Task Allocation for Heterogeneous Unmanned Aerial Vehicles

A. Schwager*, P. Müller, F. Knaak, and D. Moormann
Insitute of Flight System Dynamics at RWTH Aachen Universtiy, Wüllnerstraße 7, 52056 Aachen

ABSTRACT

To reduce the workload of operators in situations like coordinated search and rescue missions, in which a large number of heterogeneous Unmanned Aerial Vehicles (UAVs), are used, an automated fleet management can be of significant benefit. As part of the fleet management the different tasks must be allocated to the available UAVs. Multiple algorithms have been developed in the past to perform task allocation without any conflicts. In this paper we provide a concise introduction to prevalent algorithms and adapt a task allocation algorithm to work in dynamic environments, such as those commonly found in rescue operations. Based on the Asynchronous Consensus-Based Bundle Algorithm, the new "central mediator" is introduced, which guarantees a conflict-free allocation at every step of the allocation process and eliminates the communication between the agents representing the UAVs. The presented algorithm is applicable in any environment with reliable network connections between the "central mediator" and the agents. Furthermore, the algorithm is designed to be part of a fleet management system that considers heterogeneous UAVs.

1 INTRODUCTION

Unmanned Aerial Vehicles (UAVs) are becoming more vital to the efforts of emergency responders due to their cost and resource efficiency and lower operational risk compared to conventional means of aerial or ground reconnaissance. This is especially true in the case of major incidents, where nowadays an increasing number of different rescue organisations involved in the disaster management is using UAVs. This leads to a wide variety of UAVs present at the site, each suited for their dedicated purpose.

The resulting heterogeneous fleet of UAVs will possess a broad range of operational airspeeds and ranges. Moreover, in a highly dynamic environment, emergency responders must coordinate the said fleet of UAVs. Simultaneously the airspace in the vicinity of the incident is shared with other airborne rescue forces including rescue helicopters. All of these factors would leads to a substantial workload for the fleet operator, if no automation is implemented.

This paper is based on the requirements of the GrenzFlug+ research project, which focuses on the use of UAVs in cross-border rescue operations.. The project aims to automate the fleet and task management to reduce the workload of rescue forces. Objectives and their order of priority are defined by the rescue forces. These objectives are subsequently transformed into tasks by the "Task Management", as shown in Figure 1, and assigned a priority accordingly. In the subsequent stage the UAVs are assigned these tasks. This paper focuses on the second step, "Task Allocation", and assumes that the "Task Management" has provided the corresponding tasks and priorities. The "Fleet Management" reduces the operator's workload by eliminating the need to consider UAV-specific parameters when defining objectives.

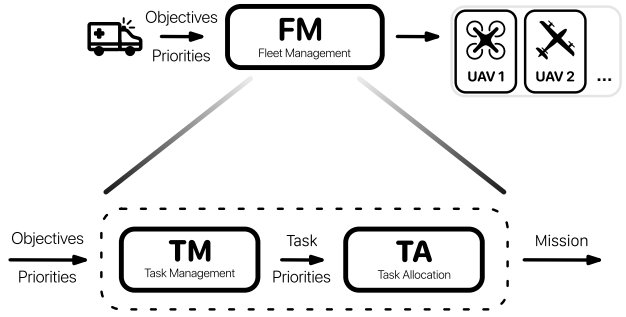


Figure 1: Generalized System Overview

To address the task allocation, a distributed algorithm based on the Asynchronous Consensus Based Bundle Algorithm (ACBBA) [1] is developed. In the algorithm each UAV is represented by an agent, which results in an agent-based approach.

In the opening section of this paper we will provide a brief overview of related work. Next, we explain the proposed approach for the task allocation algorithm, outlining the architecture, introducing the "central mediator" and giving a straight forward example. The paper concludes with the presentation of the analysis of the results of the implemented

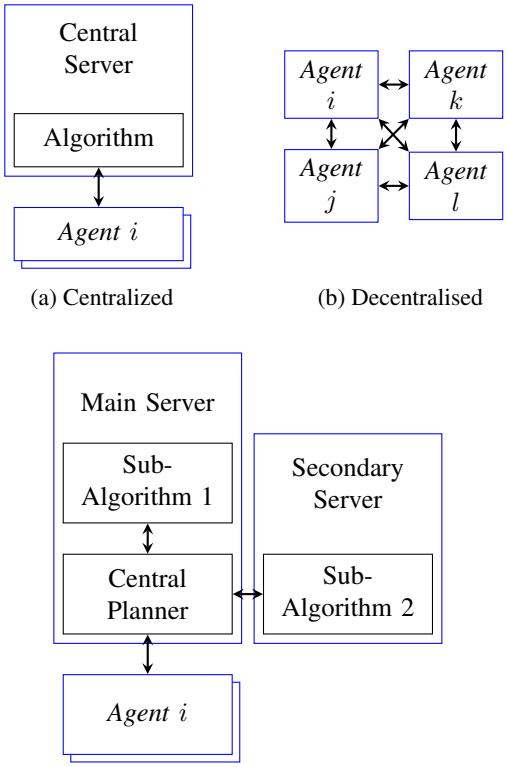
*Email address(es): schwager@fsd.rwth-aachen.de

algorithm for some test cases.

2 RELATED WORKS

2.1 Management in Multi-UAV environments

There are different architectures for the management of numerous heterogeneous UAVs. To enhance comprehension of the various architectures, Figure 2 provides an overview of the different topologies.



(a) Centralized (b) Decentralised

(c) Distributed

Figure 2: Overview of different architectures

In a centralised architecture (Figure 2 (a)) the entire task allocation problem is solved as a single optimisation problem. The advantage of centralised architectures is that they rely on a reduced amount of communication needed during the computation, as the algorithms gather relevant information a-priori and can be executed on modern multi-core computing systems. The communication performance of the single computing machine may be a limiting factor, as it needs to receive all the data relevant to fleet task management, including the necessary data from the UAVs, , services supplying airspace data and other relevant data sources. In sparsely connected environments the problem may occur that the central solver may find a solution, but cannot transmit it to all agents for execution. This may result in conflicts as not all agents work with the same task allocation. [2]

In distributed architectures (Figure 2 (c)), there are separate computing modules communicating with each other. Depending on whether they are implemented on a single machine or on separate machines, communication can be over shared memory or network connections. Distributed architectures lose performance compared to a centralised architecture as additional communication is required but enable the system to exploit the computing power of multiple machines. In contrast to decentralised architectures, the distributed architecture relies on high performance and reliable communication between the different computing machines. [2]

A decentralised architecture (Figure 2 (b)) omits the central computing unit and performs all computation on-board of the UAVs. As each agent solves the task allocation problem separately, communication is only needed to negotiate between the agents. This has the advantage, especially for sparsely connected systems, that the agents can find a solution with minimal input from other systems. Decentralised architectures are usually not designed to find the mathematically optimal solution but a solution close to the optimum to reduce the required computation time. The disadvantage is the often reduced computing performance when operating in a robust network. [2]

As discussed in Section 3.1, a distributed environment with a decentralised algorithm is chosen because it allows a flexible architecture while reducing the necessary communication with the UAV. A brief overview of the algorithms used is given below.

2.2 Consensus-Based Bundle Algorithm (CBBA)[3]

The Consensus-Based Bundle Algorithm (CBBA) is an algorithm developed using a decentralised architecture. It uses a bidding process to allocate tasks to agents. The algorithm consists of two phases:
First, in the bundle construction phase, each agent creates a bundle of tasks it wants to perform. After all agents have completed the first phase, the consensus phase starts. In this phase the agents need to exchange information about the bundles. With the information from the other agents each agent now checks whether it has been outbid by other agents for the same task. In case it has been outbid, it cannot carry out that task. As the planning for all the tasks after this one in the bundle has also become invalid, the agent must release the rest of the bundle along with the task or which it was outbid. With the reduced bundles, the bundle generation phase starts again.[4]

Choi et.al. [4] have shown that the CBBA ensures 50% optimality. As per an empirical investigation, one may anticipate performance within 93% of the optimal solution in practice [5].

2.3 Asynchronous Consensus-Based Bundle Algorithm[1]

The ACBBA is a modification of the CBBA that includes support for asynchronous operations. Similar to the CBBA, the ACBBA is designed for execution in a decentralised environment. However while the CBBA necessitates waiting for all agents to complete the bundle generation before beginning the second phase. The ACBBA modifies the consensus phase to enable bundle generation phase and consensus phases to be executed asynchronously across various agents. During the modified consensus phase, five distinct actions can be taken upon receiving a new plan from a another agent.

1. Update & Rebroadcast: Update the own state and broadcast the received plan
2. Leave & Rebroadcast: Leave own state and broadcast it instead of the received plan
3. Leave & No-Rebroadcast: The own plan is not changed and nothing is broadcasted
4. Reset & Rebroadcast: The own plan is deleted and the received plan is broadcasted
5. Update Time & Rebroadcast: Update the time stamp and broadcast the own plan with the update's time stamp

Actions 2, 4 and 5 are needed to resolve conflicts or inconsistencies between the current states of the various agents. [1]

The ACBBA has been shown to be convergent in dynamic environments as long as it can reach a solution before the environment is modified by changing the tasks to be allocated. For dynamic environments where tasks are added and deleted quicker then convergence is reached the algorithm remains operational but does not reach a convergent state. Due to the nature of the ACBBA, an analytical evaluation of the optimality of the algorithm is not possible. [1]
For lossy networks, it has been shown that the ACBBA may not always find a conflict-free solution. In order to achieve such a solution, additional measures that increase the negotiation overhead are needed [6]. This paper addresses this issue by implementing the algorithm in a distributed environment and introducing the "central mediator".

3 CONCEPT

3.1 Architecture

In a decentralised architecture, it would be necessary to compute all paths and their corresponding costs on-board. These calculations would be performed on the same machine as the flight safety critical processes, such as the conflict avoidance, or would require additional hardware, which would decrease flight performance. The sharing of these resources is only acceptable if the path planning is performed infrequently. For the given use-case the task distribution and segmentation shall be optimized for the available UAVs. This requires more iterations and an increased computational effort on each UAV. To meet this requirement, a distributed architecture was chosen for the given use-case. The algorithm runs on a single high-performance machine to ensure the necessary computational effort. For each UAV, the architecture features an individual agent that runs on a separate thread on the same machine as the central coordinating instance.

A further advantage of employing a distributed architecture, as opposed to a decentralised one, is that the UAVs do not require constant communication with one another and that no external data, such as environmental information, must be transferred to them. This reduces the bandwidth required for mission planning and therefore frees up bandwidth for communication relevant to flight safety.

As the fleet management system incorporates heterogeneous UAVs, it is feasible that tailored path planning tools for individual types of UAV may be utilised in the future. This would prove a challenge in a centralised architecture. In addition, employing a distributed approach enables future expansion of the fleet task management system via inclusion of new modules.

3.2 Task Allocation

For the task allocation within the fleet task management the ACBBA [1] has been selected as the foundation. The ACBBA is preferred due to its ability to support asynchronous operations and independence from the underlying path planner. This is a necessity when using different path planners with different execution times for a heterogeneous fleet. Furthermore, the ACBBA can adapt to a dynamic environment where new tasks are constantly added and removed; a critical feature when the task allocation algorithm is employed with the task management.

In the ACBBA all calculations are performed by the agent's on-board processing unit. In a first adaption step all calculations are moved into the distributed environment, where separate threads on the same machine perform calculations for each UAV. This allows the ACBBA's asynchronous operations to be maintained while taking advantage of the benefits of a distributed architecture.
As the ACBBA was developed for a decentralised environment, it requires significant effort to resolve conflicts arising from each UAV's individual current bidding result [1][6]. However, in a distributed environment where communication is assumed to be nearly instant and fully reliable, these complex algorithms can be replaced by a central negotiation instance - the mediator. The bundle generation phase defined in the CBBA [4] is carried out independently and asynchronously for each agent, with the consensus phase shifted from the agents to the mediator. By achieving conflict-free resolution following each iteration, the task can be terminated

and the current solution can be utilised without necessitating additional conflict checks.

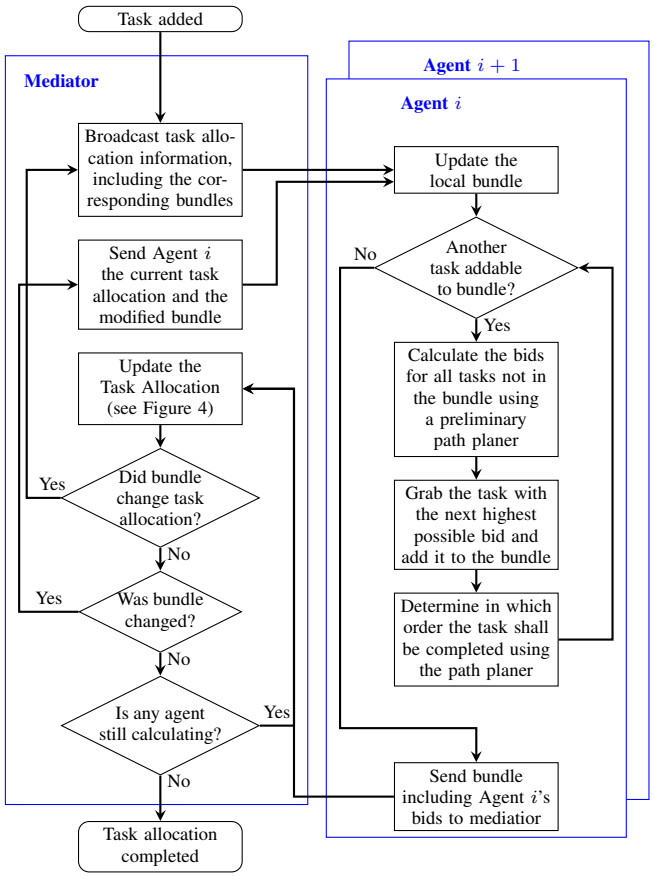


Figure 3: Flow Chart of the Task Allocation Algorithm

The procedure utilized in the proposed adaptation of the ACBBA is illustrated in Figure 3. The initial step involves broadcasting the existing tasks, along with the corresponding winning bid and the currently winning agent for each task, to all agents. In the first broadcast no winning bids and agents are included as there has been no bidding process yet.. With the task list and the current allocation each agent attempts to secure additional tasks by bidding higher than the currently allocated agent.

Firstly the agent calculates its bid for all tasks not yet in its own bundle. Next, the agent determines whether its bid would surpass the current winning bid. Finally, among all the bids that beat the current winning bid, the agent selects the one with sthe highest bid. Afterwards, the path planner is executed to generate a path that considers all tasks in the updated bundle, including the newly added task. These steps are repeated until the agent is unable to add further tasks to its bundle. This happens either when the agent has added all the tasks where it can outbid the current winning agent or, when it is unable to complete any more tasks due to reaching its limitations, which vary depending on the UAV represented by the agent. A possible example for such a limit is that the maximum flight distance achievable by the UAV has been reached or that the remaining tasks are unreachable due to obstacles.

For the specific application of a major disaster, the key consideration is the timeline for completing the task and the task's priority. Task priority measures the value of a task and is defined by the "Task Management". The time taken to complete a task is calculated by the agent and represents the cost. The bid is determined by employing equation (1) in which j denotes the task number and a_i the agent currently performing the bundle generation. The resulting bid, $c_{a_i,j}$ is determined by the task priority p_j, the time required for task completion $t_{completion,j}$ and the scaling factor T. For the scope of this paper $T = 1\,\mathrm{s}$ will be assumed.

$$c_{a_i,j} = p_j - \frac{t_{completion,j}}{T} \tag{1}$$

Once the Agent i has completed the bundle generation , it transfers the bundle along with the corresponding bids to the mediator. As the mediator always stores the most recent task allocation, the assessment of the update times for each of the bids can be omitted, thus simplifying the consensus phase. During the mediation phase, the mediator iterates through the bundle of Agent i, as depicted in Figure 4. It should be noted that a consensus phase may lead to a task, previously allocated to an agent, having no agent assigned.

If, during the consensus phase, there is a change in the task allocation, the new allocation is broadcasted to all agents along with their updated bundle, thus initiating a new bundle generation phase. If the task allocation remains unchanged, but Agent i's bundle changes, only Agent i will be send the updated information as only it needs to repeat the bundle generation phase. If neither task allocation nor bundle are changed, Agent i has achieved consensus with the present task allocation. If no other agent is performing a bundle generation phase at this point, the final task allocation was found.

3.3 Simplified Example Scenario for the Task Allocation Algorithm

For a improved comprehension of the adapted algorithm an example involving three tasks and two UAVs is given (see Figure 5). The Agent TW (a_{TW}) is a tilt-wing aircraft flying in fixed-wing configuration with a constant ground speed of $20\,\frac{\mathrm{m}}{\mathrm{s}}$, whereas the Agent MR (a_{MR})is a multi rotor system flying with $10\,\frac{\mathrm{m}}{\mathrm{s}}$ ground speed. All three tasks are waypoints to be reached with a priority factor $p_j = 20$. For the sake of simplicity, curves are omitted.

Each agent has a bundle vector storing the current bundle ($b_{a_{TW}}$ and $b_{a_{MR}}$) and a corresponding bid vector ($c_{a_{TW}}$ and $c_{a_{MR}}$). As a first step the mediator broadcasts the complete list of all tasks, with no winning agent and no winning bids.

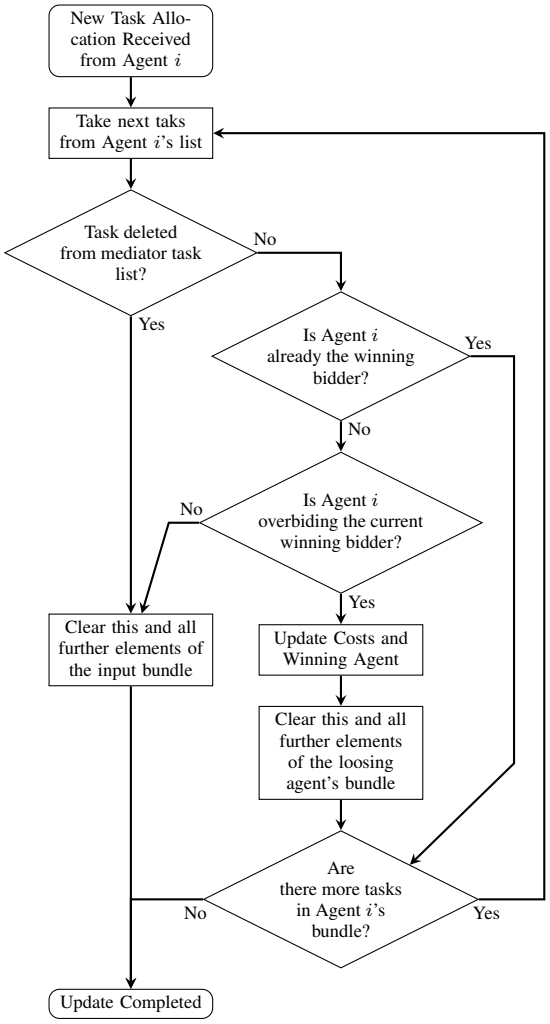


Figure 4: Flow Chart of the consensus step performed by the mediator

From the given allocation both agents generate their corresponding bundles. For the given case Agent 1 generates the bundle $b^I_{a_{TW}} = \begin{bmatrix} t_1 & t_2 & t_3 \end{bmatrix}^T$ with the corresponding bids $c^I_{a_{TW}} = \begin{bmatrix} 18 & 15 & 12 \end{bmatrix}^T$. Agent TW is quicker in completing the bundle generation phase and therefore initiates the first mediation phase. After this mediation phase the mediator has the task allocation shown in Table 1.

	$a_{winning}$	$c_{winning}$
$t_1^{I\,a}$	a_{TW}	18
$t_2^{I\,a}$	a_{TW}	15
$t_3^{I\,a}$	a_{TW}	12

Table 1: Task Allocation after the first consensus phase with the bundle from Agent TW

As the task allocation has changed, the mediator broadcasts it to both agents . As Agent TW determines that the allocation corresponds with its own bundle and thus, does not perform a new bundle generation phase while Agent MR is still running its first bundle generation phase. Therefore this broadcast does not trigger any new bundle generation phases at this moment. When Agent MR completes its initial bundle generation phase, it triggers the second mediation phase with the bundle $b^I_{a_{MR}} = \begin{bmatrix} t_3 & t_2 & t_1 \end{bmatrix}^T$ and the corresponding bids $c^I_{a_{MR}} = \begin{bmatrix} 14 & 8 & 2 \end{bmatrix}^T$.

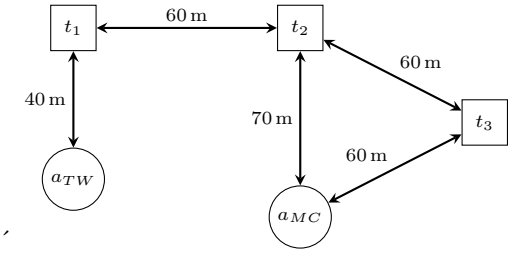


Figure 5: Locations of tasks and UAVs for the simplified example

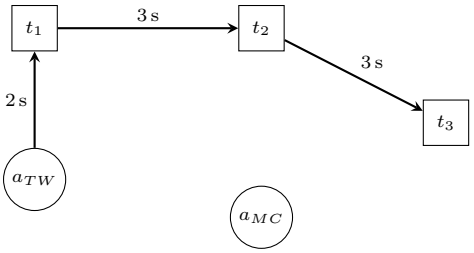


Figure 6: Simplified path planned by a_{TW}

	$a_{winning}$	$c_{winning}$
$t_1^{I\,b}$	a_{TW}	18
$t_2^{I\,b}$	a_{TW}	15
$t_3^{I\,b}$	a_{MR}	14

Table 2: Task Allocation after the second consensus phase with the bundle from Agent MR

As the Agent MR has out-bid Agent TW on on the first

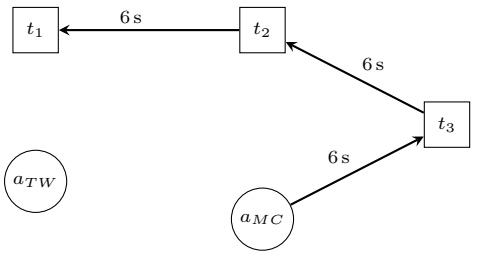


Figure 7: Simplified path planned by a_{MR}

task in its bundle and has been out-bid on the second task in its bundle, the task allocation and both agents will perform a new bundle generation phase using the input data from Table 2. During this bundle generation phase Agent MR generates the bundle $b^{II}_{a_{MR}} = [t_3]$ and the corresponding bids $c^{II}_{a_{MR}} = [14]$. Agent MR does not select tasks t_1 and t_2 as it is aware that it cannot outbid Agent TW for these tasks. Similarly, Agent TW does not select task t_3 as it acknowledges that it cannot outbid Agentt MR for this particular task. Since neither of the agents modified its bundle, the allocation remains unchanged in the following mediaton phases. Thus, the task allocation algorithm has reached a convergent state.

4 RESULTS

The task allocation algorithm, as outlined in section 3.2, is implemented using ROS2 with C++ and Python being the primary languages employed. This implementation was tested with varying number of tasks, each requiring a set point to be reached under a specific heading. The test scenario was tackled by 10 agents, each planning dubbins paths to complete the different tasks. The number of agents was selected to compare the outcomes of the ACBBA presented in [1] and the results attained through the algorithm proposed within this paper.

With the described scenario and path planner it was found that the bundle generation phase was at least four times slower than the mediation phase for the lowest number of tasks, while it was approximately fifteen times slower for the largest number of tasks. As 10 agents were used to tackle the given tasks the mediator becomes a bottleneck when the bundle generation phase is less than 10 times slower than the mediation phase. Since the utilized path planner does not consider obstacles or flight performance, it may be assumed that this issue will be resolved with the implementation of more advanced path planners.

The scenarios were executed on various systems, enabling the agents to compute independently. During the evaluation of the results, it is evident that the number of mediations is directly proportional to the time required to reach a convergent state for a given system. To have a measure of the convergence speed that is independent from the processor speed the number of mediations was used to evaluate the results. Figure 8 illustrates the number of messages against the number of tasks in the specified scenario.

For the test scenarios, a linear relationship is evident between the number of tasks and the mediator steps required. This is the expected behaviour as an increase in number of tasks leads to a larger bundle size, resulting in a more dropped tasks during the mediation phases when an agent is out bidden by another agent.

For the comparison between the ACBBA [1] and the developed algorithm, it is necessary to compare the number of

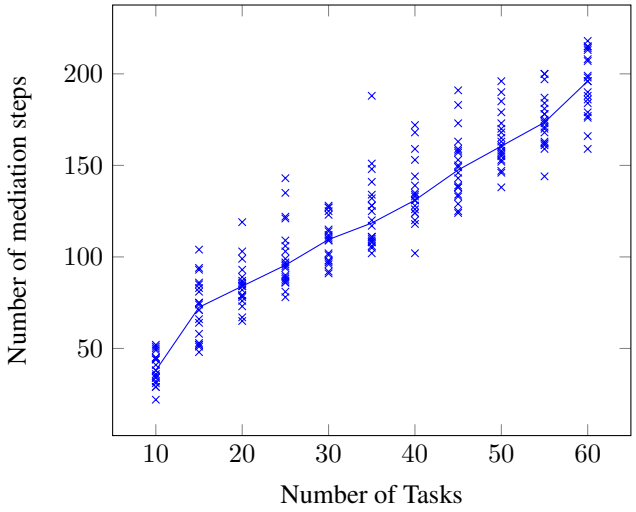


Figure 8: Number of mediator steps required to reach consensus for the given number of tasks including the median

messages used by the ACBBA and the number of mediation phases required. However, it is important to note that one mediation phase is equivalent to $n_{msg} \approx n_{agents} - 1$ messages. This relation results from broadcasting of the task allocation at the end of the mediation phase, which is the equivalent to sending a message to each of the nine other agents.

When comparing the performance of the ACBBA and the developed mediator-based algorithm, three sections exhibit distinct comparison results. For the scenarios with 20 or fewer tasks the performance of both algorithms is similar with a slight advantage for the developed mediator approach. For the scenarios with $20 < n_{task} \leq 35$, the developed mediator-based algorithm proves significant superior to ACBBA. For instance, in the scenarios with 30 tasks, the median number of messages required by the ACBBA is thrice that of the developed mediator-based algorithm.

For the scenarios featuring over 35 tasks, the ACBBA performs increasingly better, so that the median number of messages required by ACBBA for the scenarios with 60 tasks is only 30% higher than that of the developed mediator-based algorithm. An explanation for the reduction of required messages was not provided by Johnson et.al. [1]. However, it can be assumed that the agents have a limit on the maximum path length. This would result in fewer conflicts to resolve for higher numbers of tasks, thus reducing the number of messages required. The simulation of the developed mediator-based algorithm did not have a similar limit. Therefore, the comparability of the two algorithms is reduced for all scenarios with more than 35 tasks.

5 CONCLUSION

An adapted task allocation algorithm is presented for the use case of a connected UAVs in a search and rescue scenario. The algorithm has been designed to run in a

distributed environment and is based on the Asynchronous Consensual Bundle Based Algorithm [1]. This choice was made because the algorithm enables the integration of heterogeneous UAV as agents. To improve performance the consensus phase of ACBBA has been replaced by the newly introduced mediation phase, which is executed by the new central mediator. The performance improvement was demonstrated performing simulation runs of the developed mediator-based algorithm and comparing the results with simulation runs for the original ACBBA.

The task allocation algorithm has been designed with future extendability in mind. A planned extension is integrating an interface to the U-space system for the planning phase, in order to identify potential conflicts and request prioritisation or perform strategic deconfliction.

Further work is needed to determine if the mediator-based algorithm developed can also exhibit a reduction in required messages, as demonstrated by the ACBBA, for a larger number of tasks. . The function used to compute each agent's bid is also a topic for future work, as is the impact of the introduction of a range-limit for the agents.

ACKNOWLEDGEMENTS

The project "GrenzFlug+" is funded by the German Federal Ministry of Digital Affairs and Transport as part of the "Innovative Air Mobility" funding guideline based on a resolution of the German Bundestag.

REFERENCES

[1] Luke Johnson, Sameera Ponda, Han-Lim Choi, and Jonathan How. Asynchronous decentralized task allocation for dynamic environments. In *Infotech@Aerospace 2011*, volume 25, pages 912–926. American Institute of Aeronautics and Astronautics, March 2011.

[2] Sameera S. Ponda, Luke B. Johnson, Alborz Geramifard, and Jonathan P. How. Cooperative Mission Planning for Multi-UAV Teams. In Kimon P. Valavanis and George J. Vachtsevanos, editors, *Handbook of Unmanned Aerial Vehicles*, pages 1447–1490. Springer Netherlands, Dordrecht, 2015.

[3] Luc Brunet, Han-Lim Choi, and Jonathan How. Consensus-Based Auction Approaches for Decentralized Task Assignment. In *AIAA Guidance, Navigation and Control Conference and Exhibit*, Honolulu, Hawaii, August 2008. American Institute of Aeronautics and Astronautics.

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

[5] Luca Bertuccelli, Han-Lim Choi, Peter Cho, and Jonathan How. Real-Time Multi-UAV Task Assignment in Dynamic and Uncertain Environments. In *AIAA Guidance, Navigation, and Control Conference*, Chicago, Illinois, August 2009. American Institute of Aeronautics and Astronautics.

[6] Matthew Rantanen, Nicholas Mastronarde, Jeffrey Hudack, and Karthik Dantu. Decentralized Task Allocation in Lossy Networks: A Simulation Study. In *2019 16th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9, Boston, MA, USA, June 2019. IEEE.