Fast Consensus for Fully Distributed 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

In distributed multi-agent task allocation problems, the time to find a solution and a guarantee of reaching a solution, i.e. an execution plan, is critical to ensure a fast response. The problem is made more difficult by time constraints on tasks and on agents, which may prevent some tasks from being executed. This paper proposes a new distributed consensus-based task allocation algorithm that reduces convergence time with respect to previous methods, i.e. the time required for the network of agents to agree on a task allocation, while maximising the number of allocated tasks. The novel idea is to reduce the time to reach consensus among agents by using a hierarchy or rank-based conflict resolution among agents. Unlike other existing algorithms, this method enables different agents to construct their task schedules using any insertion heuristic, and still guarantee convergence. Simulation results demonstrate that the proposed approach can allocate a greater number of tasks in a shorter time than an established baseline method. Additionally, the analysis delineates dependencies between optimal insertion strategies and number of tasks per agent, providing insights for further optimisation strategies.

CCS CONCEPTS

•**Computing methodologies → Multi-agent planning;** *Multi-agent systems;* •**Theory of computation** → Distributed algorithms;

KEYWORDS

Distributed task allocation, auction-based scheduling, multi-agent systems, heuristic algorithms.

ACM Reference format:
Joanna Turner, Qinggang Meng, Gerald Schaefer, and Andrea Soltoggio. 2018. Fast Consensus for Fully Distributed Multi-Agent Task Allocation. In *Proceedings of SAC 2018: Symposium on Applied Computing , Pau, France, April 9–13, 2018 (SAC 2018),* 8 pages.
DOI: 10.1145/3167132.3167224

1 INTRODUCTION

Distributed multi-agent task allocation algorithms have many application domains such as search and rescue missions [11, 13, 16, 18], product manufacturing [5, 20], exploration [9], support in healthcare facilities [4], surveillance [1], and target tracking [17]. By removing the need to maintain a connection with a central server, distributed systems facilitate greater mission ranges and robustness to failures compared with centralised methods.

This work addresses the problem of minimising the time to convergence and maximising the number of allocated tasks in scenarios with time constraints on tasks and on agents. The task allocation problem considered requires that agents perform one task at a time, and each agent can be assigned multiple tasks that they execute based on a schedule. The predicted cost of an agent performing a task depends on other tasks in that agent's schedule. Using the iTax taxonomy [10], this is known as the single-task (ST), single robot (SR), time-extended assignment (TA) problem with in-schedule dependencies (ID). Finding the optimal solution to this problem quickly becomes computationally unfeasible as the numbers of tasks and agents grow, because of the high computational complexity. In complexity theory, the problem is said to be NP-hard [10]. Thus, solutions are sought by means of heuristic searches that do not guarantee optimality, but find good enough solutions within acceptable time. In highly dynamic environments in which new tasks and agents may come into play during execution, a fast convergence time to a solution is an essential quality of an algorithm [14].

The consensus-based bundle algorithm (CBBA) [2] is a state-of-the-art fully distributed market-based task allocation algorithm designed to provide provably good approximate solutions to the ST-SR-TA problem over networks of heterogeneous agents. CBBA iterates over two main phases [2]. In the first phase, agents construct schedules of selected tasks using a scoring function. In the second phase, agents communicate bids on their selected tasks and resolve conflicting task allocations. The number of times the two phases repeat until all agents reach consensus determines the time to convergence. This is largely determined by the number of conflicting task allocations.

The property of guaranteed convergence is vital as it ensures that agents will at some point reach a solution shared by all agents. CBBA guarantees convergence, provided that the scoring function satisfies certain constraints [2]. These constraints, however, also limit the search capability of the algorithm [6], which as a result may deliver poorer solutions. The authors in [6] address this limitation by introducing the Bid Warped CBBA (BW-CBBA) that removes some of the constraints on score functions while maintaining the property of guaranteed convergence.

In this study, we seek to reduce the convergence time by introducing a decision-making approach to resolving conflicting task allocations that is based exclusively on the agents' ranking in a hierarchy. This method guarantees convergence and removes restrictions on how tasks are included into agents' schedules. As a consequence of the rank-based conflict resolution method introduced in this paper, agents may simultaneously utilise different heuristics to construct their schedules, granting more flexibility to the distributed algorithm. We hypothesise that the proposed approach can reduce convergence time while maintaining the same or a higher number of task allocations.

The performance of the new method is quantified with comparisons with an extension of CBBA, the best known algorithm for these specific types of problems. Measuring the performance in terms of both number of allocated tasks and time to convergence revealed relationships between insertion heuristics and parameters in the problem domain. Such insights permit the design of better search strategies that take into account, e.g., whether scenarios have tight time restrictions and the ratio between tasks and available agents. Additionally, results revealed that agents employing different insertion heuristics from each other can reduce the competition for the same tasks and therefore also reduce convergence time.

2 PROBLEM STATEMENT

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. 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 initialised with the following data structures:

- A bundle $\mathbf{b}_i$ of tasks assigned to v_i ordered chronologically based on when the tasks were added. Newly assigned tasks are appended to the end of the bundle.
- A path $\mathbf{p}_i$, same as $\mathbf{b}_i$, but with tasks in the order in which v_i will execute them.

To select which tasks to add to the bundle, an agent computes a score c_{iq} for each task $t_q \in \mathbf{T}$ using a function $F_{iq}()$. Agents can take on up to L_t tasks. The length of the bundle and path, represented by $|\mathbf{b}_i|$ and $|\mathbf{p}_i|$ respectively, must be therefore less than or equal to L_t.

- A winning agent list $\mathbf{z}_i = [z_{i1}, \ldots, z_{im}]$ where an element z_{iq} stores the index of the agent who has won the task t_q according to the latest communication received by v_i. If v_i has not received or made a bid on t_q, then $z_{iq} = 0$.
- A winning bid list $\mathbf{y}_i = [y_{i1}, \ldots, y_{im}]$ where an element y_{iq} stores the winning bid for t_q corresponding to the winner z_{iq}. If there is no winner for task t_q, then $y_{iq} = 0$.

2.1 Problem Constraints

Agents can perform at most one task at a time, and each agent can be assigned multiple tasks that they execute based on a schedule, with travel times between tasks. Each agent has a maximum operating time f_i, which is the latest time at which v_i can arrive at a task t_q before running out of fuel. Each task t_q has a latest start time ξ_q after which the task expires. The predicted time of execution of $t_q \in \mathbf{p}_i$ by v_i is ς_{iq}. This time includes the duration of earlier tasks in $\mathbf{p}_i$ and travel time to and from those earlier tasks. Thus,

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

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. In dynamic settings, the topology may change and become disconnected when agents move [15]. In this study, the agents are stationary during the task allocation process, the topology remains the same and is connected. Once a plan has been agreed, the agents set off to perform their assigned tasks.

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

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

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

Where $|\mathbf{p}_i|$ denotes the number of tasks in $\mathbf{p}_i$. The constraint states that the tasks in $\mathbf{p}_i$ may not be in any other agents' paths i.e. a task may be assigned to one agent's task list at most.

3 RELATED RESEARCH

3.1 CBBA and Extensions

The consensus-based bundle aglorithm (CBBA) [2] is a distributed multi-agent multi-assignment algorithm. CBBA iterates over the following two phases:

1. The *bundle building phase*: each agent greedily builds up a bundle through a repeating process of computing scores for each candidate task and selecting the task with the highest score to add to their bundle.
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 bundles as well as all tasks that were added to the bundle after that task. If bids are tied then the agent with the lowest index wins the task. [3]

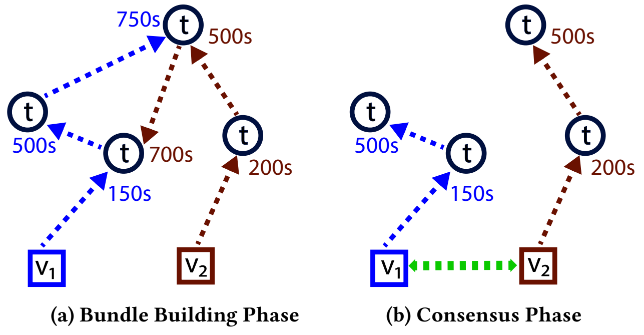


Figure 1: Example of the task allocation process. Dashed arrows represent a travel distance and are labeled with the estimated time to reach a task. Dashed link between agents represents communication. The bundle building phase (a), agents independently include tasks into their bundles to determine which tasks to execute and in which order. The consensus phase (b), agents communicate their task assignments among networked agents and resolve conflicting task assignments.

These two phases are illustrated with an example shown in Figure 1. As the consensus phase results in agents removing tasks from their bundles and creating time in their schedules, the bundle building phase is repeated to attempt to assign more tasks in the free time that is available. The two phases alternate until agents can no longer add tasks into their schedules and consensus has been reached by the team on all task assignments, such that all agents have an identical list $\mathbf{z}_i$. CBBA converges in polynomial time, within $max\{m, L_t n\} \cdot D$ iterations where D is the diameter of the network, provided that the scoring function satisfies diminishing marginal gain (DMG) [2]. DMG means that the score that v_i computes for candidate task t_q, defined as c_{iq}, cannot increase as a result of other tasks being added to the bundle $\mathbf{b}_i$ before t_q [2], such that:

$$c_{iq}(\mathbf{b}_i) \leq c_{iq}(\mathbf{b}_i \oplus_{end} t_z) \quad , \tag{4}$$

where $\oplus_{end}\ t_z$ denotes the append of t_z to the end of $\mathbf{b}_i$. The trade-off of the DMG condition is a possible performance degradation in certain scenarios.

Notable extensions of CBBA include the following: Choi et al. [3] address heterogeneous networks, and tasks that need to be serviced by multiple robots; Ponda et al. [15] address dynamic network topologies and scenarios with time constraints by incorporating time windows of validity on tasks as part of the scoring scheme; Johnson et al. [8] extend CBBA with an asynchronous communication protocol to enable agents to run the *consensus phase* on their own schedule; and BW-CBBA [6, 7] that addresses the limitations of utilising DMG score functions to rank tasks within an agent's internal decision making process.

3.2 Bid Warped CBBA

The Bid Warped CBBA (BW-CBBA) [6] decouples the *scores* that inform task selection in the bundle building phase from the *bids* that are communicated to networked agents. The idea is that the internal score function need not satisfy DMG and the external bids need not be identical to the internal scores. Only the bids that agents share with each other need to satisfy DMG to guarantee convergence. A proof is provided in [6].

3.3 Score Functions

To determine the score of a candidate task t_q, CBBA inserts t_q into $\mathbf{p}_i$ at each index l one at a time. A constraint is that the insertion cannot impact the current start times for the tasks already in the path [15] and - for the implementation in this study - satisfies the time constraints in equation (1). The score is computed at each index l and the highest score is stored as c_{iq}. The score function is defined as:

$$F_{iq}(\mathbf{p}_i \oplus_l t_q) = R_{iql} - C_{iql}, \tag{5}$$

and

$$c_{iq} = \max_l F_{iq}(\mathbf{p}_i \oplus_l t_q), \tag{6}$$

where $\mathbf{p}_i \oplus_l t_q$ denotes the inclusion of t_q into $\mathbf{p}_i$ at index l. R_{iql} denotes the reward and C_{iql} the cost for including t_q into $\mathbf{p}_i$ at index l. If the insertion of t_q cannot meet the constraints at any index in the path, then $c_{iq} = 0$.

BW-CBBA applies a bid warping function to c_{iq} that produces a DMG satisfying score $\bar{c}_{iq}$. The bid warping function G is defined as:

$$\bar{c}_{iq} = G_{iq}(c_{iq}, \mathbf{b}_i) = \min(c_{iq}, \bar{c}_{iq_j}) \quad \forall j \in \{1, \ldots, |\mathbf{b}_i|\} \tag{7}$$

where $\bar{c}_{iq_j}$ is the score of the jth element in the current bundle [6]. In other words, the bid for a candidate task must be lower than, or is made to be equal to, the lowest bid of all other tasks already in agent v_i's bundle.

3.4 Bid Warped CBBA Bundle Building Phase

The bundle building phase of BW-CBBA [6] that runs independently on each agent v_i is summarised in Algorithm 1: For each candidate t_q, v_i computes a score c_{iq} with its internal score function F_{iq} (line 4). A DMG satisfying bid $\bar{c}_{iq}$ is then created with the function G_{iq} (line 5). $\bar{c}_{iq}$ is compared with the current winning bid y_{iq} for

Algorithm 1 CBBA: Bundle Building with Non-DMG Scores [6]

```
1:  procedure BUILD BUNDLE
2:      while $|\mathbf{p}_i| < L_t$ do
3:          for $t_q \in T \setminus \mathbf{p}_i$ do
4:              $c_{iq} = \max_l F_{iq}(\mathbf{p}_i \oplus_l t_q), \quad \forall l \leq |\mathbf{p}_i| + 1$
5:              $\bar{c}_{iq} = G_{iq}(c_{iq}, \mathbf{b}_i)$
6:              $h_{iq} = \Pi(\bar{c}_{iq} > y_{iq})$
7:          end for
8:          $q^\star = \text{argmax}_q c_{iq} \cdot h_{iq}$
9:          if $\bar{c}_{iq^\star} > 0$ then
10:             $z_{iq^\star} = i$
11:             $y_{iq^\star} = \bar{c}_{iq^\star}$
12:             $\mathbf{b}_i \oplus_{end} t_{q^\star}$
13:             $\mathbf{p}_i \oplus_l t_{q^\star}$ where $l$ yielded $c_{iq^\star}$
14:         else
15:             break
16:         end if
17:     end while
18: end procedure
```

t_q. The boolean $h_{iq} = true$ if v_i outbids the current winner (line 6). The candidate task selected to be added to v_i's task list (using task index $q^\star$) is the task that has the highest score c_{iq} that also outbids the current winner (line 8). The new winning agent for $t_{q^\star}$ is set as v_i's index in z_i (line 10). The winning bid for $t_{q^\star}$ is set as $\bar{c}_{iq^\star}$ in y_i. Then, $t_{q^\star}$ is appended to the bundle, and inserted into the path where it yielded the highest score $c_{iq^\star}$. The bundle building phase terminates when no candidate tasks can outbid the current winning bids, or the maximum bundle length is reached.

4 CBBA WITH FAST CONVERGENCE DESIGN

This section introduces the rank-based conflict resolution method that reduces the time to convergence, implemented in this study as a modification to CBBA. The insertion heuristics used in combination with the rank-based conflict resolution to demonstrate the proposed method's performance are also detailed in this section. The proposed method is not limited to CBBA but may be implemented into similar distributed consensus based task allocation algorithms.

4.1 Rank-based Conflict Resolution

In standard task allocation algorithms, bids on task assignments give an indication of the optimality of an assignment with respect to an optimisation objective. When conflicting assignments occur, the agent that can perform the task most optimally keeps the assignment. This process requires that bids be comparable and therefore that agents must share a function to assign scores to their assignments. The novelty in this study is to introduce bids that are invariant to factors such as the agent's path and score function. Constant bids add stability to the convergence process and therefore speed up the rate of convergence. Additionally, this method enables agents to simultaneously use different score functions from each other. As a result of losing information from bids, a trade-off is the possible reduction in quality of the task allocation with respect to the objective being optimised by the score function. In this study we consider the number of allocated tasks and the time to convergence as the highest priority optimisation objectives, and it is therefore worth a possible reduction in optimality of secondary objectives, such as distance covered by the agents. Future work may look at autonomously adapting the task allocation method in line with the most appropriate optimisation objective given the problem domain.

Inbuilt into CBBA's conflict resolution phase is a tie-breaking heuristic based on agent identification numbers [2]. This unique numerical ID is initialised at the outset as the agent's index. When a tie occurs between bids on the same task, the agent with the lowest index wins the task. To implement conflict resolution based on agent ranking requires therefore simply that agents' bids are made to be identical at all times. The modification to the bundle building phase is shown in Algorithm 2 line 5 where a constant bid value defined as *constantBid* is applied to all bids. It is worth noting that a constant bid value satisfies the condition of DMG in equation (4) and therefore preserves CBBA's guarantee of convergence.

A key feature of this approach is that the distribution of rank is transitive i.e. every agent is either dominant or submissive relative to every other agent. As a consequence, agents lose conflicts only

Algorithm 2 CBBA: Bundle Building with EDF and agent rank bidding

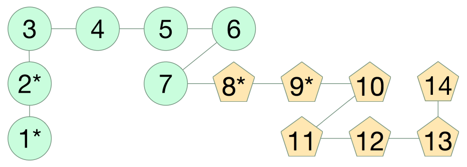
```
1:  procedure BUILD BUNDLE
2:      while $|\mathbf{p}_i| < L_t$ do
3:          for $t_q \in T \setminus \mathbf{p}_i$ do
4:              $c_{iq} = \max_l F_i(\mathbf{p}_i \oplus_l t_q), \quad \forall l \leq |\mathbf{p}_i| + 1$
5:              $\bar{c}_{iq} = constantBid$
6:              $h_{iq} = \Pi(\bar{c}_{iq} > y_{iq})$
7:          end for
8:          $q^\star = \mathrm{argmin}_q \xi_q \cdot h_{iq}, \quad \forall c_{iq} > 0$
9:          if $\xi_{q^\star} > f_i$ then
10:             $q^\star = \mathrm{argmax}_q c_{iq} \cdot h_{iq}$
11:         end if
12:         if $\bar{c}_{iq^\star} > 0$ then
13:             $z_{iq^\star} = i$
14:             $y_{iq^\star} = \bar{c}_{iq^\star}$
15:             $\mathbf{b}_i \oplus_{end} t_{q^\star}$
16:             $\mathbf{p}_i \oplus_l t_{q^\star}$ where $l$ yielded $c_{iq^\star}$
17:         else
18:             break
19:         end if
20:     end while
21: end procedure
```

to agents of higher rank. Consider that the relative rank of each agent matches its index such that v_1 is the highest ranked and v_m the lowest ranked agent. v_1 will win all conflicts on tasks that it selects from $\mathbf{T}$. v_2 will win all conflicts on tasks that it selects from $\mathbf{T} \setminus \mathbf{b}_1$. v_i will win all conflicts on tasks that it selects from $\mathbf{T} \setminus \mathbf{b}_h, \quad \forall h \in \{1, \ldots, i-1\}$. By selecting only tasks that have not been included by higher ranking agents, an agent is ensured to have winning bids, because lower ranking agents cannot challenge that. When there are no more conflicts, the system converges. A network where agents are ranked in topological order, such as in Fig. 2(a), will propagate more efficiently the assignments of higher ranked agents to lower ranked agents such as to reduce the number of conflicts, compared with a network where agents are not ranked in topological order such as in Fig. 2(b).

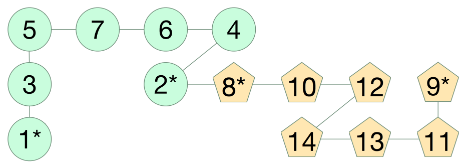
The topology of the network is a determining factor in time to convergence. If agents bidding on the same tasks are not directly connected, such as in Fig. 2(c) where agents of the same type are connected through agents of a different type, it may take many iterations to receive bids on conflicting assignments and therefore longer to resolve conflicts and converge. A finding in section 5 is that the proposed consensus strategy is most effective at reducing convergence time with the ordered row topology (Fig. 2(a)) and least effective with the unordered hybrid topology (Fig. 2(c)).

4.2 Earliest Deadline First Task Inclusion

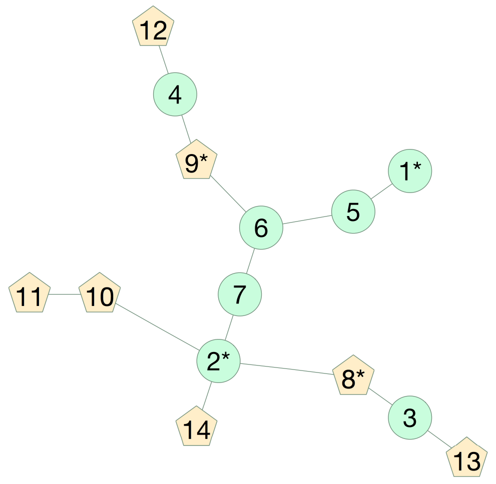
A main benefit of decoupling scores from bids, as shown by BW-CBBA, is the capability to match more closely the agent's internal decision making process to the optimisation objective, while maintaining convergence guarantees. This extension was shown to yield higher quality task allocations than baseline CBBA regardless that the communicated bids were required to be approximated [6].



(a) Ordered Row Topology



(b) Unordered Row Topology



(c) Hybrid Topology

Figure 2: Network topologies that determine the communication links between agents with two agent types (circles and pentagons). Agent indexes correspond to the agents' ranks. Agents 1-7 service medicine tasks and agents 8-14 service food tasks. Agents with or without a star (*) employ different insertion heuristics as explained in section 5.1.

EDF is a well known scheduling algorithm in which tasks with the earliest deadlines are given highest priority. EDF has recently theoretically and empirically been shown to be fast and effective at maximising the number of allocated tasks in a similar scenario [12]. An inclusion strategy such as EDF can cause a high number of conflicts as all agents prioritise tasks in the same order. By applying EDF task inclusion to a subset of agents, with the remaining agents using a different strategy, the number of conflicts can potentially be reduced and therefore speed up convergence compared with all agents using EDF (this is tested in section 5). EDF is implemented on line 8 of Algorithm 2. The best task, with index $q^{\star}$, is selected as the task with the earliest deadline for which h_{iq} evaluates as true.

If the agent's fuel limit is earlier than the earliest deadline, the agent selects the task with the highest score. This condition is added on lines 9–11. A scenario with fuel constraints and no deadlines on tasks is used to further evaluate the performance of the proposed method in section 5.

5 PERFORMANCE ANALYSIS

In this section, the performance of the proposed rank-based conflict resolution is tested and compared using 3 different heuristics against the baseline BW-CBBA that is used as a benchmark. These different combinations are evaluated as a function of the number of iterations until convergence, the number of allocated tasks, and the distance travelled per task. A range of topologies is used to assess these performances since, as described in section 4.1, the topology affects the allocation dynamics. An increasing number of tasks with a fixed number of agents is also used to assess the performance of the algorithms ranging from when the system is 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. A variation in the time constraints is also applied to further demonstrate the performance of the proposed method.

5.1 Assessing Performance

The combinations of the proposed rank-based conflict resolution with three different heuristics, and the benchmark algorithm, are detailed as follows:

(1) **EDF-Rank**: Selecting tasks based on EDF (section 4.2) and rank-based conflict resolution (section 4.1) - (Algorithm 2).
(2) **Score-Rank**: Selecting tasks based on score function (section 3.3) and rank-based conflict resolution (section 4.1). This configuration is Algorithm 2 with lines 8, 9 and 11 removed.
(3) **Mixed-Rank**: Selecting tasks based on either EDF (section 4.2) or score function (section 3.3) and rank-based conflict resolution (section 4.1). This configuration applies EDF task selection to 4 agents and applies task selection based on scores to the other 10 agents.
(4) **Score-Bids**: Selecting tasks based on score function and convergence with varying bids. This configuration is the benchmark Algorithm 1, first introduced in [6].

5.2 Experimental Setup

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 build on the environment types described in [19, 21]. One agent type provides medicine, the other provides food. The survivors are likewise equally split into those requiring food and those requiring medicine. The scenario specifications are summarised in Table 1. The mission takes place in a 3D space. The task locations are uniformly distributed within this 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 on each 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 reward and cost for the scoring function (equation 5) were set as $R = 10\,000$ and $C_{iql} = \Delta D_{iq}(\mathbf{p}_i)/vel_i$ where $\Delta D_{iq}(\mathbf{p}_i)$ is the distance travelled by the agent to reach the candidate task location from its previous location in $\mathbf{p}_i$, and vel_i is the velocity of agent i.

The total iterations for one simulation is expressed as the last iteration number at which an allocation change was made, either through inclusion or removal. The travel distance is represented as the average travelling distance per task for all agents with the final task allocation. The number of agents was fixed at 14 and the number of tasks tested was 84, 112, 140, 168, 196, and 266. These numbers were selected to cover a range from under-constrained to over-constrained. The increase in the number of tasks was arbitrarily selected. The number of agents 14 was selected as the largest number to fit within computer performance limitations. The agent network topologies used are illustrated in Fig. 2. The topology is initialised at the outset and remains constant through the task allocation process. Each setup was run 50 times with the same configuration but different initial conditions. Results are shown as averages over those 50 runs.

5.3 Results

Figure 3 plots the results of Score-Rank, EDF-Rank, Mixed-Rank and the benchmark Score-Bids across the different topologies as a function of the total number of tasks. The trend in the number of allocations is consistent across topologies. The algorithm using EDF allocates the highest average number of tasks for the lower 3 task numbers. In the best case, EDF-Rank allocates 17.4 more tasks on average than Score-Bids with ordered row topology. For all task numbers, Score-Rank allocates more tasks than Score-Bids. In the best case, Score-Rank allocates 8.2 tasks more on average than Score-Bids. Compared with EDF-Rank, The algorithms using Score allocate the most tasks for the highest 2 task numbers. A general trend is that EDF allocates the most tasks when the system is under-constrained i.e. the lower 3 task numbers. When the system is over-constrained, i.e. the higher 3 task numbers, Score allocates the most tasks by a clear margin.

The performances of the algorithms using Rank consistently average at 7 iterations, with below 0.5 standard deviation, at all numbers of tasks with the ordered row topology. In comparison, the benchmark Score-Bids ranges from 13.5 to 17.5 average with between 4 and 5 standard deviation. The average number of iterations for the Rank algorithms increases with the unordered row topology, but remain lower than for Score-Bids. With the hybrid topology, Score-Rank consistently converges in fewer iterations on average than Score-Bids, whereas EDF-Rank converges slower on average than Score-Bids 5 out of 6 times.

With the unordered row and hybrid topologies, Mixed-Rank achieves higher average allocations than the Score algorithms 5 out of 6 times. Similar results are achieved with the ordered row topology. With the unordered row and hybrid topologies, the corresponding average iterations for Mixed-Rank are second lowest. In the best case for the hybrid topology, Mixed-Rank allocates 11.7 tasks more than Score-Bids in 3.9 iterations on average fewer than Score-Bids.

The average travel distances per task are consistent across the three topologies. EDF-Rank gives the highest travel distance by a significant margin, between 3 and 4 times greater than Score-Bids, which achieves the lowest average distance. As might be expected, Mixed-Rank falls between EDF-Rank and Score-Rank proportionally to the split of agents using either heuristic. Interestingly, while Score-Rank gives higher average distances than Score-Bids, there remains a clear advantage towards optimising travel distances by using Score-Rank compared with EDF-Rank. With the higher numbers of tasks, there is not a significant difference in average travel distance between Score-Rank and Score-Bids. These results give an indication of the trade-off for speeding up convergence with a marginal increase in average travel distance, using the proposed method.

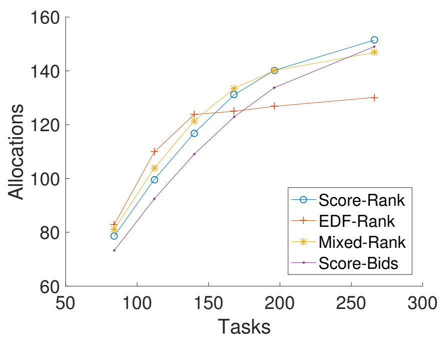
Figure 4 shows the results for the scenario with time constraints on agents without deadlines on tasks, using the unordered row topology. Score-Rank and Score-Bids are compared. The numbers of allocations consistently match for both algorithms. In the best case, Score-Rank converges in less than half the number of iterations compared with Score-Bids. The average travel distance per task is significantly higher with Score-Rank for the lower two task numbers. However, for the higher task numbers, the average travel distance is the same for both Score-Rank and Score-Bids.

Table 1: Scenario Specification

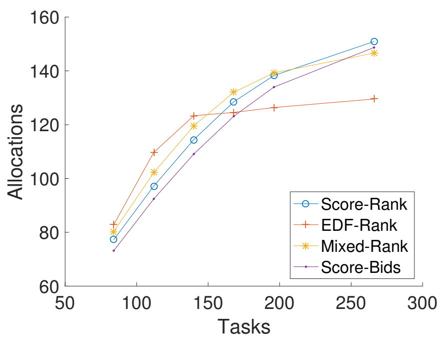
	Medicine	Food
Agent Speed	30m/s	50m/s
Agent Battery	Between 2500 and 5000 seconds	
Agent Start Position	10 000m x 10 000m x 0m ground space	
Task Duration	300 seconds	350 seconds
Task Deadline	Between 0 and 5000 seconds	
Task Location	10 000m x 10 000m x 1000m 3D space	

6 DISCUSSION AND CONCLUSIONS

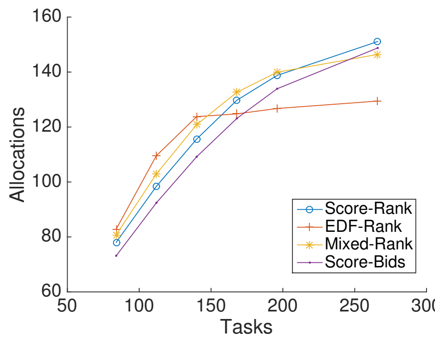
Simulation results showed that the proposed rank-based conflict resolution combined with insertion heuristics proved successful for minimising time to convergence while maximising task allocations. The findings suggest that the proposed approach of rank-based conflict resolution is most effective and can strongly reduce convergence time when agents' ranks are determined by the network topology. Future work may look at a theoretical analysis of the proposed method to formally compare the average and worst case convergence times with previous methods. The proposed method may also be extended to assign agents' ranks based on the network topology. The performance of the proposed method may be further assessed under time-varying topologies. Another result in this study is that fast consensus can be effectively achieved by employing multiple selection strategies across agents. Although intuitive, the proposed experiments showed for the first time in simulation the advantage of such an approach. These results motivate further studies to devise algorithms that can select the appropriate strategy autonomously and accordingly to a dynamically changing number of tasks, number of agents and network connectivity links.



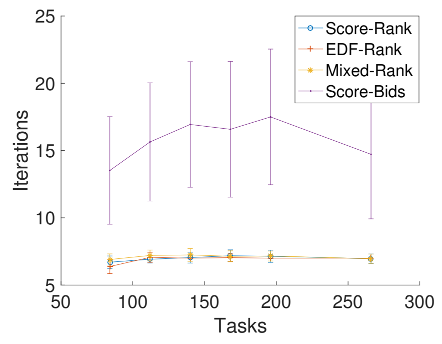
(a) Ordered Row Allocations



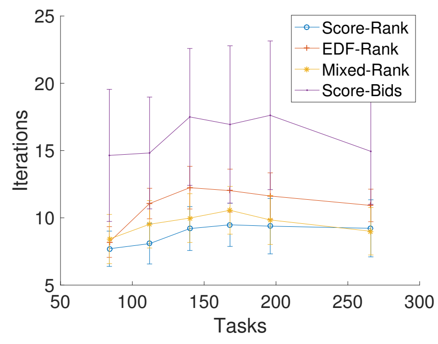
(b) Unordered Row Allocations



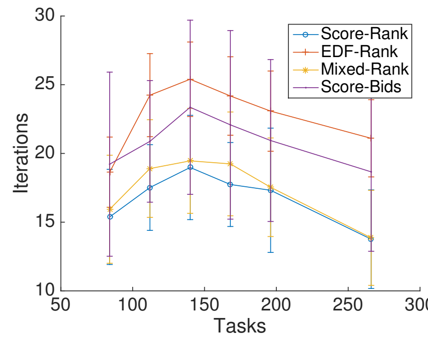
(c) Hybrid Allocations



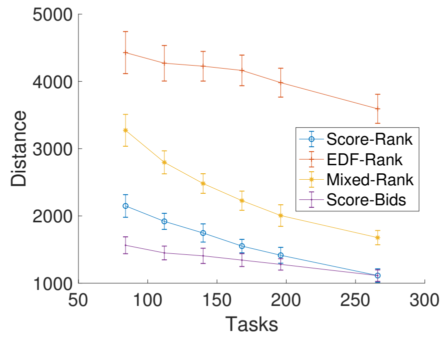
(d) Ordered Row Iterations



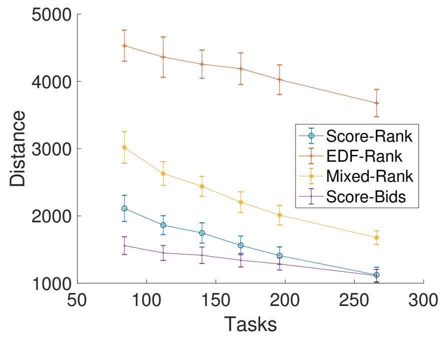
(e) Unordered Row Iterations



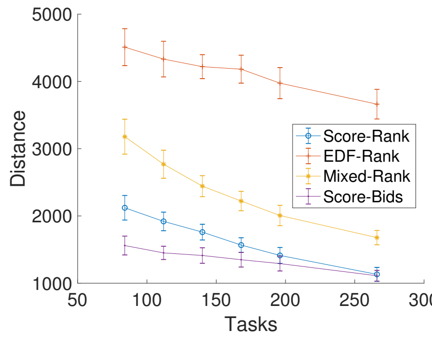
(f) Hybrid Iterations



(g) Ordered Row Travel Distance

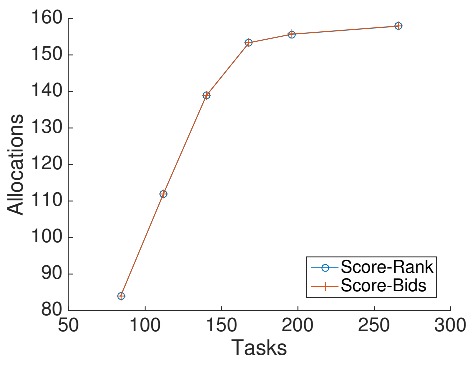


(h) Unordered Row Travel Distance

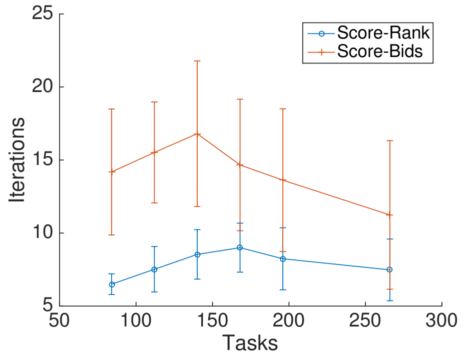


(i) Hybrid Travel Distance

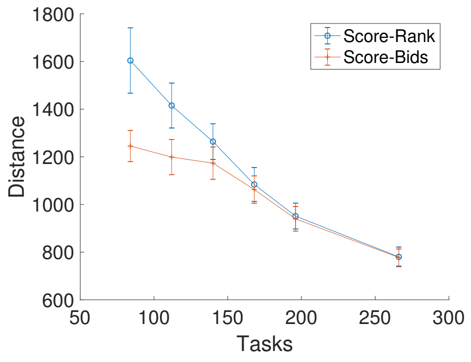
Figure 3: Average allocations, iterations, and travel distance per task across different network topologies in a scenario with time constraints on tasks and on agents. The number of agents is 14 and the numbers of tasks are 84, 112, 140, 168, 196, and 266. The error bars represent standard deviation.



(a) Allocations



(b) Iterations



(c) Travel Distance

Figure 4: Average allocations, iterations, and travel distance per task for a scenario with fuel constraints on agents and without deadlines on tasks using the unordered row topology.

ACKNOWLEDGMENTS

The authors would like to thank Dr. A. Trehan for his insightful feedback. The authors would also 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] 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.
[3] Han-Lim Choi, Andrew K. Whitten, and Jonathan P. How. 2010. Decentralized task allocation for heterogeneous teams with cooperation constraints. *2010 American Control Conference*, 3057–3062.
[4] Gautham P. Das, Thomas M. McGinnity, Sonya A. Coleman, and Laxmidhar Behera. 2014. A Distributed Task Allocation Algorithm for a Multi-Robot System in Healthcare Facilities. *Journal of Intelligent & Robotic Systems* 80, 1 (2014), 33–58. https://doi.org/10.1007/s10846-014-0154-2
[5] Hamed Fazlollahtabar, Mohammad Saidi-Mehrabad, and Jaydeep Balakrishnan. 2015. Mathematical optimization for earliness/tardiness minimization in a multiple automated guided vehicle manufacturing system via integrated heuristic algorithms. *Robotics and Autonomous Systems* 72 (oct 2015), 131–138. https://doi.org/10.1016/j.robot.2015.05.002
[6] Luke B. Johnson, Han-Lim Choi, Sameera Ponda, and Jonathan P. How. 2012. Allowing Non-Submodular Score Functions in Distributed Task Allocation. In *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*. IEEE, 4702–4708.
[7] Luke B. Johnson, Han-Lim Choi, Sameera S. Ponda, and Jonathan P. How. 2017. Decentralized task allocation using local information consistency assumptions. *Journal of Aerospace Information Systems* (2017). https://doi.org/10.2514/1.I010461
[8] Luke B. Johnson, Sameera S. Ponda, Han-Lim Choi, and Jonathan P. How. 2010. Improving the Efficiency of a Decentralized Tasking Algorithm for UAV Teams with Asynchronous Communications. *AIAA Guidance, Navigation, and Control Conference* (2010).
[9] Athanasios Ch. Kapoutsis, Savvas A. Chatzichristofis, Lefteris Doitsidis, João Borges Sousa, Jose Pinto, Jose Braga, and Elias B. Kosmatopoulos. 2015. Real-time adaptive multi-robot exploration with application to underwater map construction. *Autonomous Robots* (2015), 1–29. https://doi.org/10.1007/s10514-015-9510-8
[10] 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
[11] Yugang Liu and Goldie Nejat. 2013. Robotic Urban Search and Rescue: A Survey from the Control Perspective. *Journal of Intelligent & Robotic Systems* 72, 2 (2013), 147–165. https://doi.org/10.1007/s10846-013-9822-x
[12] 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.
[13] James Parker, Ernesto Nunes, Julio Godoy, and Maria Gini. 2015. Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork. *Journal of Field Robotics* (2015). https://doi.org/10.1002/rob.21601
[14] Sameera S. Ponda. 2012. *Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams*. Ph.D. Dissertation.
[15] Sameera S. Ponda, Josh Redding, and Han-Lim Choi. 2010. Decentralized planning for complex missions with dynamic communication constraints. *2010 American Control Conference* (2010), 3998–4003.
[16] Alberto Quattrini Li, Riccardo Cipolleschi, Michele Giusto, and Francesco Amigoni. 2016. A semantically-informed multirobot system for exploration of relevant areas in search and rescue settings. *Autonomous Robots* 40, 4 (2016), 581–597. https://doi.org/10.1007/s10514-015-9480-x
[17] Cyril Robin and Simon Lacroix. 2015. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots* 40, 4 (2015), 729–760.
[18] Joanna Turner, Qinggang Meng, and Gerald Schaefer. 2015. Increasing allocated tasks with a time minimization algorithm for a search and rescue scenario. *IEEE International Conference on Robotics and Automation (ICRA)* (2015), 3401–3407.
[19] 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* (2017).
[20] Lihui Wang, Shadi Keshavarzmanesh, Hsi Yung Feng, and Ralph O. Buchal. 2009. Assembly process planning and its future in collaborative manufacturing: A review. *International Journal of Advanced Manufacturing Technology* 41, 1-2 (2009), 132–144. https://doi.org/10.1007/s00170-008-1458-9
[21] Amanda Whitbrook, Qinggang Meng, and Paul WH Chung. 2017. Reliable, distributed scheduling and rescheduling for time-critical, multiagent systems. *IEEE Transactions on Automation Science and Engineering* (2017).