Task, Owner_Offer)

E] Communication_Received(message)  //Each agent could receive task, bid, or result update request
    If message.type = Task_complete  //Update task information based on communicated observation
        Update_global_waiting(message.Current_Task)
        Update_Global_Cost(message.Current_Task)

    If message.type = Negotiation THEN
        Negotiation_Offers [message.IP] = message.My_offer   //Collect received offer

    If message.type = Task_request:
        My_Offer = compute_Negotiation_Matrix(message.New_Task) //Using Eq. 4
        //If this agent is better than the announcer agent then reply with its offer
        If (My_Offer <message.Owner_Offer) THEN
            //Send the offer only to the requester agent (message.IP)
             Message.IP.Communicate (Negotiation, IP, My_Offer)
```

compensated when another agent communicates its updated observation. For example, when one agent finishes a task and communicates the observed cost distribution to the other agents, if a second agent was out of range the announcer repeats the message at least three times while considering delays. If the message does not reach the second agent, the observation is recommunicated by default when a third agent finishes its task. Hence, the probability of missing important information is highly limited in the proposed system.

5 Experimental evaluation

To measure the effect of the semantic negotiation and the incremental update of the uncertain cost, the performance of SAUDE was tested with virtual and real robots. Its performance is compared with three of the latest and most powerful algorithms in DTA literature: DTAG, DTAS, and DTAP. These algorithms have a similar structure to SAUDE; they are fully distributed, perform online task allocation, and support heterogeneous agents by avoiding predefined routes planning. It is thus possible to measure the effects of the new features added to SAUDE's structure. Several trials with different numbers of robots were conducted with each algorithm, and heterogeneity was introduced in the physical capabilities and hardware component levels of the robots.

Before discussing the detailed experimental results, this section presents the common setup for each chosen task and the type of heterogeneity introduced in each experiment. The task and performance measurements are presented after that. Then, the simulation results are analyzed. The details of the experiments conducted with real robots are also presented. An additional experiment was conducted to measure the performance of the methods with real and virtual robots working together. Finally, the common findings are discussed at the end of this section.

5.1 Setup

The proposed method was implemented using the ROS, Python, and C ++ languages and then integrated into a simulated patrolling tool[2] and a real patrolling tool[3] to assess the performance of the methods when applied to real and virtual robots. The simulation is realistic and reflects the same environments and parameters as the real robot tool. The simulator was built using ROS and the task environment was built using ROS_Stage. Hence, the ROS topics and nodes are defined in the same way as in the real robot tool. Each robot, whether it is a virtual robot that works in the simulation or real robot that works in the real environment, publishes its action and subscribes to the tool nodes.

Because the real environment is dynamic and uncertain, the tools were modified to match this new requirement. The tools were altered to accept robots' failure and never stop, even if some robots seem to be dead. This is important in a dynamic environment because robots could take a long time to localize, and thus other robots should perform the task and not stop working. Dynamic obstacles were also implemented to introduce uncertainty. Odometry with ROS navigation was used for distance observations and AMCL was used to collect real distance observations.

In order to benchmark the performance of SAUDE, different experiments were conducted with virtual and real robots, and the performance of SAUDE was compared with three well known benchmarks (DTAP, DTAG, and DTAS). In order to measure the performance with different levels of heterogeneity and dynamicity, it was decided to conduct three main experiments. In the first experiment (virtual robots), the robots had the same hardware components but small differences in their capabilities. In the second experiment (real robots), real robots with different capabilities were used, which increased the dynamicity of the environment, especially because humans were walking in the environment in addition to the obstacles. In the third experiment (real & virtual robots), robots with different hardware components and capabilities were used in addition to the same dynamics introduced in the second experiment, which increased the heterogeneity.

As summarized in Table 1, in the virtual robot experiment, three tests were conducted with two, four, and six robots, respectively. Each test was repeated three times for each method, and a Cumberland map was used as the environment. On the contrary, when real robots were introduced in the second and third experiments, a Massachusetts Institute of Technology (MIT) conference room was used to conduct the experiment. The details of each environment are explained in the respective sections on the experiments.

Variability was introduced through many factors that affect the uncertainty of the cost. In particular, the robots' localization, path planning, communication delays, and obstacle avoidance affected the performance and caused inconsistencies among the trials. Thus, each test was repeated at least three times, and each trial lasted for 30 min with each method. The most significant version was chosen as the main

[2] The simulator was originally developed by David Portugal, and we modified it to include SAUDE and support agents with different capabilities. We also integrated tools for the semantic ontology. The original version of the simulator is available at https://github.com/davidbsp/patrolling_sim.

[3] This is another version of the patrolling simulator that allow real robots to navigate with virtual robots. We improved this version further to support the turtlebots used in the experiment and to included SAUDE. We also integrated the needed tools for the semantic ontology and introduced different capabilities for the robots. The original version of the tool is available at https://github.com/gennari/patrolling_sim.

Table 1 Experiment characteristics and setup

Experiment	Map	#Meth.	#Tests/Meth.	#Trials/Test	Trial Length	#Robots	Dynamics
Virtual	Cumberland	4	3	3	30 min	Two virtual Four virtual Six virtual	Moving obstacles
Real	MIT Facility	4	1	3	30 min	Three real	Moving obstacles and walking humans
Virtual with real	MIT Facility	4	1	3	30 min	Three real and one virtual	Moving obstacles and walking humans

trial. Moreover, the results of all the trials were analyzed by an ANOVA test to identify the common findings of all the trials.

During each trial, the map structure was changed every 10 min by moving some walls and adding others, which produced a map that was different from the one the robots knew. Moreover, when the real robots were introduced, one person would walk in the environment every couple of minutes to create a more realistic situation. This introduced dynamics and uncertainty into the environment. Additionally, several scenarios were considered with different numbers of robots to study the effects of having a small to large number of robots and how well each algorithm could scale.

5.2 Robots characteristics

In order to introduce heterogeneity among the robots, different characteristics and capabilities were defined for each robot, as illustrated in Table 2. In the patrolling task, the velocity and acceleration were found to be positive capabilities, whereas the mass was a negative capability. In the experiments, the hardware components were different between the virtual and real robots. In the virtual robots, a laser was used to collect the observations and a Nvidia GEFORCE GTX processor was used to run the simulation. In the real robots, however, an Asus Xtion Pro 3D depth camera was used to collect the observations and speakers with microphones were used to interact with the environment. The real robots had Nvidia Jetson Tegra K1 processors and Wi-Fi with Bluetooth were used for communication.

In general, the first two experiments considered that the robots had the same hardware components but different capabilities. This would allow the measurement of the extent to which SAUDE improved the allocation when the capabilities were considered in the negotiation. In the third experiment, however, the robots considered were different in terms of their capabilities and hardware components. Hence, it would be possible to test SAUDE with increased heterogeneity and measure its effect on the performance.

5.3 Experiment task

During the experiments, the robots collaborated to patrol one area and visit all the checkpoints as fast as possible to protect it from intruders. The ultimate goal of this patrolling task is to reduce the total patrolling time per cycle, such that no checkpoint is left unattended for a long time. Hence, it is important that each robot select the most appropriate target that could reduce the total wait time and update this allocation depending on the current circumstances. A patrolling problem was chosen because of its dynamic nature. In this problem, the robots can react to changes in the environment and adapt to unexpected situations or failure, and this requires a powerful level of coordination (Farinelli et al. 2017). Robots can have different capabilities, such as speed, weight, or energy, and they need to determine the cost of reaching a certain target and negotiate to determine who is better to go to a certain target. Although patrolling tasks have been studied before, the performance of DTA with distributed and online decisions in real robots have been neglected, and the focus of most studies on patrolling is typically path planning and coordination.

In the patrolling problem, the shortest path can change with respect to the uncertain distribution of the cost. In SAUDE, the uncertain space of patrolling can be defined as $\{V, E, \xi\}$, where $V = \{1,\ldots, n\}$ is the set of tasks, $E = \{(i,j) \mid i,j \in V\}$ is the set of arcs (task order), and $\xi = \{\xi_{i,j} \mid i,j \in A\}$ is a set of positive uncertain variables representing the arc cost, where $\xi_{i,j}$ are independent $\forall$ $i,j \in A$. The shortest path length $f(\xi)$ is an uncertain variable and can be discovered using the Dijkstra algorithm for the inverse uncertain distribution ($\psi^{-1}(\alpha)$), where $\psi^{-1}(\alpha) = f(\Phi_1^{-1}(\alpha), \Phi_2^{-1}(\alpha)\ldots, \Phi_n^{-1}(\alpha))$. To select a task for negotiation, a common measure called utility is used to measure the reward of selecting task n. This reward (Eq. 10) is the target waiting time from the last time it was served minus the uncertain cost of doing the new task n. This way, targets that have waited longer with shorter distance to a robot are chosen first.

$$R(m, n) = WaitingTime(n) - \psi_n^{-1}(\alpha) \tag{10}$$

Table 2 Robots characteristics and hardware components

Experiment	Robots	Positive Capability	Negative Capability	Hardware Components	Communication Components
Virtual	Robot_0	Velocity = 0.4	Mass = 0.10	Laser sensor Nvidia GEFORCE GTX processor Kobuki mobile base	Wi-Fi
	Robot_1	Velocity = 0.15	Mass = 0.01		
	Robot_2	Velocity = 0.4	Mass = 0.01		
	Robot_3	Velocity = 0.1	Mass = 0.20		
	Robot_4	Velocity = 0.4	Mass = 0.10		
	Robot_5	Velocity = 0.15	Mass = 0.01		
Real	Robot_0	Velocity = 0.15 Acceleration = 1.5	Mass = 2.4	Xtion Pro 3D depth camera sensor Microphone Speakers Nvidia Jetson Tegra K1 processor Kobuki mobile base	Wi-Fi Bluetooth
	Robot_1	Velocity = 0.20 Acceleration = 1.75	Mass = 2.4		
	Robot_2	Velocity = 0.25 Acceleration = 2	Mass = 2.4		
Virtual with real	Robot_0	Velocity = 0.15 Acceleration = 1.5	Mass = 2.4	Xtion Pro 3D depth camera sensor Microphone Speakers Nvidia Jetson Tegra K1 processor Kobuki mobile base	Wi-Fi Bluetooth
	Robot_1	Velocity = 0.20 Acceleration = 1.75	Mass = 2.4		
	Robot_2	Velocity = 0.25 Acceleration = 2	Mass = 2.4		
	Robot_3	Velocity = 0.4	Mass = 0.10	Laser Sensor Nvidia GEFORCE GTX processor Kobuki mobile base	Wi-Fi

5.4 Performance measurements

During the experiments, the targets' waiting time reported by each robot is recorded. From this information, different analytical results are computed and analyzed. First, the average, standard deviation, and maximum waiting time of all targets are analyzed. These measures show the general behavior of each method and how much waiting time each method creates on average. The result is represented in a bar graph, where the x-axis represents each method and the y-axis represents the metrics of the waiting time.

To compare the behavior of each method as the number of robots increases, the waiting time frequency is illustrated in a temporal histogram. The histogram shows the density of the waiting time; for example, the methods that allocate robots to visit the targets more frequently have higher frequency on the low side of the waiting time axis. In the histogram figures, the x-axis represents the waiting time and the y-axis represents the frequency reported for each range of time. Each method is visualized with different colors to compare the difference as the number of robots increases.

To analyze the effect of the variability introduced by real robots, it is important to measure the variance of the targets' waiting time. A low variance indicates that the result is more consistent and less affected by the robots' variabilities and the environment's consistency. This information is illustrated using a temporal scatter plot. In this plot, the x-axis represents the targets' IDs, and the y-axis represents their waiting time. Low and compressed results in the plot show that the algorithm has a low variance and persistently low waiting time. Outliers can also be visualized in this type of plot.

In addition to measuring the performance for the most significant trial, each test was repeated three times with each method. This was done because a method could be affected by the variability and perform differently depending on the current situation. Thus, it was decided to measure the results over more than one trial and apply a statistical test to measure the total difference among all the methods with respect to all the trials. An ANOVA test was chosen to measure the significant difference while using a Tukey pairwise test. The results are visualized using a critical difference (CD) diagram to show the group to which each method belongs and their rank when considering the performance over all the trails. In this graph, the x-axis represents the percentage of the least square (LS) mean of the waiting time from the total LS means. Methods that belong to the same group have insignificant differences, and are indicated by lines connecting them. It is thus possible to represent which methods have significantly better or worse results than the others.

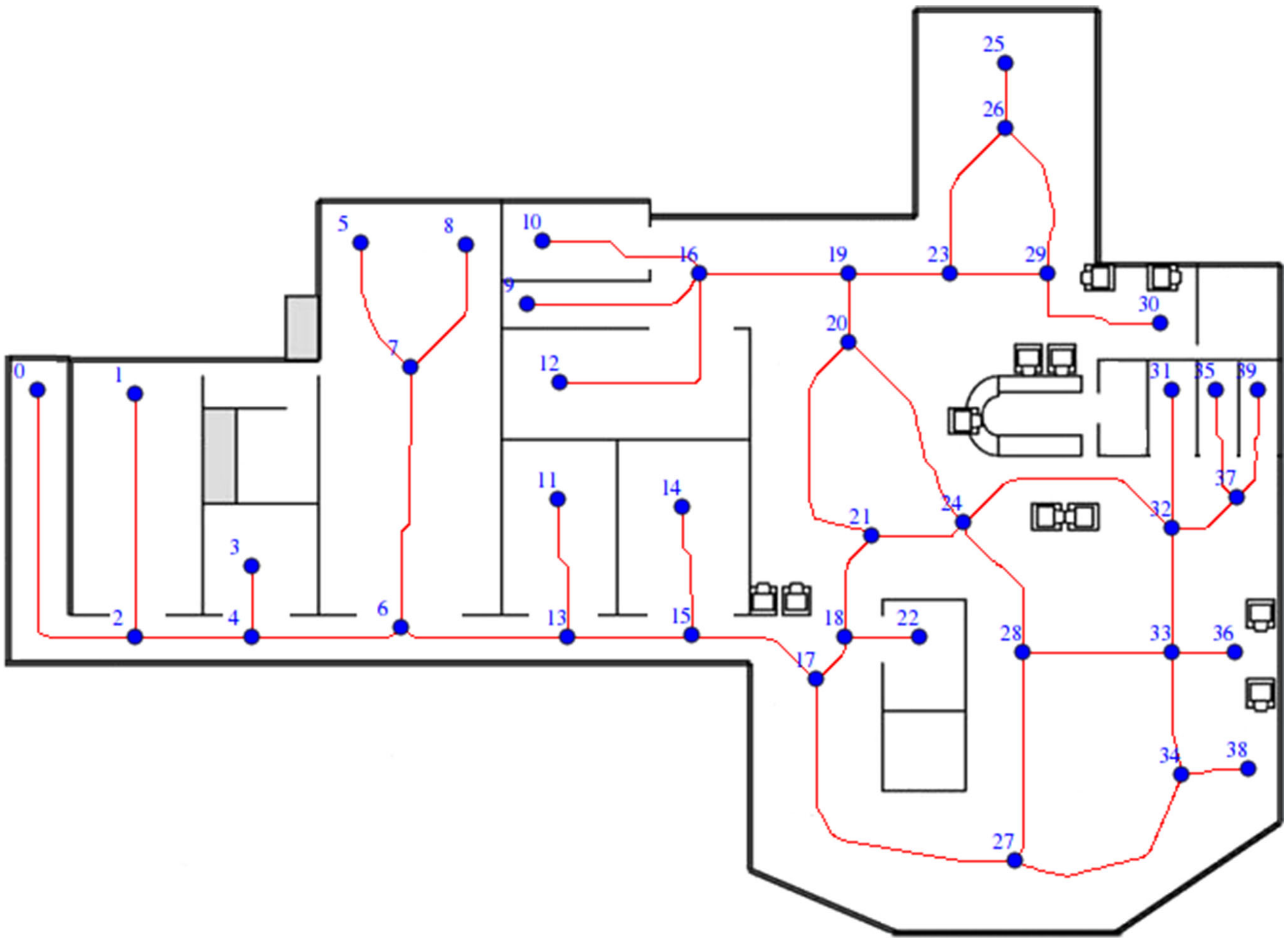


Fig. 7 Environment of virtual robot experiments

5.5 Virtual robots experiment

Before testing the performance on real robots, it is important to measure the performance in a simulation with better control over the environment. The performance was measured in three tests with two, four, and six robots, respectively. Each test was repeated three times for each method, and a Cumberland map was used as the environment. The Cumberland map, illustrated in Fig. 7, consists of 40 targets for the patrolling task, and the room walls were implemented to move using ROS_Stage to create dynamics in the environment and available tasks.

In order to measure the performance of each method, the average (mean), maximum (max), and standard deviation (std dev) of the waiting time were collected at the end of each test as illustrated in Table 3. From the results, it is clear that SAUDE had the best average waiting time in all the tests, regardless of the number of robots used. DTAP and DTAG were similar to each other in terms of the average waiting time, again regardless of the number of robots. DTAS, however, behaved differently in each experiment. Its average waiting time was better than DTAP and DTAG when the number of robots was two but worse than the others with four and six robots. However, this is not the case when considering the std dev and max of the waiting time.

When considering the std dev and max of the waiting time, DTAS was much worse than the other algorithms in the all tests. DTAP and DTAG interchanged their order of performance from one test to another. When two robots were used, DTAG had the lowest std dev and max waiting time, and SAUDE came in second place. When four robots were used, SAUDE came in second place after DTAP in terms of the std dev, but it had the lowest max waiting time. When six robots were used, SAUDE came in third place after DTAP and DTAG, respectively, in terms of the std dev and max waiting time.

SAUDE reduces the average waiting time, even though the std dev is not the least. This indicates that its allocation was initiated faster than those of the other algorithms. In terms of the std dev, although SAUDE is not far from the lowest method, further investigation is required to understand this difference, as discussed next. As for the maximum waiting

Table 3 Waiting time metrics in seconds with two, four, and six virtual robots

	SAUDE	DTAP	DTAS	DTAG
Two virtual robots				
Mean	**270.5**	357.5	291.7	351.4
std dev	173.1	143.2	203.7	**141.6**
Max	582.9	733.1	1134.4	**565.9**
Four virtual robots				
Mean	**121.6**	153.8	403	149.7
std dev	77.6	**72.7**	412.3	89.2
Max	**396**	450.5	1218.8	520.8
Six virtual robots				
Mean	**93.2**	100.6	147.2	96.6
std dev	103.7	**55.9**	110.6	63.8
Max	776.1	**307.2**	1284.1	350.5

The bold means the least values explained in the text, it was only identified for visualization and easier readings

time, the behavior of SAUDE changes with the number of robots, owing to the variability of the environment. It is also possible that the maximum value is an outlier in the system. Hence, further analysis of its frequency is conducted next.

The frequency of the waiting time range is recorded to compare the behavior of the methods as the number of robots increases. As illustrated in Fig. 8, SAUDE behaved consistently in all the tests. Regardless of the number of robots, most of the targets in SAUDE waited less than 200 seconds (s), which is still better than the other algorithms. This result confirms that SAUDE initiates the allocation faster than the other methods, and can coordinate well, even if there is a lack of resources. Although all the other algorithms behaved worse with a lower number of robots, SAUDE's visiting frequency was consistent in all the experiments.

On the contrary, in DTAP and DTAG, most of the targets' waiting times were between 300 and 600 s when two robots were used. Most of targets waiting times in DTAS were between 200 and 500 s, and some targets were left waiting more than 1200 s. As the number of robots increased, DTAP, DTAG, and DTAS behaved differently. With four or six robots, most of the targets' waiting times were less than 300 s in DTAP, DTAG, and DTAS. This indicates that these three methods cannot coordinate well when the number of robots is low. Although scalability is important, dealing with a low number of resources is also important, especially if the organization is financially limited and cannot afford a high number of robots. DTAS coordination took even more time with an increased number of robots, which explains the sudden difference in waiting time between DTAS and the other methods.

After comparing the performance for the most significant trial, it can be seen that the performance of a method

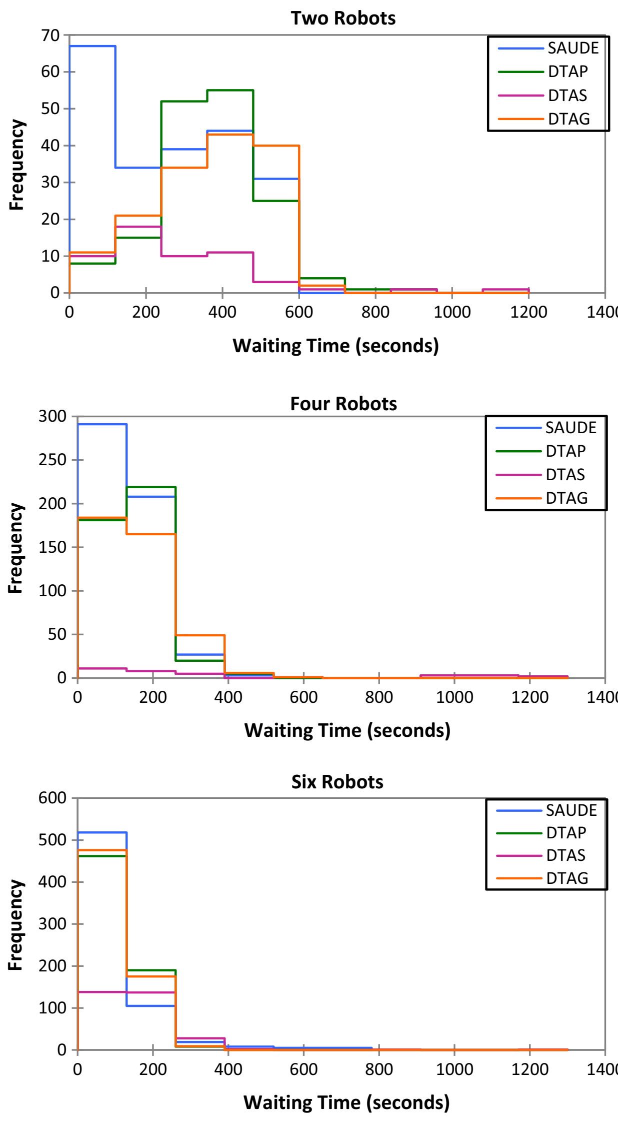


Fig. 8 Waiting time histogram of virtual robots' experiments

sometimes changes from one test to another. This is due to the effect of dynamics and uncertainty in the environment. Although the experiments were conducted in a simulation, variability still existed to some extent, which affected the consistency of the methods. Therefore, every test was carried out three times for each method, and the global waiting time for all targets was recorded every 3 min. Using an ANOVA statistical test with Tukey pairwise test, it is possible to accurately measure the significant difference among all methods.

The results of the two-robot test, illustrated in Fig. 9, show that SAUDE ranks first with respect to all the trials. The line connecting SAUDE and DTAS indicates that these two methods belong to the same group but SUADE comes in first place. The repeated trials verify that SAUDE's reduction in waiting time is insignificant compared to DTAS when two robots are used. However, DTAP has significantly worse results than

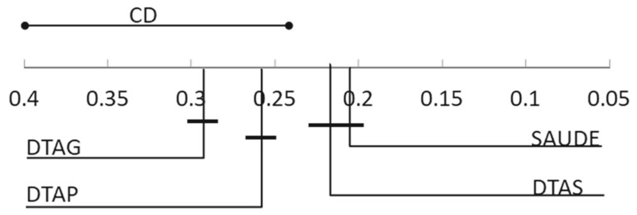


Fig. 9 ANOVA with Tukey for two-virtual-robots' experiment

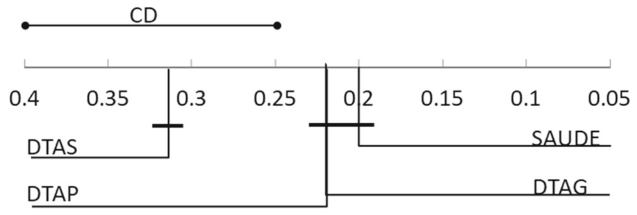


Fig. 10 ANOVA with Tukey for four-virtual-robots' experiment

SAUDE and DTAS and better results than DTAG. Finally, DTAG shows the worst results.

When using four robots, some methods behaved differently, as illustrated in Fig. 10. SAUDE still ranked first, whereas DTAG jumped to second place and had an insignificant difference compared to SAUDE and DTAP. This indicates that DTAG can perform better with a higher number of robots. DTAP came in third place, showing an insignificant difference compared to DTAG and SAUDE when repeating all the tests. However, this is not the case with DTAS; it came in last place when using four robots, showing significantly worse results than the other methods. This result can be attributed to the frequency measured in Fig. 8, which indicate that DTAS took more time to coordinate with an increased number of robots. Moreover, Table 3 shows that the std dev increases significantly when increasing the number of robots in DTAS compared to DTAP and DTAG; this was concealed by the low average of DTAS when the number of robots was small but became more obvious with an increased number of robots. Hence, DTAS is not as scalable as the other methods.

Figure 11 shows the results of increasing the number of robots to six, which confirm the conclusion reached with four robots. SAUDE scaled the best and always came in first place, even when the tests were repeated. However, when the number of robots increased to six, the improvements in performance became more significant. SAUDE alone belongs to the first group, indicating that it can coordinate better than the other methods, and its allocation can significantly improve task performance. DTAG and DTAP are still in second and third place, respectively, but they belong to a group with significantly worse results than SAUDE and better results than DTAS. Finally, DTAS came in last place with significantly worse results than the other methods, indicating that its

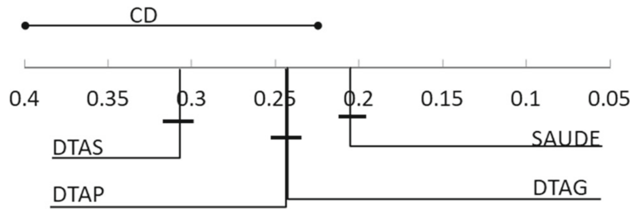


Fig. 11 ANOVA with Tukey for six-virtual-robots' experiment

allocation strategy affected the robots' scalability and coordination.

From the ANOVA results of all the tests, SAUDE scales better than the other methods when increasing the number of robots. Even though it performs well with a low number of robots, the improvement becomes more significant when increasing the number of robots. SAUDE is consistent, coming in first place regardless of the number of robots. However, to further confirm these results, it was important to test the performance using a real environment with real robots and analyze the results for the case with more variability and true dynamics. The results of this test are presented and discussed in the following sections.

5.6 Real robots experiment

To test the performance with real robots, three Turtlebots known as Kobuki were used. The experiments were conducted on the MIT campus, and the network used for communication was the campus open network. This added more variability to the test owing to the shared and uncontrollable provider. ROS was used to build the map using one of the robots, and the map was shared among the robots and used for the patrolling task. Figure 12 shows the map built based on the MIT facility, where ten targets were chosen for the patrolling task, and movable tables were folded and used as portable walls to create dynamics in the environment.

In each trial, the methods were run for 30 min, and some of the walls were moved every 10 min. One person walked in the environment every couple of minutes to create a more realistic situation. In terms of the capabilities, Robot_0 was the fastest and had the highest acceleration, followed by Robot_1 and Robot_2, respectively. All three robots had similar masses.

Figure 13 illustrates the waiting time metrics, which show that SAUDE had the lowest values in all the measures. Compared to SAUDE and DTAP, the performances of DTAS and DTAG were not efficient. This indicates that they took more time to serve the targets and their plan caused the robots to neglect more targets. Moreover, unlike the simulation experiment, SAUDE had a lower average, std dev, and max waiting time than DTAP, DTAG, and DTAS. This is due to the cost update and capability reflection introduced in SAUDE, which

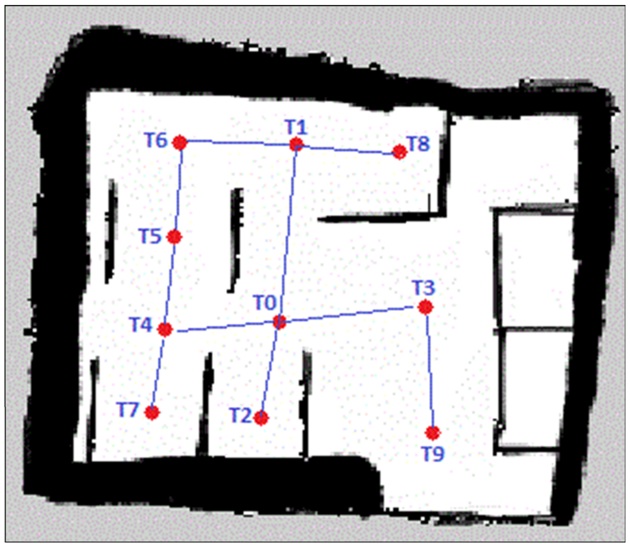
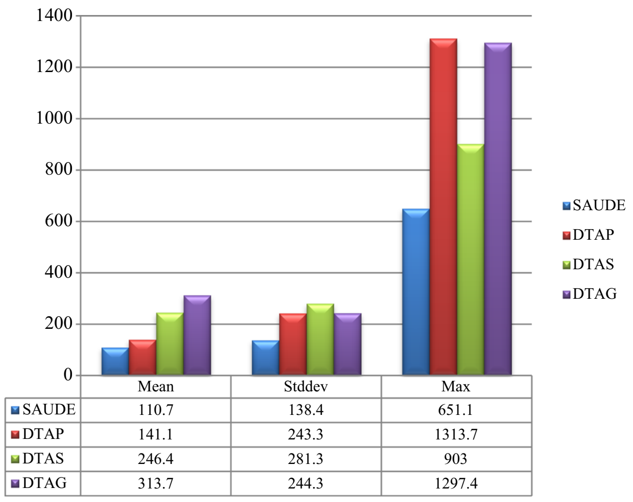


Fig. 12 Environment of real robots' test



	Mean	Stddev	Max
🟦 SAUDE	110.7	138.4	651.1
🟥 DTAP	141.1	243.3	1313.7
🟩 DTAS	246.4	281.3	903
🟪 DTAG	313.7	244.3	1297.4

Fig. 13 Waiting time metrics (in seconds) with three-real-robots' experiment

cause the method to perform better with increased the dynamics and heterogeneity. Hence, SAUDE is more scalable than the other methods with respect to the environmental dynamics and heterogeneity.

In addition to the average waiting time, it was important to measure the variance in the targets' waiting time to understand the consistency of the allocation. Using a temporal scatter plot to represent the results shows the consistency of the methods over time and the frequency of the outliers that represent the neglected targets. From the results, illustrated in Fig. 14, SAUDE has a lower distribution compared to the other methods. The results for DTAS are the most scattered, indicating that the allocation variance is very high. Hence, some targets were served frequently while other targets were neglected for more than 800 s. DTAP and DTAG have less variance than DTAS, but the figure shows outliers in these methods. This indicates that some targets were neglected for a very long time compared to the other targets.

SAUDE and DTAP were compressed more at the beginning. In the first 10 min, both methods served the targets much more frequently than later. After the first ten min, robots slowed down and took more time to allocate their tasks. This behavior was caused by the obstacles starting to move after the first 10 min. This change in the environment caused the robots to take more time to determine new route plans, because the environment was different from the original one. However, their behavior was still acceptable and produced a much lower waiting time than DTAS and DTAG. This, in general, indicates that SAUDE is more consistent than the other methods, and its allocation strategy allowed the robots to consider all the targets and avoid having outliers.

The performance analysis of the algorithms with repeated trials using ANOVA and Tukey pairwise tests is illustrated in Fig. 15. SAUDE belongs to the first group with DTAP. However, even though SAUDE showed better waiting time metrics and variance, the ANOVA test shows that it came in second place after DTAP. This can be attributed to uncontrollable reasons caused by the uncertainty introduced by the real robots and network connections. It was noticed that when repeating the test three times for each method, the order based on the average waiting times for DTAP and SAUDE interchanged. Computing the LS mean for all the trials exposed this effect and measured their performance under such circumstances. However, it is important to note that although SAUDE comes after DTAP, the difference is insignificant.

The allocation plan produced by SAUDE significantly reduced the targets' waiting time compared to DTAS and DTAG, whereas the difference in the average waiting time was insignificant compared to DTAP. Moreover, DTAS came in third place and belongs to the second group with significantly better results than DTAG. Finally, DTAG came in last with significantly worse results than the other methods. This indicates that DTAG's greedy behavior causes the robots to take more time to coordinate when the environment changes. Even when repeating trials with real robots, SAUDE belongs to the first group in terms of performance. Nevertheless, measuring the performance using real robots that have different physical capabilities but similar hardware components is insufficient. It is also important to measure the performance with more robot heterogeneity, as explained in the following section.

5.7 Real with virtual robots experiment

To further analyze the effect of heterogeneity on the level of hardware components and physical capabilities, it was

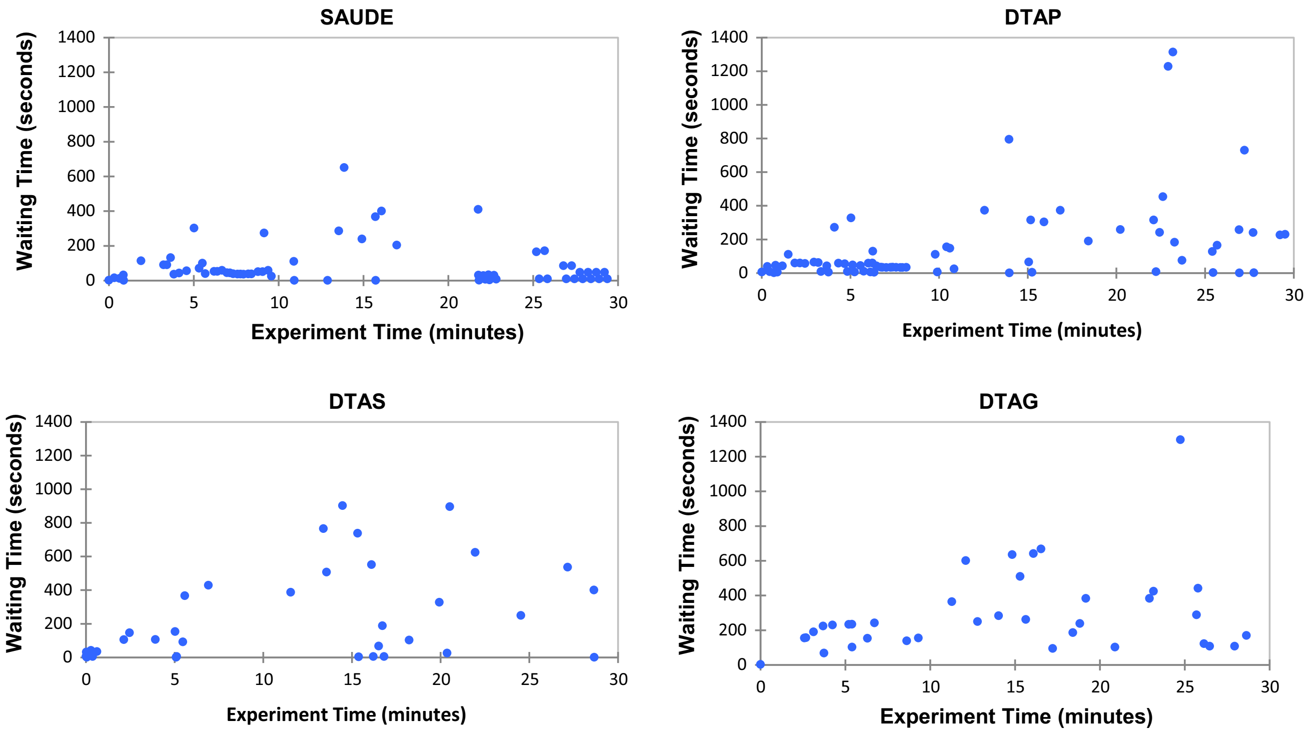


Fig. 14 Temporal scatter pot with three-real-robots' experiment

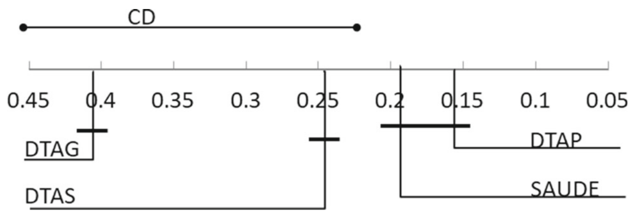
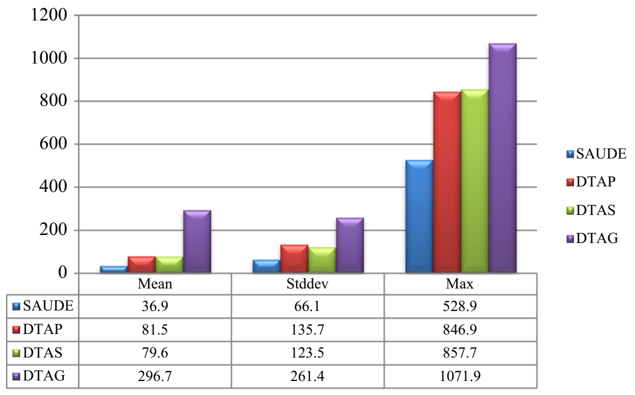


Fig. 15 ANOVA with Tukey for three-real-robots' experiment



	Mean	Stddev	Max
SAUDE	36.9	66.1	528.9
DTAP	81.5	135.7	846.9
DTAS	79.6	123.5	857.7
DTAG	296.7	261.4	1071.9

Fig. 16 Waiting time metrics (in seconds) for four-real-with-virtual-robots' experiment

decided to repeat the real experiment and add virtual robots that communicate with the real robots to perform the task together. The virtual robots have different hardware components, and the definitions of their capabilities are also different. The same setup and number of trials used in the real robot experiment are used in this experiment, but the total number of robots increased to four because of the additional virtual robot.

During the experiment, as in the case of the real robot experiment, it was noticed that SAUDE's performance was much better than the other algorithms when virtual robots were introduced. SAUDE reduced the waiting time by more than 50% compared to all the other methods. As shown in Fig. 16, the mean, std dev, and max waiting time of SAUDE are the lowest compared to DTAP, DTAG, and DTAS. This reduction is even more than the result recorded in the previous experiments, confirming that SAUDE is more scalable with respect to dynamics and heterogeneity. Even though virtual and real robots were both used in this experiment, SAUDE managed to communicate and allocate tasks better than the other methods. Hence, the semantic negotiation introduced into SAUDE allows the robots to coordinate better than the other methods when the heterogeneity among the robots increased. DTAP and DTAS had similar results to each other, whereas DTAG was significantly worse. This result also confirms that DTAG cannot scale well with heterogeneous robots, and its performance cannot be guaranteed in this case.

When considering the variance in the targets' waiting time with respect to the experiment time, the results found in the

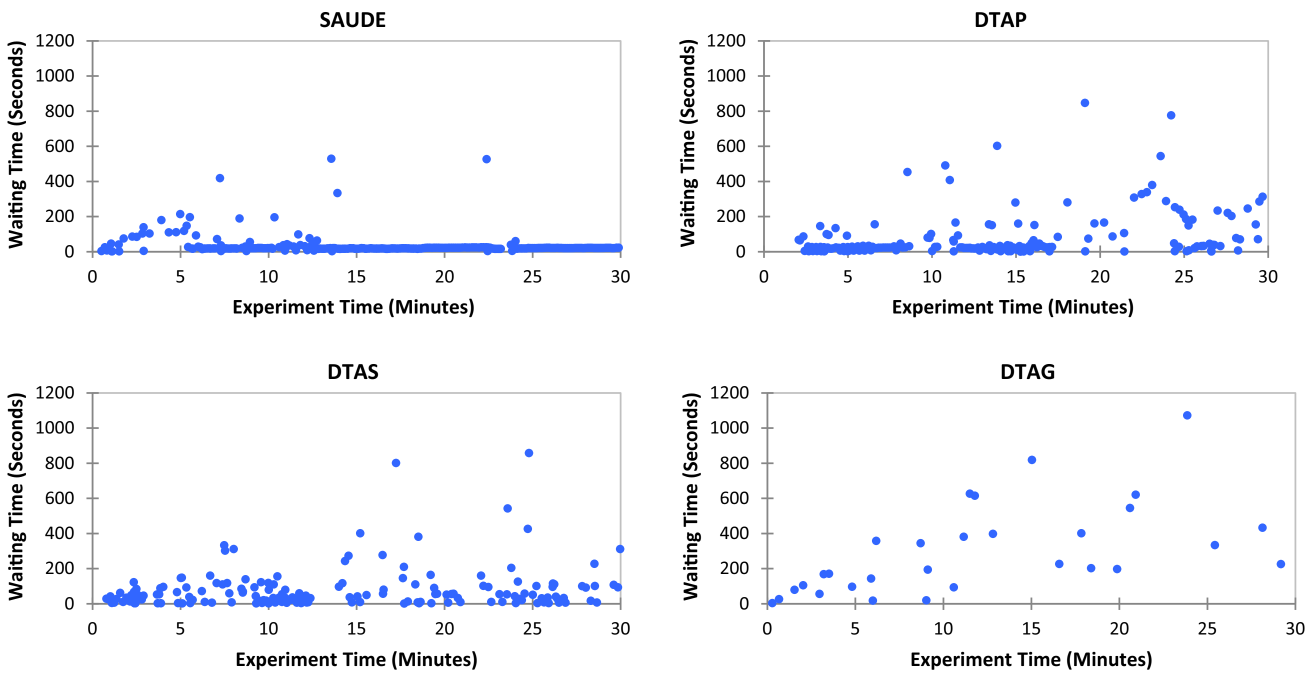


Fig. 17 Temporal scatter pot for four-real-with-virtual-robots' experiment

real robot experiment are also confirmed here. As shown in Fig. 17, the difference between SAUDE and the other methods is more obvious than in the previous experiments. The distribution in SAUDE is lower than in the other methods, to the extent that it appears almost as one line, especially in the second half of the test. This indicates that robots using SAUDE allocated themselves equally to serve the targets with a low waiting time. DTAP and DTAS have a similar variance to each other, but it is slightly more compressed in DTAS. On the contrary, the results for DTAG are highly scattered, which explains the high waiting time metrics discussed before.

After the first 10 min, the obstacles started to move and much like the previous experiment, DTAP, DTAS, and DTAG performed worse because the robots took longer to finish. However, it was interesting to find that this was not the case with SAUDE, where the variance dropped after the first 10 min. This is possible because the method considered the capabilities of the robots, and the virtual robots found the plan faster than the real robots. Thus, the allocation utilized the available resources regardless of their heterogeneity, allowing the virtual robots to do more tasks. This indicates that SAUDE allowed robots to coordinate better than the other methods, even though the robots had different hardware components.

This experiment was also repeated three times for each algorithm, and the global waiting time of all the targets was recorded every 3 min. ANOVA statistical tests with Tukey pairwise analyses were also applied to the results to illustrate the significance, as shown in Fig. 18. From the results,

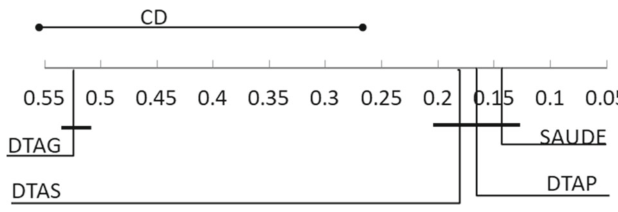


Fig. 18 ANOVA with Tukey for four-real-with-virtual-robots' experiment

SAUDE's rank is better compared to the other methods when robots' heterogeneity increased. Although there is no significant difference between SAUDE, DTAP, and DTAS when considering all the trials, the result that SAUDE ranks better compared to the previous experiments gives an intuition that SAUDE can scale better with the increased variability.

When considering the ANOVA results, DTAG has significantly worse results than SAUDE, DTAP, and DTAS. This verifies that the greedy approach is not an appropriate solution when heterogeneity and dynamics are introduced into the problem. Not only does DTAG have significantly worse results, but its rank is also far from the other methods, indicating that it suffered the most from the increased heterogeneity.

6 Discussion

In the proposed DTA method, the challenges identified in the literature are addressed and new features are introduced. In SAUDE, the allocation is fully distributed and agents do not need to understand each other's model as long as they understand their own. The allocation can be performed online, and the communication overhead is reduced by attaching the information with the acknowledgement messages instead of sending them separately. The use of semantic-based negotiation optimized the task allocation with respect to agents' capabilities and the cost. In this strategy, the current, past, and new workloads are also considered to balance the workload.

Because of the use of the local ontology, it is possible for each agent to identify its positive and negative capabilities and compute its offer using the negotiation matrix. This offer is computed locally by each agent, and thus the agents can work in parallel to speed up the negotiation, and communication failures do not affect the system. During the negotiation, all the agents are performing other tasks, and they do not have to wait to start negotiating. When a new task is available, each agent decides online what the next subtask to perform is and collaboratively finish the given task. Therefore, each agent starts to select the next job while it is performing its current sub-task.

Because the cost distribution is updated incrementally using the uncertainty theory, it was possible to immediately reflect changes in the environment to the allocation while preserving its consistency. This allows the agents to avoid pre-computation and update the cost with respect to the current observation. Memory overloading is also avoided by preserving only the last observation. Excessive computation when computing the shortest path, as in the current uncertainty-theory-based methods, is avoided by selecting only one confidence value and computing the cost accordingly.

From the empirical results, it is possible to note some general observations of the online distribution of the DTA methods tested in the experiment. When it comes to the greedy-based method, even though DTAG behaved well in the simulation, its performance was poor with real robots. DTAS, however, performed better when applied to real robots compared to the simulation. DTAP and SAUDE showed good performance in both cases. However, SAUDE was more consistent, and its significance increased with increased heterogeneity and dynamics.

When testing the methods with different numbers of robots using the simulation, the targets' waiting times were consistent with SAUDE. Even though the ANOVA results showed that SAUDE did not perform significantly better than the other methods when the number of robots was low, it must be emphasized that SAUDE came in first place, regardless of the number of robots. It also served the targets equally, whereas methods with similar average waiting times resulted in outliers and neglected certain targets. Because of the proposed negotiation matrix, it was possible to utilize the available resources and balance the workload to serve all the targets. This consistency not only confirms the workload balance, but also the scalability of SAUDE with a large number of robots.

In addition, applying SAUDE to real and heterogeneous robots further confirmed its scalability, not only with respect to the number of robots, but also the environmental variability and robots' heterogeneity. Its overall performance was much better than the rest of the methods, and it introduced lower variance, even if the robots had different hardware components. This reduction in variance shows that SAUDE is consistent and that it allows robots to efficiently perform their tasks, even with increased variability. This was possible because of the incremental update of the cost, where the uncertainty was reflected in the selection decision and continuously updated.

When real and virtual robots are working together, the variance in SAUDE, unlike with other methods, became even better after changing the environment. This occurred because the method considered the capabilities of the robots, and the virtual robots found the plan faster than the real robots could. Thus, the allocation utilized the available resources regardless of their heterogeneity and allowed the virtual robots to perform more tasks. This indicates that SAUDE's balance between the quality and efficiency of the allocation worked well, allowing robots to coordinate better than the other methods, even though the robots had different hardware components.

In addition, general observations were recorded during the experiments. When SAUDE was used, it was noted that robots with good capabilities interrupted robots with bad capabilities and took over their tasks. This was possible because the assigned robot took too much time while a better one that had finished its current task was found to be more suitable. This contributed to completing the tasks more quickly and utilizing the available resources as efficiently as possible. Moreover, in all the methods, blocking a road on the map caused the robots to take more time to reach the targets on the other side of this road, such as going to target "eight" when blocking the road between target "zero" and "one," as illustrated in Fig. 12. The robots tried to localize themselves and find a new path plan to visit the blocked targets, and in most cases, this took so much time that the robot dropped the task and did something else. However, in SAUDE, when a robot found that a blocked road was open again, it was communicated with the other robots, and the robots with best capabilities allocated themselves to the neglected targets. This was possible because of the robots' semantic capabilities and cost-updating features. Because of this behavior, fast robots were assigned to targets with a higher priority, thereby reducing the total waiting time. Nevertheless, it must be indicated that although trails were repeated to uncover common results, such results could change with different maps. Further tests are needed in the future for such scenario to ensure whether this statement holds or not.

With respect to the drawbacks of SAUDE, as recorded from the experiments, it was found that this method performs better with real robots than the simulation. The waiting time metrics showed that SAUDE is not far away from DTAP and DTAG when the environment is controlled with low heterogeneity. However, it significantly improves with increased dynamicity and heterogeneity in the real robot experiments. It cannot be said that this is a bad behavior, because this study is targeting the problem of heterogeneity; but it is important to study the cause of this behavior as other factors such as map type could have an effect. From our observations, when the environment is less dynamic and heterogeneous, the performance was more strongly affected by the route plan than the allocation efficiency. Hence, in the future, new path planning strategies needs to be developed to work with the proposed allocation method, such that robots can better plan their tasks when they are homogenous.

In the real robot experiments, SAUDE showed the lowest waiting metrics but the ANOVA test showed that SAUDE comes in second place after DTAP. Although the difference was insignificant, this behavior was attributed to uncontrollable reasons caused by the network connections. A further investigation needs to be conducted in the future with different levels of communication failure to test whether this behavior is consistent or only due to outliers in the system. Moreover, it was noted that the communication slowed down when obstacles were moving. Although SAUDE was not affected by the changes as much as the other methods, this reduction is attributed to route re-planning. To solve this problem, a novel planning algorithm must be developed to continuously plan possible routes with respect to a changing environment and avoid temporarily stopping the robot to come up with a new plan.

Finally, even though SAUDE addressed some of the current gaps in the literature and improved the performance, there were some technical issues faced when applying the methods to the Turtlebot robots. In this research, the biggest technical issue in using Turtlebots with the Tegra K1 processor was that the processor can only work on the ARMv7 operating system, which is incompatible with many applications and tools. In particular, it was not possible to install prolog on the robots' processors; to solve this problem, an external PC was used to run the prolog server and communicate the ontology with the robots. However, this did not affect the distribution of the robots, because the PC was only used at the beginning to bridge each robot with its capabilities. Another way to avoid this incompatibility problem could be by replacing the Tegra K1 processor with a notebook and attaching it on the robot. However, this would increase the monetary cost, as each robot would require a separate notebook.

7 Conclusion

In this paper, an online distributed DTA method was proposed, and its performance was analyzed. The main purpose was to avoid the current problems in the literature that arise because of dynamic environments and agents' heterogeneity. In the proposed method, the semantics of the agents' capabilities were adopted into a negotiation matrix. This matrix allowed each agent to rank itself based on its capabilities, workload, and task cost. Based on this rank, it was possible for each agent to allocate itself to the most suitable subtask and coordinate with the other agents to collaboratively complete a given task. The task cost was incrementally updated using the uncertainty theory to overcome the environment's dynamics and agents' variability. It was possible to reflect the current observations and overcome their uncertainty using the proposed cost function.

From the empirical results, the use of semantic-based negotiation and incrementally updating the cost can improve the allocation efficiency and preserve its quality. The simulation showed that SAUDE allows the robots to be consistent in serving their targets, whether there are many or few robots. The negotiation strategy utilized all possible resources and caused the robots to change their allocation strategy depending on the current circumstances. Applying SAUDE to real robots confirmed the method's consistency and showed that the incremental update of the cost reduced the effects of changes in the environment. Applying the method to robots with different physical capabilities and hardware components further showed the advantages of semantic-based negotiation. The performance of the robots significantly improved and overcame the environment's variability and uncertainty.

Although the results of the experiments indicate how semantic-based negotiation and uncertain cost updating are helpful to task allocation, further tests need to be conducted with real robots in a real environment. In the future, the number of virtual and real robots should be increased, and the performance of the proposed method should be tested in a more complex environment with a different communication level and different types of maps. Autonomous mapping can also be introduced into the real robots to allow the robots to build their maps during navigation. Currently, in SAUDE, the ontology is built offline using the task and the robot's description. Therefore, autonomous semantic ontology tools should be integrated with KnowRob to allow the robots to build their own ontologies and continuously update the task description online.

Acknowledgements The authors would like to thank Saudi Aramco for funding the research reported in this paper under the "Saudi Aramco Ibn Khaldun Fellowship for Saudi Women" in partnership with the Center for Clean Water and Clean Energy at MIT. Further thanks are given to

Katy G Muhlrad for helping in implementing the ontological part of the proposed method.

References

Bimba, A. T., Idris, N., Al-Hunaiyyan, A., Mahmud, R. B., Abdelaziz, A., Khan, S., et al. (2016). Towards knowledge modeling and manipulation technologies: A survey. *International Journal of Information Management, 36*(6, Part A), 857–871. https://doi.org/10.1016/j.ijinfomgt.2016.05.022.

Choi, H. L., Brunet, L., & How, J. P. (2009). Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics, 25*(4), 912–926. https://doi.org/10.1109/tro.2009.2022423.

Drenjanac, D., Tomic, S. D. K., & Kühn, E. (2015). A semantic framework for modeling adaptive autonomy in task allocation in robotic fleets. In *2015 IEEE 24th international conference on enabling technologies: Infrastructure for collaborative enterprises, 15–17 June 2015* (pp. 15–20). https://doi.org/10.1109/wetice.2015.29.

Farinelli, A., Iocchi, L., & Nardi, D. (2017). Distributed on-line dynamic task assignment for multi-robot patrolling. *Autonomous Robots, 41*(6), 1321–1345. https://doi.org/10.1007/s10514-016-9579-8.

Gao, Y. (2011). Shortest path problem with uncertain arc lengths. *Computers & Mathematics with Applications, 62*(6), 2591–2600. https://doi.org/10.1016/j.camwa.2011.07.058.

García, P., Caamaño, P., Duro, R. J., & Bellas, F. (2013). Scalable task assignment for heterogeneous multi-robot teams. *International Journal of Advanced Robotic Systems, 10*(2), 105. https://doi.org/10.5772/55489.

Hristoskova, A., Agüero, C. E., Veloso, M., & Turck, F. D. (2013). Heterogeneous context-aware robots providing a personalized building tour. *International Journal of Advanced Robotic Systems, 10*(1), 14. https://doi.org/10.5772/54797.

Hsieh, C.-F., Chen, R.-C., & Huang, Y.-F. (2014). Applying an ontology to a patrol intrusion detection system for wireless sensor networks. *International Journal of Distributed Sensor Networks, 10*(1), 634748. https://doi.org/10.1155/2014/634748.

Huang, H., & Zhuo, T. (2017). Multi-model cooperative task assignment and path planning of multiple UCAV formation. *Multimedia Tools and Applications*. https://doi.org/10.1007/s11042-017-4956-7.

Iocchi, L., Nardi, D., Piaggio, M., & Sgorbissa, A. (2003). Distributed coordination in heterogeneous multi-robot systems. *Autonomous Robots, 15*(2), 155–168. https://doi.org/10.1023/a:1025589008533.

Khamis, A., Hussein, A., & Elmogy, A. (2015). Multi-robot task allocation: A review of the state-of-the-art. In A. Koubâa & J. R. Martínez-de Dios (Eds.), *Cooperative robots and sensor networks 2015* (pp. 31–51). Cham: Springer.

Liu, B. (2015). *Uncertainty theory* (5th ed.)., Springer Uncertainty Research Berlin: Springer.

Liu, L., & Shell, D. A. (2012a). Large-scale multi-robot task allocation via dynamic partitioning and distribution. *Autonomous Robots, 33*(3), 291–307. https://doi.org/10.1007/s10514-012-9303-2.

Liu, L., & Shell, D. A. (2012). Tackling task allocation uncertainty via a combinatorial method. In *2012 IEEE international symposium on safety, security, and rescue robotics (SSRR), 5–8 Nov. 2012* (pp. 1–6). https://doi.org/10.1109/ssrr.2012.6523871.

Liu, H., Zhang, P., Hu, B., & Moore, P. (2015). A novel approach to task assignment in a cooperative multi-agent design system. *Applied Intelligence, 43*(1), 162–175. https://doi.org/10.1007/s10489-014-0640-z.

Lozenguez, G., Adouane, L., Beynier, A., Mouaddib, A.-I., & Martinet, P. (2016). Punctual versus continuous auction coordination for multi-robot and multi-task topological navigation. *Autonomous Robots, 40*(4), 599–613. https://doi.org/10.1007/s10514-015-9483-7.

Obitko, M., & Marik, V. (2002). Ontologies for multi-agent systems in manufacturing domain. In *Proceedings. 13th international workshop on database and expert systems applications, 2–6 Sept. 2002* (pp. 597–602). https://doi.org/10.1109/dexa.2002.1045963.

Parker, J. E. (2013). Task allocation for multi-agent systems in dynamic environments. In *Paper presented at the Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, St. Paul, MN, USA.

Pippin, C., Christensen, H., & Weiss, L. (2013). Performance based task assignment in multi-robot patrolling. In *Paper presented at the proceedings of the 28th annual ACM symposium on applied computing*, Coimbra, Portugal.

Portugal, D., & Rocha, R. P. (2016). Cooperative multi-robot patrol with Bayesian learning. *Autonomous Robots, 40*(5), 929–953. https://doi.org/10.1007/s10514-015-9503-7.

Research, S. C. f. B. I. (2017). A free, open-source ontology editor and framework for building intelligent systems. https://protege.stanford.edu/.

Sheng, Y., & Gao, Y. (2016). Shortest path problem of uncertain random network. *Computers & Industrial Engineering, 99,* 97–105. https://doi.org/10.1016/j.cie.2016.07.011.

Su, H.-H., Su, L., Dornhaus, A., & Lynch, N. (2017). Ant-inspired dynamic task allocation via gossiping. In *5th workshop on biological distributed algorithms (BDA 2017)*, Washington, DC (in press).

Tenorth, M. M. (2011). *Knowledge processing for autonomous robots*. Doctoral dissertation, Technische Universität München.

UNIHB. (2016). Deliverable D5.2: Technical report/publications on knowledge-base realization. *Sherpa: Smart collaboration between humans and ground-aerial robots for improving rescuing activities in Alpine environments*.

Ure, N. K., Omidshafiei, S., Lopez, B. T., Agha-Mohammadi, A. A., How, J. P., & Vian, J. (2015). Online heterogeneous multiagent learning under limited communication with applications to forest fire management. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), Sept. 28 2015–Oct. 2 2015* (pp. 5181–5188). https://doi.org/10.1109/iros.2015.7354107.

Wei, C., Hindriks, K. V., & Jonker, C. M. (2016). Dynamic task allocation for multi-robot search and retrieval tasks. *Applied Intelligence, 45*(2), 383–401. https://doi.org/10.1007/s10489-016-0771-5.

Wicke, D., Freelan, D., & Luke, S. (2015). Bounty hunters and multiagent task allocation. In *Paper presented at the proceedings of the 2015 international conference on autonomous agents and multi-agent systems*, Istanbul, Turkey.

Wurm, K. M., Dornhege, C., Nebel, B., Burgard, W., & Stachniss, C. (2013). Coordinating heterogeneous teams of robots using temporal symbolic planning. *Autonomous Robots, 34*(4), 277–294. https://doi.org/10.1007/s10514-012-9320-1.

Zhang, L., Zhong, H., & Nof, S. Y. (2015). Adaptive fuzzy collaborative task assignment for heterogeneous multirobot systems. *International Journal of Intelligent Systems, 30*(6), 731–762. https://doi.org/10.1002/int.21725.

Dr. Hebah ElGibreen is an assistant professor at the Department of Information Technology at King Saud University; and an MIT fellow under MRL lab. She received her M.Sc. and Ph.D. degrees from King Saud University in 2009 and 2015, respectively. During her graduate studies, Dr. Hebah has developed a strong interest in artificial intelligence and Machine Learning. She started her career in 2007 by working as a teacher assistant at NXT LEGO robot program for gifted girls, sponsored by "King Abdul-Aziz & His Companions Foundation for the Gifted" program. She started to work in academia from 2008, where she played different roles. In 2016, she was granted a fellowship position at MIT funded by Saudi Aramco in partnership with the Center for Clean Water and Clean Energy at MIT. She has participated in a number of conferences and conducted different training sessions and workshops. She published many conferences and journals papers and was awarded for her work, where she won different awards on the academic and professional level. In 2007, she won the 2nd place of "educating by computers" track in the National Competition in Computer Skills, sponsored by College of Telecom and Information. In 2014, she got a grant from King Abdulazaiz City For Science and Technology to fund her research. In 2016 she won the third place of the Excellence in Research Activity Contest. She was also awarded as the Best Instructor and was voted to be the Best Academic Adviser at her department in the year of 2017.

Kamal Youcef-Toumi earned his advanced degrees (M.S. 1981 and Sc.D. 1985) in Mechanical Engineering from MIT. His undergraduate degree (B.S. in Mechanical Engineering awarded in 1979) is from the University of Cincinnati. He joined the MIT Mechanical Engineering Department faculty in 1986. He is a Professor of Mechanical Engineering at MIT. He is also the Director of the Center for Clean Water and Clean Energy at MIT and KFUPM, and Director of the Mechatronics Research Laboratory (MIT). Youcef-Toumi's research and teaching interests have focused on design, modeling, simulation, instrumentation, and automatic control systems. The applications have included manufacturing, robotics, automation, metrology and nano/biotechnology. Youcef-Toumi served on many scientific and technical committees in the US and around the world. He has also served as a consultant and conducted research and development for many companies and government agencies. He is interested to use these experiences to contribute to several activities including (1) curriculum (content, teaching methods, and digital learning), (2) hands-on students' experience, (3) innovation and entrepreneurship ecosystem, (4) research and development with industrial focus and sponsorships, and (5) partnerships for establishing leadership in selected areas.