To appear in: International Conference on Autonomous Agents and Multiagent Systems AAMAS 2018, Stockholm, Sweden

Distributed Strategy Adaptation with a Prediction Function in Multi-Agent Task Allocation

Joanna Turner
Loughborough University
Loughborough, UK
joanna.z.turner@gmail.com

Qinggang Meng
Loughborough University
Loughborough, UK
q.meng@lboro.ac.uk

Gerald Schaefer
Loughborough University
Loughborough, UK
gerald.schaefer@ieee.org

Andrea Soltoggio
Loughborough University
Loughborough, UK
a.soltoggio@lboro.ac.uk

ABSTRACT

Coordinating multiple agents to complete a set of tasks under time constraints is a complex problem. Distributed consensus-based task allocation algorithms address this problem without the need for human supervision. With such algorithms, agents add tasks to their own schedule according to specified allocation strategies. Various factors, such as the available resources and number of tasks, may affect the efficiency of a particular allocation strategy. The novel idea we suggest is that each individual agent can predict locally the best task inclusion strategy, based on the limited task assignment information communicated among networked agents. Using supervised classification learning, a function is trained to predict the most appropriate strategy between two well known insertion heuristics. Using the proposed method, agents are shown to correctly predict and select the optimal insertion heuristic to achieve the overall highest number of task allocations. The adaptive agents consistently match the performances of the best non-adaptive agents across a variety of scenarios. This study aims to demonstrate the possibility and potential performance benefits of giving agents greater decision making capabilities to independently adapt the task allocation process in line with the problem of interest.

KEYWORDS

Distributed Task Allocation; Scheduling; Multi-Agent Systems

ACM Reference Format:
Joanna Turner, Qinggang Meng, Gerald Schaefer, and Andrea Soltoggio. 2018. Distributed Strategy Adaptation with a Prediction Function in Multi-Agent Task Allocation. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), Stockholm, Sweden, July 10–15, 2018,* IFAAMAS, 9 pages.

1 INTRODUCTION

Distributed systems, made up of multiple connected agents operating together to achieve some objectives, are receiving increasing attention for a variety of applications [13]. Such systems are best suited to applications in which objectives can be broken down into multiple jobs or tasks, which can be autonomously performed by the different agents in the network. Examples include cloud computing systems, distributed sensor networks, and multi-robot teams. The benefits of a distributed architecture over a centralised one include increased reliability and scalability, as well as the absence of bottlenecks that exist with centralised controllers.

A task allocation problem aims to find a global feasible assignment of tasks to agents while optimising one or more objectives. Distributed consensus-based task allocation algorithms can solve task allocation problems without a centralised controller as agents engage in a cooperative planning process consisting of two phases [3]. In the first phase, an agent constructs a schedule of selected tasks through an internal decision-making process. This process has previously been referred to as a utility function [17], a score function [16], or an objective function [15]. In the second phase, agents communicate bids on their selected task assignments and resolve conflicts by assigning tasks to the agents with the highest bids.

The task allocation problem in this paper falls under the single-task (ST), single robot (SR), time-extended assignment (TA) problem under the Gerkey and Matarić taxonomy [9]. Agents perform one task at a time, and each agent can be assigned multiple tasks that they execute based on a schedule. Travel times, task durations, task deadlines, and fuel constraints are a factor. Finding the optimal solution to this task allocation problem in real-time environments becomes computationally unfeasible as the numbers of tasks and agents grow. In complexity theory, the problem is said to be NP-hard [17]. Heuristics, or rules of thumb, have been devised to seek good enough solutions in a realistic time frame but without a guarantee of optimality. The effectiveness of a given heuristic is dependent on various factors including the constraints and parameters of the problem being solved and the objective being optimised [11].

Current state-of-the-art consensus-based task allocation algorithms incorporate heuristics into agent score functions in order to optimise a given objective. While extensive research has been done in the area of multi-agent learning of optimal policies [2, 21], as far as the authors are aware, consensus-based task allocation algorithms have not been designed to adapt online to changing environmental factors [15, 24]. This paper introduces the novel idea of learning a prediction function and adopting a strategy switching behaviour that allows agents to independently adapt task allocation strategies in line with changing environmental factors, and boost performance. The learned function is effectively a prediction

Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018), M. Dastani, G. Sukthankar, E. André, S. Koenig (eds.), July 10–15, 2018, Stockholm, Sweden.

mechanism that uses past experience to select which task allocation strategy yields the optimal global task allocation.

The proposed method is tested through a simulated search and rescue scenario. Two heuristics that have been previously shown to perform well in such a scenario are earliest deadline first (EDF) and nearest task first (NTF) [19]. The prediction functions were trained to predict which heuristic, between the two, will yield the most task allocations. The following assumptions are made: an agent does not have knowledge of the time availability of other agents in the system, and does not have knowledge of the decisions made by other agents concerning the optimal heuristic. The input for the prediction function is limited to information about task assignments received locally from networked agents. The reasoning for these choices is to show that the proposed adaptive method can be applied to consensus algorithms by exploiting the communications necessary for consensus, and without requiring any additional information to be communicated among agents. Results showed that for the majority of scenarios tested, the agents were able to predict and switch to the optimal heuristic based on observations of locally communicated task assignments, without a significant impact on the time to convergence. Additionally, results showed that an additional gain in performance could be achieved by enabling the agents to independently adapt their consensus strategy.

The paper is organised as follows: Section 2 defines the task allocation problem of interest, summarises related research, and describes the state-of-the-art consensus-based bundle algorithm (CBBA) algorithm. Section 3 introduces the proposed adaptive approach as an extension of CBBA, and the method used to test its performance. Section 4 illustrates the results that compare the proposed approach against baseline CBBA. Section 5 summarises the findings and proposes future directions.

2 PROBLEM AND EXISTING METHODS

Given a team of n agents and m tasks, the problem of interest is to allocate tasks to agents with the following assumptions: agents autonomously decide which tasks to take on using a scoring function that computes a score for that agent to perform a certain task. These score functions often incorporate heuristics designed to optimise a specified objective. Agents then communicate with each other to reach consensus on which agents take which tasks. To do so, agents place bids on their selected tasks, share the bids by communicating with each other, and the agent with the highest bid wins the task. Agents co-operate to maximise the number of allocated tasks and to reach an agreed allocation (consensus). Tasks and agents are subject to time constraints.

Formally, $\mathbf{V} = [v_1, \ldots, v_n]$ and $\mathbf{T} = [t_1, \ldots, t_m]$ represent the set of n agents and m tasks, respectively. Each agent $v_i \in \mathbf{V}$ is initalised with:

- A path $\mathbf{p}_i$ of tasks assigned to v_i in the order in which v_i will execute those tasks.
- A winning agent list $\mathbf{z}_i = [z_{i1}, \ldots, z_{im}]$ where an element z_{ik} stores the index of the agent who has won the task t_k according to the latest communication received by v_i. If v_i has not received or made a bid on t_k, then $z_{ik} = 0$.
- A winning bid list $\mathbf{y}_i = [y_{i1}, \ldots, y_{im}]$ where an element y_{ik} stores the winning bid for t_k corresponding to the winner z_{ik}. If there is no bid for task t_k, then $y_{ik} = 0$. Bids on tasks are greater than 0 and less than or equal to $MaxBid$.

2.1 Problem Constraints

Each agent has a maximum operating time f_i, which is the latest time at which v_i can arrive at a task t_k before running out of fuel. Each task t_k has a latest start time ξ_k after which the task expires. The predicted time of execution of $t_k \in \mathbf{p}_i$ by v_i is ς_{ik}. This time includes the duration of earlier tasks in $\mathbf{p}_i$ and travel time to and from those earlier tasks. Thus,

$$\varsigma_{i,k} \leq min(\xi_k, f_i) \quad . \tag{1}$$

Due to these time constraints, it may not be possible to assign all tasks. If a task is not already in $\mathbf{p}_i$ and satisfies the time constraints, it is a *candidate task* and can be considered for inclusion.

Agents communicate with each other via links determined by a network topology. This topology may be restricted, e.g. by communication range.

2.2 Objective Function

The primary global objective $J^\star$ for the problem of interest is to maximise the number of allocated tasks, formally defined as

$$J^\star = \max \left\{ \sum_{i=1}^{n} |\mathbf{p}_i| \right\} \tag{2}$$

$$\text{s.t.} \quad \mathbf{p}_i \cap \mathbf{p}_j = \emptyset, \text{where} \quad i \neq j \quad , \tag{3}$$

and $|\mathbf{p}_i|$ is the number of tasks in $\mathbf{p}_i$. The constraint states that a task may be assigned to one agent's task list at most.

2.3 CBBA and Extensions

The consensus-based bundle algorithm (CBBA) is a distributed multi-agent multi-assignment algorithm [3]. CBBA iterates over the following two phases:

(1) The *task inclusion phase*: each agent greedily builds up a path (or schedule) through a repeating process of computing scores for each candidate task and selecting the task with the highest score to add to their path.
(2) The *consensus phase*: agents communicate $\mathbf{z}_i$ and $\mathbf{y}_i$ to neighbouring agents i.e. those with communication links based on a network topology. When there are conflicting assignments, the highest bid wins and losing agents remove the task from their path [3].

These two phases alternate until consensus has been reached by the team on all task assignments. The number of times the two phases repeat until all agents reach consensus determines the time to convergence.

Notable extensions of CBBA include the following: Choi et al. [4] address heterogeneous networks, and tasks that need to be serviced by multiple robots; Ponda et al. [22] address scenarios with time constraints by incorporating time windows of validity on tasks as part of the scoring scheme; Johnson et al. [14] extend the CBBA with an asynchronous communication protocol to enable agents to run the *consensus phase* on their own schedule; and BW-CBBA [16] that addresses the limitations of utilising submodular score functions to rank tasks within an agent's internal decision

making process. To address uncertainties such as unknown task durations, a robust extension of CBBA is presented in [23] that embeds expected-value and worst-case stochastic metrics into the framework.

2.4 Multi-Agent Learning: Related Work

Learning and adaptation in multi-agent systems is an established research field [25, 26, 30]. A commonly used approach to learning in multi-agent systems is reinforcement learning (MARL), in which agents learn actions and policies through trial and error from a feedback of rewards and punishment. Extensive research has been done in this area. In early research, Littman [18] proposed a Markov games framework for MARL that allows for multiple adaptive agents with conflicting goals. Tan [27] investigated whether agents that learn cooperatively outperform agents that do not. The study showed that sharing learned policies could speed up learning with a cost in communication. In more recent work, Garland and Alterman [8] developed distributed learning techniques to improve coordination among agents. By learning from past experiences of successful cooperation with other agents, and by learning probabilities of individual actions succeeding, agents were able to more efficiently solve coordination problems. Empirical results demonstrated that distributed learning of individual agents improved performance of the whole system, including costs of communication and planning. Hu et al. [12] proposed knowledge transfer mechanisms to demonstrate how knowledge of individually learned policies can be utilised to learn better joint policies. The study exploited sparse interactions in multi-agent systems to improve the performance of multi-agent reinforcement learning. Panait and Luke [21] provide a comprehensive survey of MARL, as well as evolutionary learning, for cooperative teams.

Reinforcement learning has recently been applied to applications with centralised task allocation architectures, such as cloud computing [20], where a scheduler that handles scheduling for multiple resources uses reinforcement learning to learn the best policies to reduce execution time. Gombolay et al. [10] proposed a method to automatically learn scheduling heuristics from expert demonstrations using inverse reinforcement learning for a centralised scheduler.

Building on the evidence of the utility of learning in multi-agent systems, but as opposed to previous studies, we focus in particular on fully distributed consensus-based task allocation algorithms with the aim to integrate learning and decision making into established algorithms such as CBBA. The challenge is to augment consensus-based fully distributed algorithms with increased flexibility to adapt to a large parameter space of scenarios. In fact, the relationship between different task-inclusion heuristics and their effectiveness under different allocation scenarios is not easy to infer a priori. In the following section, we describe how we augment the system to implement distributed strategy adaptation.

3 LEARNING STRATEGY ADAPTATION

This section introduces the proposed adaptive approach for consensus-based task allocation that enables agents to individually predict and select the best task inclusion strategy with the aim to automatically maximise task allocation.

3.1 Heuristic Strategies

In task allocation problems that require agents to execute multiple tasks, the heuristic with which agents include tasks into their schedules is key to optimising allocations. The appropriateness of any heuristic varies as conditions change, such as the number of tasks to agents, the time constraints, and the travel times between tasks. The two heuristics used in this study are earliest-deadline-first (EDF), and nearest-task-first (NTF) [19]. With EDF, agents prioritise tasks with the earliest deadline to include into their schedules, while with NTF, agents prioritise tasks that are nearest to the previous location in their schedule. In a standard environmental setting (see Section 3.6 for an example), EDF allocates more tasks than NTF under conditions with relatively few tasks per agent. As the number of tasks per agent increases, the travel time between tasks becomes a greater factor and eventually NTF allocates the most tasks. As a result, traditional consensus-based task allocation algorithms perform often sub-optimally because there is not a single strategy that works well in all scenarios.

In this study, the key idea is that optimal strategies can be inferred online, i.e., during execution, and locally, i.e., in a distributed fashion for each agent. Thus, the information exchanged by the agents to reach consensus can be exploited to implement a distributed adaptive system. We devised a decision-making mechanism in which agents use the local information available to predict the best heuristics online. The process is shown to result result in the optimisation of the number of allocated tasks under a variety of different conditions.

To infer a rule connecting the local observations made by an agent and the appropriate heuristic, we use supervised classification learning. With supervised learning, the learning algorithm uses labeled training data to infer a general rule or function that maps inputs to outputs. In this study, the input is an agent's observation following a consensus phase, and the prediction is the heuristic that will yield the highest number of allocated tasks overall.

The proposed method exploits locally available task assignment information that is necessary for consensus. Agents are able to resolve conflicting assignments through sharing information about which agents are assigned to which tasks [3]. Thus, the proposed method can be integrated into consensus-based algorithms without additional communication overhead. We describe the implementation of the proposed method as an extension to CBBA.

3.2 Agent Observations

As described in Section 2.3, reaching consensus requires that agents exchange the list $\mathbf{z}$ of agent-task allocations. The list $\mathbf{z}_i$ corresponds to agent v_i's local knowledge of the current global task allocation. From these communications v_i can make the following local observations:

- The set of assigned tasks: $\mathbf{a} = \{k \in \mathbf{z}_i \mid \mathbf{z}_{ik} > 0\}$ and the set of unassigned tasks: $\bar{\mathbf{a}} = \{k \in \mathbf{z}_i \mid \mathbf{z}_{ik} = 0\}$. The cardinalities $|\mathbf{a}|$ and $|\bar{\mathbf{a}}|$ denote the total numbers of assigned tasks and unassigned tasks respectively.
- The set of tasks assigned to other agents not including tasks assigned to v_i is defined as: $\mathbf{o} = \{k \in \mathbf{z}_i \mid \mathbf{z}_{ik} > 0 \wedge \mathbf{z}_{ik} \neq i\}$, where $|\mathbf{o}|$ denotes the cardinality of $\mathbf{o}$.

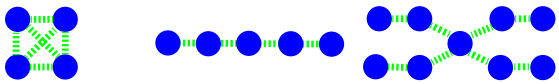
When accounting for heterogeneous agents with different capabilities to perform different tasks, the observations refer to the tasks that v_i is capable of performing i.e. $|\mathbf{a}|$ denotes the number of compatible assigned tasks. We refer to the total number of compatible tasks as m_c. In summary, each agent can derive the following information from received communications: the number of assigned and unassigned tasks, the number of tasks assigned to other agents, and the total number of compatible tasks. This information is used to predict which allocation strategy is more likely to perform better as explained in the following sections. It is worth noting that additional information can be derived and used for predictions. We focus on the set described above as the most informative for the problem of interest, and allow for the possibility of extending the set of inputs in future work.

3.3 Learning Systems

The main focus of this study is the integration of learning and decision making into CBBA to prove that, within the established framework of such an algorithm, appropriate predictions and decisions can be made. Thus, off-the-shelf supervised learning algorithms were used with default parameters to implement the prediction function. It is important to note that learning the prediction function is performed centrally and offline, while the adaptation of the strategy, using the learned function, is performed online and in a distributed fashion. We used two popular supervised learning methods: support vector machine (SVM) and neural network (NN). The proposed adaptive method using SVM and NN are referred to as CBBA$^{+}{}_{\text{SVM}}$ and CBBA$^{+}{}_{\text{NN}}$, respectively. The SVM model used a radial basis function kernel with the MATLAB function for binary classification *fitcsvm*. Using MATLAB's Neural Pattern Recognition toolbox, the network was trained using scaled conjugate gradient backpropagation and had a single hidden layer with 10 nodes. The inputs used for training corresponded to the observation: $[|\mathbf{o}|, |\bar{\mathbf{a}}|]/m_c$. Two outputs corresponded to the classification predictions of which strategy (between EDF and NTF) led to the most tasks being allocated in previous task allocation experiments with non-adaptive strategies. The SVM model returns 0 or 1 corresponding to the classification prediction of an observation. The neural network returns a real number between 0 and 1, indicating the confidence in the classification prediction, where an output of 1 indicates the highest confidence in the classification, and an output of 0 indicates the lowest confidence in the classification.

3.4 Distributed Strategy Adaptation

The task allocation algorithm with the added prediction function is shown with pseudocode in algorithm 1. Before the task allocation procedure begins, each agent is initialised with an index h that determines with which heuristic the agent includes tasks into its schedule (line 3). This initialisation can be done through a uniform random assignment. Once the task allocation procedure is in progress, predictions can be made using locally received information about task assignments. The Predict function (line 5) uses $\mathbf{z}_i$ to make a prediction on the optimal heuristic and returns a heuristic index h. This index is passed to the Task Inclusion Phase (line 6) where v_i includes tasks into its schedule according to the heuristic corresponding to h. During the consensus phase (line 7), $\mathbf{z}_i$ is updated.



(a) Fully connected (b) Row connected (c) Star connected

Figure 1: Network topologies: agents are represented as circles and communication connections are represented as dashed lines between agents. Different topologies affect the timing at which each agent acquires information on allocations. Thus, the decision making capabilities of the agents may be affected by different topologies.

The prediction function is shown with pseudocode in algorithm 2. Applying a limit to the number of times that an agent can switch functions is fundamental to maintain the guarantee of convergence of the algorithm (see [3] for details on convergence). Therefore, a condition *SwitchCondition* on line 3 determines whether a prediction can be made and therefore whether the heuristic can be changed or not. In this study, we set that limit to 1 to test the basic concept that one single switch of strategy is sufficient to increase the overall number of allocated tasks. For a real-time system operating in a dynamic environment, in which agents converge locally rather than as a group [14], a refractory period could be implemented to allow for multiple switches over time with a delay in between. To ensure that the agent has sufficient information to make a prediction, the *SwitchCondition* applied in this study requires that the agent has received task assignment information from other agents, such that: $|\mathbf{o}| > 0$. If the condition returns false then the agent's current heuristic index is returned. Before computing a prediction, the input for the prediction function is normalised by dividing it by the number of tasks: $input = [|\mathbf{o}|, |\bar{\mathbf{a}}|]/m_c$ (line 4). The output computed by the prediction function $f_{predict}$ is evaluated to determine which heuristic is likely to generate the optimal task allocation. If the predicted heuristic is different from the agent's current heuristic, then the agent unassigns all tasks previously assigned to itself so that it can rebuild its schedule with the predicted optimal heuristic (lines 7-11). CBBA's task inclusion phase (see [3]) is the algorithm's point of highest time complexity, consisting of three nested loops. As the proposed adaptive strategy function runs outside of the task inclusion phase and does not require loops, the algorithm's time complexity is unaffected.

Agents communicate task allocations according to a network topology, which impacts the task assignment information that an agent holds at any given time. Different topologies result in different delays to the agents receiving sufficient information for making an accurate prediction. Early predictions may not be meaningful and may benefit from being delayed until more task assignment information is gathered from the rest of the network. Figure 1 illustrates examples of common topologies. With a fully connected topology (Figure 1(a)), each agent receives information about all other agents' task assignments at every communication round. With a row topology (Figure 1(b)), it may take many rounds of CBBA for an agent's allocations to be propagated through the network.

Considering the possible delays in receiving sufficient information, and the requirement to limit the number of times an agent switches heuristic, we apply basic rules to ensure that the adaptive system is able to work under such constraints. After the training phase, the NN gives an output of 0.5 when no strategy has a clear advantage over the other. A question is, how much more advantageous a strategy needs to be to trigger a switch? Given the incompleteness of the information available to agents through local communication, it is reasonable to assume that a strategy prediction requires a confidence margin to signal that switching is advantageous. A ROC (receiver operating characteristic) curve shows graphically the true positive rate as a function of the false positive rate for different cut-off points. A ROC analysis therefore can provide experimental evidence to estimate a confidence parameter so that switching occurs with a desired probability. In the proof-of-concept presented in this study, we limit the investigation by simply increasing the threshold to switch for $f_{evaluate}$ in CBBA$^{+}{}_{\text{NN}}$ to an arbitrary value of 0.6. This simple adjustment prevents strategy switching when no clear advantage for one strategy can be inferred. By doing this, we prevent unnecessary strategy switching and maintain a low number of iterations as shown in the analysis later. Further studies could address the tuning of such a parameter to achieve the best compromise between reactive switches and number of iterations to convergence. In other words, introducing this condition effectively results in the agent delaying a decision until there is a sufficient confidence in either heuristic. Similarly, we adapted the decision system to operate with predictions from the SVM: in this case, CBBA$^{+}{}_{\text{SVM}}$ agents can perform a switch only when $T > 5$, so that each agent has received task allocation information from multiple other agents before switching heuristic in the worst case topology tested. These mechanisms highlight the important fact that decision making in consensus-based algorithms cannot be simply left to a prediction function, but needs to take into consideration the collective multi-agent dynamics. Future studies may investigate further the tuning and implications of different decision making rules.

A factor that affects allocations in consensus-based algorithms is the conflict resolution mechanism. The most common approach to resolving conflicts in consensus task allocation algorithms is to assign tasks to the highest bidder. This process can either happen via an auctioneer [7], or can be fully distributed as with CBBA [3]. Variations of this process exist to account for different problem constraints [1, 4–6]. A second mechanism consists of utilising relative ranking among agents [28]. To assess how well the proposed approach generalises with different conflict resolution strategies, we tested both the bid-based and the rank-based conflict resolution procedures. The implementation of rank-based conflict resolution is easily performed thanks to the tie-breaking heuristic based on agents' unique identification numbers [3] built into CBBA. If all agents place bids of the same value, all conflicts are resolved based on the agents' IDs, which can be thought of as the agents' rank. Agents place all bids equal to the constant *MaxBid* for Rank-based conflict resolution.

Algorithm 1 Task allocation outer-loop iterative procedure with predictive function running on v_i

initialise timer $T \leftarrow 1$
$converged \leftarrow false$
initialise h
while *converged* is $false$ **do**
  h = Predict(h,$\mathbf{z}_i$)
  TaskInclusionPhase(h)
  Consensus Phase
  $converged \leftarrow$ Check Convergence.
  $T \leftarrow T + 1$
end while

3.5 Benchmark Algorithms

The proposed adaptive approach with CBBA$^{+}{}_{\text{NN}}$ and CBBA$^{+}{}_{\text{SVM}}$ is compared to variations of the non-adaptive baseline CBBA:

- CBBA$_{\text{EDF}}$ - all agents use EDF.
- CBBA$_{\text{NTF}}$ - all agents use NTF.
- CBBA$_{50/50}$ - half of the agents use EDF and half use NTF.

CBBA resolves conflicting task assignments by assigning tasks to the highest bidder. For each of these algorithms, agents place bids to the value determined by the score function using NTF. Thus, conflicting task assignments are resolved based on which agent can reach the task fastest from the previous location in their schedule.

3.6 Preparation of Dataset

A simulated search and rescue scenario is used to test the performance of the algorithms, with a rescue team equally split into two agent types with different functions. The scenarios in this paper are based on the established environment types described in [29]. One agent type provides medicine, the other provides food. The survivors are likewise equally split into those requiring food and those requiring medicine. The task allocations for these two job types are solved independently, but require agents of both types to contribute in message passing and to resolve conflicts. The scenario specifications are summarised in Table 1. The task locations are uniformly distributed within a 3D space, while the agents' starting positions are uniformly distributed on the 2D ground space. The deadlines for starting each rescue and the battery limits for each

Algorithm 2 Prediction function for optimal task inclusion strategy running on v_i

function PREDICT(h_{curr},$\mathbf{z}_i$)
  Compute $|\mathbf{o}|$,$|\bar{\mathbf{a}}|$ from $\mathbf{z}_i$
  if *SwitchCondition* is *true* **then**
    $input = [|\mathbf{o}|, |\bar{\mathbf{a}}|]/m_c$
    $output = f_{predict}(input)$
    $h_{new} = f_{evaluate}(output)$
    if $(h_{curr} \neq h_{new})$ **then**
      Empty $\mathbf{p}_i$
      Set all $\mathbf{z}_{ik} = i$ to $\mathbf{z}_{ik} = 0$
      $h_{curr} = h_{new}$
    end if
  end if
  return h_{curr}
end function

agent are uniformly distributed. Given the random initialisation of task and agent locations and deadlines, it is sometimes impossible for some tasks to be started by any agent before its deadline.

The training set is generated by running task allocation experiments under various configurations. The task and agent numbers were selected to cover a range from under-constrained to over-constrained. Over-constrained signifies that there are a greater number of tasks than can be assigned given the time constraints, while under-constrained signifies that there is enough capacity to assign all tasks. Each observation was labeled corresponding to whether CBBA_{EDF} or CBBA_{NTF} yielded the highest number of allocated tasks overall at the time of convergence. Under a star communication network topology, the number of agents was fixed at: 14, and the numbers of tasks were: 84, 112, 140, 168, 196, 266. Under a fully connected communication network topology, the numbers of agents were: 4, 6, 8, 10, 12, 14, 16 and the number of tasks was fixed at: 130. To add variation, this latter setup was repeated with both agent types able to service both task types. The increase in number of tasks and agents were arbitrarily selected within a range to cover a variety of tasks to agent ratios, from under-constrained to over-constrained. Each setup was run 50 times with the same configuration but different initial conditions.

From simulations running these configurations with CBBA_{EDF} and CBBA_{NTF}, the observations: $[|\mathbf{o}|, |\bar{\mathbf{a}}|]/\ m_c$, were taken from each agent at each iteration starting from $T = 2$ to the time of convergence. Given the high number of agents deployed in one scenario and the repetition of scenarios, input vectors $[|\mathbf{o}|, |\bar{\mathbf{a}}|]/\ m_c$ with identical values and labels may be observed in the dataset. Such data points are effectively duplicates and can be safely removed from the dataset. After removal of duplicates, the labeled data set consisted of approximately 6000 unique observations. Cases for which the two heuristics were equivalent were left in.

4 PERFORMANCE ANALYSIS

The simulation results compare the performances of the different algorithms with respect to average task allocations and iterations until convergence at the end of the task allocation process. In real-time systems, the time to reach a solution may be critical to successfully completing the mission. Thus, our analysis also investigates whether strategy adaptation allows the system to converge to a solution within similar time to non-adaptive algorithms. The total iterations for one simulation is determined by the last time an allocation change was made by any agent, either through inclusion or removal. A marginal increase in average execution time per iteration is expected with the adaptive strategies compared to the non-adaptive algorithms. In real-time settings, variable factors that depend on the specific implementation, such as the time required for communication, the processing speed, the number of tasks, the number of times the agent attempts to make a prediction, are all factors that may impact the proportional increase in average execution time. These points are worth investigating in future work to evaluate the trade-off. Results are shown as averages over 50 runs.

Table 1: Scenario Specification

	Medicine	Food
Agent Speed	30m/s	50m/s
Agent Battery	Between 2500 and 5000 seconds	
Agent Start Position	10 000m x 10 000m x 0m ground space	
Task Duration	300 seconds	350 seconds
Task Deadline	Between 0 and 5000 seconds	
Task Location	10 000m x 10 000m x 1000m 3D space	

4.1 Unseen Row Topology, Task Numbers, and Rank Conflict Resolution

This section shows the results of tests comparing the algorithms operating with 14 agents under conditions not seen in training: under a row topology, with different task numbers, and a different conflict resolution strategy. In Figure 2(a), the proposed $\text{CBBA}^{+}{}_{\text{NN}}$ and $\text{CBBA}^{+}{}_{\text{SVM}}$ both match the best average numbers of allocations achieved by the non-adaptive approaches showing that the agents are correctly predicting and selecting the optimal heuristic under different conditions. The number of iterations until convergence are similar for the proposed adaptive approach and the non-adaptive approach that the agents are selecting, indicating that strategy adaptation maintains a similarly low number of iterations as the non-adaptive cases. $\text{CBBA}^{+}{}_{\text{NN}}$ takes marginally longer to converge on average than CBBA_{EDF} and $\text{CBBA}^{+}{}_{\text{SVM}}$ at 130 tasks, but matches the fastest convergence time of CBBA_{NTF} at 220 and 250 tasks. $\text{CBBA}^{+}{}_{\text{SVM}}$ is relatively faster to converge for the lower task numbers and relatively slower for the higher task numbers compared with $\text{CBBA}^{+}{}_{\text{NN}}$.

Figure 2(b) shows the results with all algorithms using the Rank-based conflict resolution strategy, where agents resolve conflicts on task assignments based on agents' ranks. $\text{CBBA}^{+}{}_{\text{NN}}$-Rank still matches the best allocation numbers compared with the non-adaptive approaches, mostly unaffected that Rank consensus was not seen during training. $\text{CBBA}^{+}{}_{\text{SVM}}$-Rank matches the best average numbers of allocations for the lower task numbers, but for the higher numbers shows a drop in performance compared with $\text{CBBA}^{+}{}_{\text{NN}}$-Rank. However, $\text{CBBA}^{+}{}_{\text{SVM}}$-Rank still allocates significantly more tasks on average than CBBA_{EDF}-Rank. The time to convergence for each algorithm is faster overall with Rank consensus, and average time taken is comparable for each algorithm. $\text{CBBA}^{+}{}_{\text{NN}}$-Rank takes at most 2 extra iterations on average than the slowest non-adaptive algorithm, and at best 1 iteration less. $\text{CBBA}^{+}{}_{\text{SVM}}$-Rank is the slowest to converge.

Figure 2(c) shows that when agents use the NTF heuristic in the scenarios with the higher numbers of tasks, the agents allocate more tasks overall on average if combined with Rank conflict resolution. We repeated the experiments for $\text{CBBA}^{+}{}_{\text{NN}}$ with the added condition that if an agent predicts that NTF is the optimal heuristic, it also switches to using Rank conflict resolution. The results are plotted as NN-Switch. Figure 2(c) shows that for the higher number of tasks, NN-Switch benefits from the higher allocations enabled by Rank conflict resolution, as well as the faster convergence time compared with $\text{CBBA}^{+}{}_{\text{NN}}$ and CBBA_{NTF}. For the higher number of tasks, the convergence time for NN-Switch is closest to $\text{CBBA}^{+}{}_{\text{NN}}$-Rank, which has the fastest convergence. In the lower task numbers, NN-Switch benefits from the higher allocations afforded by using bids for conflict resolution, and matches the

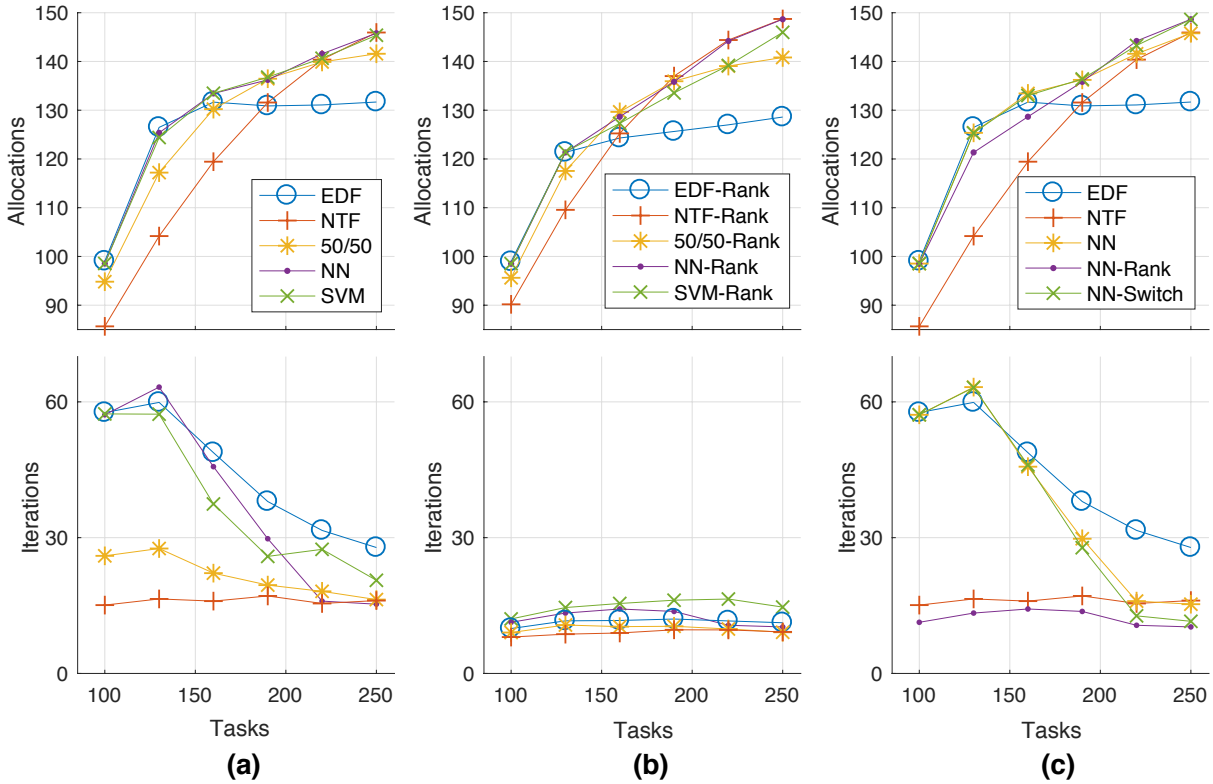


Figure 2: Average task allocations (top) and average iterations until consensus (bottom) for scenarios with different task numbers (100,130,160,190,220,250) and a fixed number of networked agents (14), connected with a row topology. In a) five algorithms are compared: all agents self-assign tasks with the earliest-deadline-first (EDF) heuristic; all agents self-assign tasks with the nearest-task-first (NTF) heuristic, agents are split half and half into using EDF and NTF respectively (50/50); agents are initialised with 50/50 and then optionally switch to EDF or NTF based on a trained neural network (NN) prediction; agents are initialised with 50/50 and then optionally switch to EDF or NTF based on a support vector machine (SVM) prediction. In b) the same algorithms resolve conflicts according to the relative ranking of agents (Rank). In c) with NN-Switch, the task inclusion is as with NN, and conflict resolution switches to Rank if the optimal task inclusion heuristic is predicted to be NTF.

slower convergence times of CBBA$^+_{NN}$ and CBBA$^+_{EDF}$. In these scenarios, the task inclusion strategy and the consensus strategy both affect the performance of the task allocation algorithm.

4.2 Unseen Agent Numbers and Task Numbers

This section shows the results of tests comparing the algorithms operating with different and varying numbers of agents, as well as task numbers unseen in training. Figure 3(a) and Figure 3(b) plot results with a fixed number of agents (10) and different numbers of tasks unseen in training. In Figure 3(a) the topology is fully connected and in Figure 3(b) it is star connected. Under both the full and star topologies, CBBA$^+_{SVM}$ is able to consistently match the average allocations achieved by the best non-adaptive approaches in a comparable convergence time. CBBA$^+_{NN}$ performs marginally less well in this scenario compared with CBBA$^+_{SVM}$. With the fully connected topology, CBBA$^+_{NN}$ falls short of achieving the highest average allocations for 4 out of the 6 task numbers. With the star topology, CBBA$^+_{NN}$ only falls short once when the best non-adaptive algorithm is CBBA$_{50/50}$, which is not used for training. The convergence times of the proposed adaptive approaches are again comparable to the non-adaptive baseline approaches.

Figure 3(c) plots results for simulations with different unseen agent numbers and a fixed number of tasks (130) under a fully connected network. With the higher number of agents (11,13, and 15), CBBA$^+_{NN}$ and CBBA$^+_{SVM}$ perform well in allocating tasks by accurately predicting and selecting the optimal heuristic. The proposed adaptive algorithms perform less well for the lower number of agents (5 and 7). CBBA$^+_{SVM}$ matches CBBA$_{50/50}$ for number of allocations which achieves the second highest average allocations of the non-adaptive approaches, while CBBA$^+_{NN}$ predicts incorrectly that EDF is the optimal heuristic. With 5 and 7 agents, CBBA$^+_{NN}$ and CBBA$^+_{SVM}$ also converge marginally slower than the non-adaptive approaches.

It is worth noting that CBBA$_{50/50}$ performs well in all the tested scenarios and offers good convergence speed. Adjusting the ratio to have more agents using EDF proportionally increases the average number of allocations for the lower task numbers, and reduces the average number of allocations for the higher task numbers. The

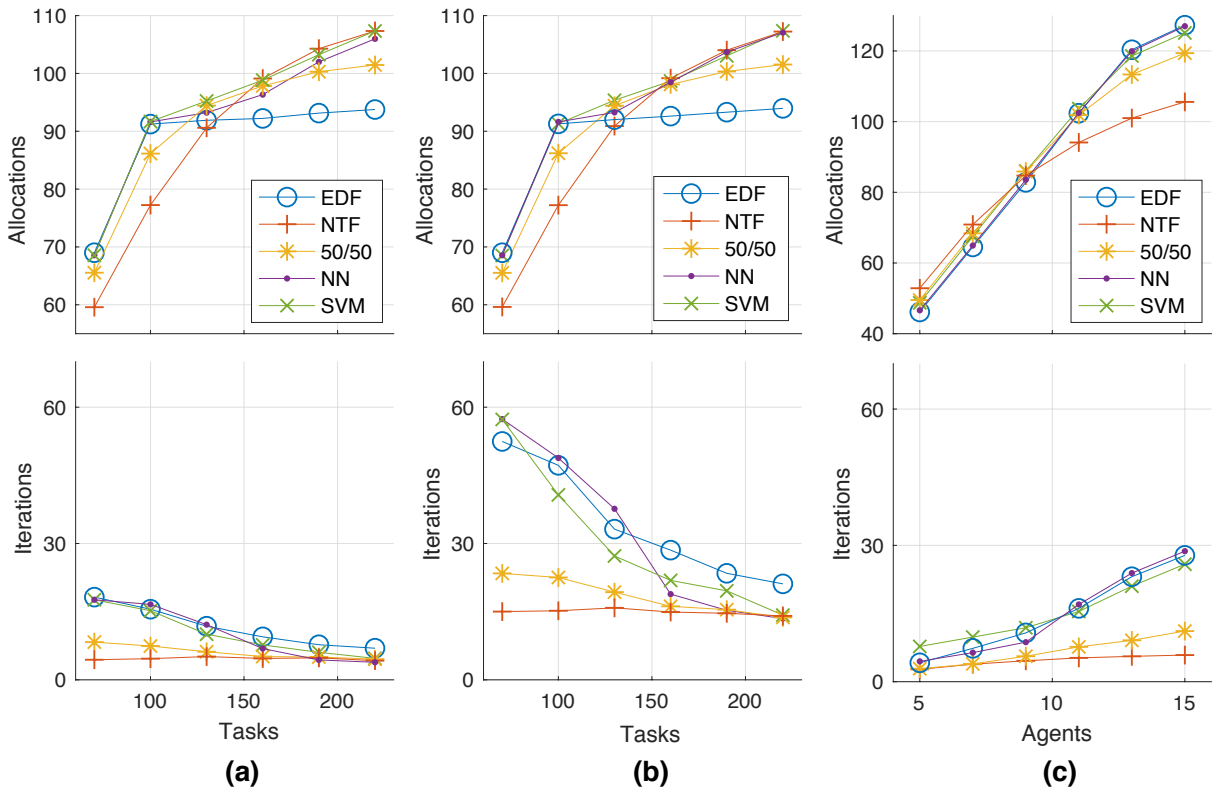


Figure 3: Average task allocations (top) and average iterations until consensus (bottom). In a) and b) task numbers are (70,100,130,160,190,220) with a fixed number of agents (10). In a) agents are connected with a fully connected topology, in b) a star topology. In c) a fixed number of tasks (130), agent numbers are (5,7,9,11,13,15) with a fully connected topology. The algorithms using EDF, NTF, 50/50, NN, and SVM are compared.

inverse holds true when the ratio favours agents using NTF. For simple problems, this static approach is a viable alternative to the proposed approach. The proposed adaptive approach instead offers a proof of concept that can be extended for more complex scenarios. In fact, more sophisticated heuristics can be added or learned to give agents greater adaptability and ability to optimise the task allocation. Such an increase in flexibility and ability to optimise the task allocation would justify the use of the proposed adaptive approach compared to a static approach.

5 CONCLUSIONS

This study investigated the possibility and potential performance gain of enabling distributed agents to independently adapt their task allocation strategies according to locally received information. An adaptive distributed approach is proposed that combines a prediction function with a decision making capability to select the predicted optimal strategy. Results showed that in the majority of scenarios tested, a performance gain was achieved by using the proposed approach. Agents were able to predict and select the optimal task inclusion heuristic to optimise the number of allocated tasks. In a minority of cases tested, when the number of agents was lowest, the agents predicted the incorrect heuristic. However, this resulted in a performance no worse than the non-adaptive strategy. Preliminary results showed that agents could further optimise the task allocation by adapting their conflict resolution strategy.

Factors such as the training data, the inputs, the machine learning tool, and the time of the prediction, are all factors that may impact the accuracy of the predictions, and are therefore interesting points to consider more deeply in future work. The proposed method could be extended to support a greater number of heuristics. For problems of greater complexity, additional inputs could be tested for increased accuracy. Additional inputs may include the number of tasks an agent removes during a round due to conflicts. Furthermore, the proposed approach could be adapted to support agents in learning the best strategy online, as well as adapting to changing optimisation objectives.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] Giulio Binetti, David Naso, and Biagio Turchiano. 2013. Decentralized task allocation for surveillance systems with critical tasks. *Robotics and Autonomous Systems* 61, 12 (2013), 1653–1664. https://doi.org/10.1016/j.robot.2013.06.007

[2] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38, 2 (2008), 156–172. https://doi.org/10.1109/TSMCC.2007.913919

[3] Han-Lim Choi, Luc Brunet, and Jonathan P. How. 2009. Consensus-based decentralized auctions for robust task allocation. *IEEE Transactions on Robotics* 25, 4 (2009), 912–926.

[4] Han-Lim Choi, Andrew K. Whitten, and Jonathan P. How. 2010. Decentralized task allocation for heterogeneous teams with cooperation constraints. In *American Control Conference (ACC)*. IEEE, 3057–3062.

[5] Rongxin Cui, Ji Guo, and Bo Gao. 2013. Game theory-based negotiation for multiple robots task allocation. *Robotica* 31, 6 (2013), 923–934. https://doi.org/10.1017/S0263574713000192

[6] Donato Di Paola, Andrea Gasparri, David Naso, and Frank L. Lewis. 2014. Decentralized dynamic task planning for heterogeneous robotic networks. *Autonomous Robots* 38 (2014), 31–48. https://doi.org/10.1007/s10514-014-9395-y

[7] M. Bernardine Dias, R. Zlot, N. Kalra, and A. Stentz. 2006. Market-Based Multirobot Coordination: A Survey and Analysis. *Proc. IEEE* 94, 7 (2006), 1257–1270. https://doi.org/10.1109/JPROC.2006.876939

[8] Andrew Garland and Richard Alterman. 2004. Autonomous agents that learn to better coordinate. *Autonomous Agents and Multi-Agent Systems* 8, 3 (2004), 267–301.

[9] Brian P. Gerkey and Maja J. Matarić. 2004. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research* 23, 9 (2004), 939–954. https://doi.org/10.1177/0278364904045564

[10] Matthew Gombolay, Reed Jensen, Jessica Stigile, Sung-Hyun Son, and Julie Shah. 2016. Apprenticeship Scheduling: Learning to Schedule from Human Experts. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, 826–833. http://dl.acm.org/citation.cfm?id=3060621.3060736

[11] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4, 2 (1968), 100–107.

[12] Yujing Hu, Yang Gao, and Bo An. 2015. Learning in Multi-agent Systems with Sparse Interactions by Knowledge Transfer and Game Abstraction. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*. IFAAMAS, 753–761.

[13] Yichuan Jiang. 2016. A survey of task allocation and load balancing in distributed systems. *IEEE Transactions on Parallel and Distributed Systems* 27, 2 (2016), 585–599.

[14] Luke Johnson, Sameera Ponda, Han-Lim Choi, and Jonathan P. How. 2010. Improving the Efficiency of a Decentralized Tasking Algorithm for UAV Teams with Asynchronous Communications. In *AIAA Guidance, Navigation, and Control Conference*. 8421.

[15] Luke B. Johnson, Han-Lim Choi, and Jonathan P. How. 2016. The Role of Information Assumptions in Decentralized Task Allocation: A Tutorial. *IEEE Control Systems* 36, 4 (2016), 45–58.

[16] Luke B. Johnson, Han-Lim Choi, Sameera Ponda, and Jonathan P. How. 2012. Allowing Non-Submodular Score Functions in Distributed Task Allocation. In *51st IEEE Conference on Decision and Control (CDC)*. 4702–4708.

[17] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. 2013. A comprehensive taxonomy for multi-robot task allocation. *The International Journal of Robotics Research* 32, 12 (oct 2013), 1495–1512. https://doi.org/10.1177/0278364913496484

[18] Michael L. Littman. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning (ICML)*. 157–163.

[19] Hakim Mitiche, Dalila Boughaci, and Maria Gini. 2015. Efficient Heuristics for a Time-Extended Multi-Robot Task Allocation Problem. In *First International Conference on New Technologies of Information and Communication (NTIC)*. IEEE, 1–6.

[20] Alexandru Iulian Orhean, Florin Pop, and Ioan Raicu. 2017. New scheduling approach using reinforcement learning for heterogeneous distributed systems. *J. Parallel and Distrib. Comput.* (2017). https://doi.org/10.1016/j.jpdc.2017.05.001

[21] Liviu Panait and Sean Luke. 2005. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems* 11, 3 (2005), 387–434.

[22] Sameera Ponda, Josh Redding, Han-Lim Choi, Jonathan P. How, Matt Vavrina, and John Vian. 2010. Decentralized planning for complex missions with dynamic communication constraints. In *American Control Conference (ACC)*. IEEE, 3998–4003.

[23] Sameera S. Ponda. 2012. *Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams*. Ph.D. Dissertation.

[24] Sameera S. Ponda, Luke B. Johnson, Andrew N. Kopeikin, Han-Lim Choi, and Jonathan P. How. 2012. Distributed planning strategies to ensure network connectivity for dynamic heterogeneous teams. *IEEE Journal on Selected Areas in Communications* 30, 5 (2012), 861–869. https://doi.org/10.1109/JSAC.2012.120603

[25] Peter Stone and Manuela Veloso. 2000. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots* 8, 3 (2000), 345–383.

[26] Milind Tambe. 1997. Towards flexible teamwork. *Journal of artificial intelligence research* 7 (1997), 83–124.

[27] Ming Tan. 1993. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*. 330–337.

[28] Joanna Turner, Qinggang Meng, Gerald Schaefer, and Andrea Soltoggio. 2018. Fast Consensus for Fully Distributed Multi-Agent Task Allocation. In *The 33rd Symposium On Applied Computing (SAC '18)*. ACM/SIGAPP. https://doi.org/10.1145/3167132.3167224

[29] Joanna Turner, Qinggang Meng, Gerald Schaefer, Amanda Whitbrook, and Andrea Soltoggio. 2017. Distributed Task Rescheduling With Time Constraints for the Optimization of Total Task Allocations in a Multirobot System. *IEEE Transactions on Cybernetics* 99 (2017), 1–15.

[30] Gerhard Weiß. 1995. Adaptation and learning in multi-agent systems: Some remarks and a bibliography. In *International Joint Conference on Artificial Intelligence*. Springer, 1–21.