Distributed task allocation for swarm robots in a constrained communication environment using a grid based allocation process

NAMES
Deakin University, Australia
EMAILS

Abstract

Semi-Autonomous Multi Robot Systems provide alternative solutions for real-life problems such as firefighting and rescuing in a rural environment. Task allocation is a important part of the system and many task allocation algorithms exist to handle such problems however some important considerations such as communication constraints, scalability, performance are not all addressed by every algorithm. This paper presents an allocation process which addresses theses issues. This allocation process is incorporated into two different types of suitable algorithms and tested using various experiments, each of which address a important task allocation metric. The results from these experiments are discussed in terms of positives and negatives for an ideal task allocation algorithm.

1 Introduction

Bushfires in Australia are a widespread and regular occurrence that have contributed significantly to shaping the nature of the continent over millions of years. Bushfires have killed approximately 800 people in Australia since 1851 and billions of animals. In 2012, the total accumulated cost was estimated at $1.6 billion. Bushfires are especially problematic in remote areas where it is difficult to stop the spread of fires due to the terrain and logistical problems. [27]

With the exponential rise of technology in the past decade, it has become possible to implement a Semi-Autonomous Multiple Robot system to solve complex problems. A system comprising of a swarm of robots cooperating with each other can be used in our scenario to mitigate and stop the spread of fires faster without putting human lives in danger. [2, 5, 7, 18, 24]

Task allocation is an important part of such a system and tries to replace a human allocator from the system. Considering the importance of optimal allocation in the system, task allocation algorithms need to be designed carefully to address all the challenges of using robotic swarms in remote areas. One key challenge that comes with the environment is the communication costs between the robots. As the computational resources are scarce and ideally should be prioritized towards the actual problem solving, using centralized cooperation methods are not efficient or scalable in such a system. [12, 16, 21]

Another important requirement of such an algorithm is the ability to maintain the quality and speed of allocations when the number of robots are increased in the system. As real world mission zones are dynamic, the number and types of robots for each specific problem set can be different. Thus, the task allocation process should be able to account for the variance in the quantity and type of robots while also ensuring the quality and speed of the allocations. [8]

The main goal of this paper is to present an grid based allocation process for task allocation and implement two different types of algorithms using the allocation process as the main framework. For each of these algorithms several different experiments are designed, each of which addresses a key challenge of the scenario. From the results of the experiments the algorithms are compared with each other and their positives and negatives are discussed according to the type of scenario.

To implement experiments for the algorithms identified in the literature review, a testing system is proposed which represents a 3D rural area with all the necessary components to test decentralized task allocation in a real-life scenario. The simulation area is divided into various zones meant to highlight the characteristics of such area and how these characteristics also influence task allocation. One major disadvantage of decentralization is the processing problem required by the agents in the system which can cause optimization issues negating the advantages of decentralization. To address this issue, a grid based allocation process is also proposed. [9,14,16]

The algorithms are evaluated using different experiments, each of which aim to address a key challenge of

a Multi Robot System in a communication constrained environment. The main challenges addressed are suitability in relation to the scenario, scalability in terms of the performance and number of agents in the system, the quality of task assignments, communication resources required. The results of these experiments are presented in form of visual graphs and plots, each aiming to highlight the differences between the algorithm. Based on the results, a modified version of the algorithms is presented. [12, 16]

The remainder of the paper is organised as follows. Section 2 presents the related work. The technical details of the system and the algorithms are presented in Section 3. Section 4 presents the experiments on which the algorithms are evaluated and the results of the experiments. Final conclusions and future ideas are presented in Section 5.

2 Related Work

Task allocation techniques have been researched extensively over the years and are still evolving over time. Task allocation assigns the tasks to agents and aims to find the most optimal task assignment solution. Most of the task allocation algorithms focus on ensuring that the distance travelled by the robot is minimized while also ensuring that the robot is capable to perform the task successfully. An important part of task allocation is task re-allocation. In a case where the task allocated to the agent cannot be performed successfully, it must be reallocated to another agent. However, repeated preempting can sometimes cause the agents to never complete their tasks causing the whole system to fail. [21]

Also, because the problem of task allocation is a dynamic decision that can vary in time due to different environmental and system changes, the problem needs to be solved iteratively over time. This makes the problem of task allocation more difficult to solve optimally. [8]

A common way to approach the Multiple Robot Task Allocation (MRTA) problem is to treat it as a Optimal Assignment Problem (OAP) where the objective is to optimally assign a set of robots to a set of tasks in such a way that optimizes the overall system performance subject to a set of constraints. [13]

The problem of OAP can be defined mathematically as:

1. Robots: a team of mobile robots r_i; i = 1, 2,.., n.
2. Tasks: a set of tasks t_j; j = 1, 2,.., n.
3. Utility: a set of robots' utilities, u_{ij} is the utility of robot i to execute task j.

For a single sensor task, the problem is to find the optimal allocation of robots to tasks, which will be a set of robot and task pairs. [6] For the general case, the problem is to find the optimal allocation of a set of tasks to a subset of robots, which will be responsible for accomplishing it. [28]

In some MRTA approaches such as market-based approaches, each robot r ∈ R can express its ability to execute a task t ∈ T, or a bundle of tasks G ∈ T through bids br(t) or br (G). The cost of a bundle of tasks can be simply computed as the sum of costs of the individual tasks where f is the number of tasks of the bundle G. The group's assignment determines the bundle G ∈ T of tasks that each robot r ∈ R receives.

"When there are more tasks than agents, OAP can still be used in its iterative form: in each computation, the optimal solution is found with one task per agent, leaving excess tasks unassigned. Assuming enough time elapses between task completions, this iterative algorithm can be used to find a new assignation upon each task completion." [17]

Most OAP based algorithms use a cost model which works around principles such as costs, priority/fitness/reward to formulate a model. A combination of these principles is used to calculate the utility or value of performing a task. [1, 15, 23]

- Cost: a characterization of the cost that it takes for a robot to execute a task. Examples are time to reach a goal, distance traveled, energy consumed, etc.
- Priority: a characterization of the urgency of completing a task. Depending on how the system is designed, higher priority tasks can preempt all lesser tasks.

$$Utility = Priority - Cost$$

Essential it comes down to the ability, gain, need to do a task versus the cost of doing a task to calculate the utility. It is important to note that utility can never be negative, and a floor function is used to ensure that,

$$Utility = max(Reward - Cost, 0)$$

Some OAP based algorithms also opt for an auction based models which are inspired from real life economic negotiations in an auction where there is a seller and multiple bidders. The simplest auction model can be generalised as a three-step negotiation: [13, 22]

1. Step 1: a task is published to the agents by some entity, usually called auctioneer. In many occasions the auctioneers are the agents themselves.
2. Step 2: agents suited to the tasks reply to the auctioneer with a bid on the task.
3. Step 3: the auctioneer awards the task to some agent, after evaluating the received bids.

While the auction based algorithms are very simplistic and ensure good performance, they struggle with scalability. However, a partial auction based system which combines with another model such as the cost model can be used to ensure high performance while also being scalable to a certain extent. [26]

3 Proposed Approach

For the purposes of this research, a simulation model is developed which represents a 3D rural area. The system consists of various tasks/targets and agents/robots. The tasks are heterogeneous and require robots with some specific skills to accomplish them, similarly the robots are also heterogeneous and capable of handling different subset of tasks. The System is used to compare the different algorithms under subsets of constant parameters.

The 3D simulation area consists of four different zones each with their own characteristics to better represent a real-life rural area and thus assist in the task allocation process. Each zone has a priority modifier which changes the base quality of any task. This modifier helps to give tasks in certain key zones higher priority over similar tasks in other low priority zones.

- Zone-1 is a Major Town center which acts as the main population hub of a regional area. The allocation priority factor in this zone is the maximum.
- Zone-2 is a regional Farmland which acts as a major economical zone. The allocation priority factor in this zone is second to Z1.
- Zone-3 is part of a National park occupied with rare flora and fauna which helps the region to attract visitors.
- Zone-4 is also a part of the National park with fewer attractions and has the lowest allocation priority factor in the 3D region.

Within this 3D Area, tasks of varying characteristics are randomly generated in different zones. Instead of having a single centralized communication hub keeping track of all the tasks, it is possible to have many separate communication hubs in the form of scout robots who find and add the tasks in the queue. These scout robots act as intermediaries which send the task details to the robots, who then make an independent decision on whether an allocation should be made. This decentralization process allows the system to be dynamic and flexible.

For the proposes of this research, how the tasks are located and added in the queue is not relevant to this study and it is presumed that tasks are dynamically added to the task queues by scout robots and then made visible to other work-capable robots.

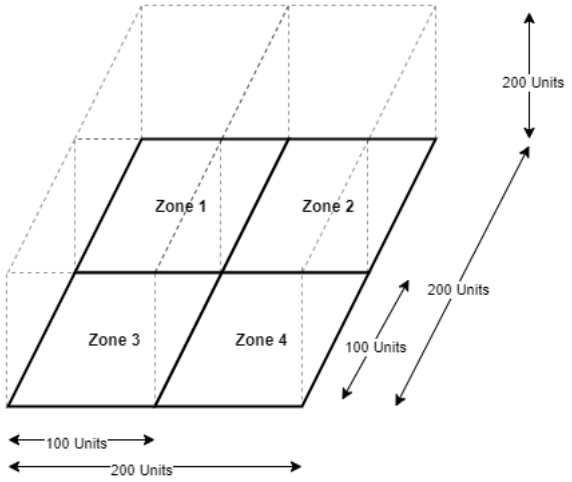

Figure 1: Representation of the zones in the simulation.

3.1 Grid Based Allocation Process

A major disadvantage of decentralization is the processing power required by the agents and the increased time taken by the robots to run the algorithm. If improperly implemented, this can cause optimization issues which sometimes negate the positives of decentralization itself. To address this problem of optimization, a grid/zone based allocation process is proposed. [3, 4, 10, 20] The process of allocation in this process works as follows:

- *Step-1 Scout searches for robots in the same zone:* Upon being discovered by a scout robot, the task is added to its queue and initially is not visible to all the robots. The scout robot tries to locate robots in the same zone as the task and if any robots are found, task details are relayed to them for processing.
- *Step-2 Robot is capable and not busy:* The robots which receive the task information begin checking whether they are capable of performing the specific task concurrently. If the robot is capable and has no current assignments, it will allocate the task to itself and send confirmation of allocation to the scout robot. Whichever working robot sends the confirmation first has the allocation accepted and the task status is changed to allocated by the scout robot.
- *Step-3 Robot is capable and busy:* If the robot is capable but already has an active assignment, it will compute the qualities of the current and new task. Using a differential factor which can be customised according to specific mission requirements, it will assess whether a reallocation is feasible. If the difference between task qualities is greater than the differential factor, then reallocation will be performed. This information is relayed back to the scout robot

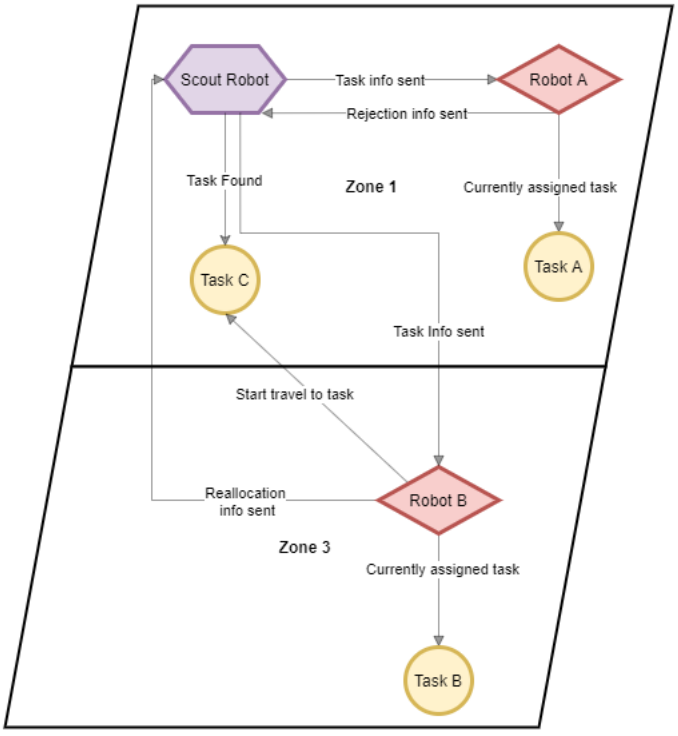

Figure 2: Example of one of the possible grid base allocation scenario.

which manages the status of both the tasks accordingly.

- *Step-4 Robot is not capable or reallocation is not feasible:* If the robot cannot perform the task or it can but reallocation is not feasible, no allocation can be done and the same information is relayed back to the scout robot. The scout robot will wait for responses from other robots in case where they were able to accept the task.
- *Step-5 No suitable robots in the same zone:* If no robots could accept the task in the zone, the scout robot will relay the task information to other nearby zones iteratively repeating the steps 2 to 4 until an allocation is found for the task.
- If no suitable robots were found for the task, steps 1-5 are repeated n-1 times where n are the number of tasks or until all the tasks have been allocated. In a case where no allocations could be made even after n-1 tries by the scout robot, it will wait for other tasks to be finished before making any further attempts to send task information to the robots.

Figure-2 shows an possible scenario which of the grid base allocation process. In the visual Robot A and B are currently busy with their tasks A and B respectively. The scout robot which is present in Zone 1 finds a new Task C. The scout robot follows the grid based allocation process and sends the task C info to Robot A first. Robot A computes the information and decides reallocation is not feasible and sends a rejection message. As there are no more robots in Zone-1, The scout robot then sends the task info to the robot in the closest zone, which is Zone-3. Robot B also computes the task information and decides reallocation is feasible and sends the reallocation message to the scout robot. The scout robot manages the status of the tasks B and C to "allocated" and "not allocated" respectively.

This proposed grid based allocation process is implemented by all of the tested algorithms in addition to their own specific allocation process. The algorithm-1 shows the pseudo-code for this allocation process.

Algorithm 1: Grid Based Allocation

```
Input: List of Tasks and Robots in the
       environment.
Function GridBasedAllocation(task_list,
 robot_list)
    if task_list is not Empty then
        run ⟵ 0
        while run < number of tasks - 1 do
            for task in task_list do
                taskZone ⟵ task.getZone()
                for robot in robot_list do
                    if robot.getZone == taskZone
                     and task.taskAllocated is False
                     then
                        /* Call Individual
                           Algorithm Code on
                           every robot until an
                           allocation is found
                           */
                if task.taskAllocated is False then
                    robotZoneList ⟵
                     getRobotZoneList(robot_list)
                    for robotZone in robotZoneList
                     do
                        /* Call Individual
                           Algorithm Code on
                           every robot in nearby
                           zones step by step or
                           until allocation is
                           found              */
```

3.2 Decentralised Bees Algorithm

The Distributed Bees Algorithm is a modified decentralized version of the original Bees algorithm. The Bees algorithm is inspired by real life behaviour of bees in nature. When the bees are foraging for food, the scout bees

will head out and survey different areas and come back to the hive with their results. Depending on the quality or quantity of food found, they will waggle dance in front of other bees. The duration of this dance reflects the quality of the food. [11, 19] In our scenario, it is presumed that the scout robots find new tasks and their associated priority and communicate that information to the robots as needed.

The DBA uses a cost model to decide allocations. Each task has an associated quality which is a specific-scenario scalar value the represents the tasks priority or complexity where higher value requires more robots to be allocated. Each task also has a cost associated with it which is dependent on the Euclidean distance of any robot to the target location. This cost and quality are used to calculate the utility of the task where higher quality is good.

In a centralized version the utility probability values can be easily used to calculate the most optimized allocations with the downside of communication. But as the decentralization aspect is particularly important in the scenario a modified version of the algorithm is used. This modified version uses the grid based technique to allow the closest robots to access their ability to perform a task. Algorithm-2 shows the pseudo-code for the modified. Decentralised Bees Algorithm.

Algorithm 2: Decentralised Bees Algorithm

```
Input: A Task class object which has
       information about a task.
Output: Boolean value which tells if an
        allocation was made.
Function DBA(task)
    capability ⟵ checkCapability(task)
    if capability is True then
        status ⟵ checkCurrentStatus()
        if status is False then
            distance ⟵ calculateDistance(task)
            quality ⟵ getQuality()
            utility ⟵ quality ÷ distance
            assignTask(task, utility)
            return True
        else
            current_quality ⟵
             getCurrentQuality()
            new_quality ⟵ getQuality()
            if current_quality − new_quality >
             differential_factor then
                distance ⟵
                 calculateDistance(task)
                quality ⟵ getQuality()
                utility ⟵ quality ÷ distance
                assignTask(task, utility)
                return True
            else
                return False
    else
        return False
```

3.3 S+T algorithm

The S+T algorithm is based on a distributed market-based approach with temporal agents and is an extension of the SIT algorithm. The algorithm acts like any market-based approach with a few modifications such as the introduction of services. It is presumed that in a MRS, certain tasks will require more than one robot to complete it or make it achievable. For example, consider an example related to our scenario, A person is stuck in one of the Zones surrounded by bushfires. The robot which is capable of doing the task cannot reach the person directly so it requires additional support from other robots to clear the fire first. This pre-requisite task is called as an service. The services are generated dynamically and dependent on the robots and the system. [25, 26]

While this algorithm is not completely decentralized as it needs a robot to act as an auctioneer (temporal agent), the communication costs can be reduced using the grid based approach while still keeping the major advantage of the cooperation this algorithm provides.

When a task is found by a scout robot, the robot broadcasts its information to the nearest robot in the zone. The nearest robot checks if its capable, not busy and registers the utility value as its own bid for the task. If the task requires no service, then the robot allocates that task to itself. Else, the task is also broadcasted to the robots in the zone, who also check whether they are capable and register their own bid to the auctioneer. The robot which can complete the task with no or minimal service tasks is assigned to the task. The pseudo-code for this modified version is shown in Algorithm-3.

4 Experiments and Results

4.1 Comparison Metrics

For the experiments to have any meaningful results, comparison metrics need to be defined which the experiments use to highlight the advantages and disadvantages of the various algorithms. Some of these metrics are more important in determining a specific criteria than the others and that will be mentioned if it applies.

These metrics are as follows:

1. The time taken to allocate all the tasks: also called global allocation time, used to check the performance of the algorithm in allocating all the tasks in the queue.The value is stored in seconds.

Algorithm 3: S + T Algorithm

```
Input: A Task class object which has
       information about a task.
Output: Boolean value which tells if an
        allocation was made.
Function SPT(task)
    capability ⟵ checkCapability(task)
    if capability is True then
        status ⟵ checkCurrentStatus()
        if status is False then
            if serviceRequired() is False then
                assignTask(task, utilityTask)
                return True
            else
                service ⟵ getTaskService()
                selfCapability ⟵
                 checkCapability(service)
                if selfCapability is True then
                    assignService(service,
                     utilityService)
                    assignTask(task, utilityTask)
                    return True
                else
                    assignTask(task, utilityTask)
                    return broadcastService-
                     ToOtherRobots()
        else
            current_quality ⟵
             getCurrentQuality()
            new_quality ⟵ getQuality()
            if current_quality − new_quality >
             differential_factor then
                if serviceRequired() is False then
                    assignTask(task, utilityTask)
                    return True
                else
                    service ⟵ getTaskService()
                    selfCapability ⟵
                     checkCapability(service)
                    if selfCapability is True then
                        assignService(service,
                         utilityService)
                        assignTask(task, utilityTask)
                        return True
                    else
                        assignTask(task, utilityTask)
                        return broadcastService-
                         ToOtherRobots()
            else
                return False
    else
        return False
```

2. The time taken to allocate a specific task: also called local allocation time, used to check the performance of the algorithm in allocating specific tasks in the queue. Local run-time for crucial tasks is minimized at the cost of the global run-time by the ideal algorithm. The value is stored in seconds.
3. If reallocation is performed, how many times: used to check if reallocation is performed. An ideal algorithm will prevent multiple re allocations to prevent looping and optimization problems. The value is stored as integers.
4. If reallocation is performed, time taken to reallocate: used to check the performance of the algorithm in reallocating the tasks in the queue. This metric is important as it reflects on a algorithms ability to be flexible. The value is stored in seconds.
5. Average Utility value of an allocation: used to check how good an average allocation made by the algorithm is. An ideal algorithm will have a higher average utility value. The value is stored as a scalar value.

4.2 Experiments

To evaluate the proposed methodology, different experiments are designed each aiming to address a key characteristics such as validity of algorithm, scalability in terms of performance and number of agents and tasks in the system, quality of allocations made by the algorithms.

The experiments are as follows:

1. Experiment-1 Validity Test: This experiment tests whether the algorithm logic is working alongside the grid based allocation process. The experiment is focused around the validity and feasibility of the algorithms in making allocations for a small set of agents and tasks in the system. The ideal algorithm will be able to demonstrate that the grid based allocation process is working correctly alongside its own decision making process. The number of agents and tasks in this experiment are kept as low as possible to keep focus on the validity aspect. The specific details for this experiment are as follows:
    - The number of robots are set to two, one aerial and one ground robot respectively. The locations of these robots are randomized in the 3D environment.
    - The number of tasks are set to two, one aerial and one ground based task respectively. The type and the location of the tasks are randomized.
    - For each of the algorithms, the experiment is done. The system components from the experiments are visualised in the 3D environment to

demonstrate the algorithms are working properly.

2. Experiment-2 Scalability Tests: These series of experiments test the scalability of the algorithms in terms of performance when the number of robots and tasks in the system are incrementally increased. The focus of this experiment is to see how the algorithms will measure in respect to average global allocation-time, average local allocation-time, how many times was reallocation necessary and the extra time consumed during reallocation process. The ideal algorithm will be able to demonstrate that it can handle task allocation efficient and without fault when the number of entities in the system are huge. The specific details for this experiment are as follows:

   - The number of tasks in the system are always equal to the number of robots. This is done to keep the focus on the scalability aspect of the algorithms instead of distributability of resources.
   - The number of robots and tasks are increased in the order of 10,50,100,150,200,250. The location, types of these robots and tasks are randomized respectively.
   - To account for the randomization in the experiment during the incremental process, each set of robot and tasks are run ten times and the results are averaged.

   As the increase in number and variance of robots and tasks in the system is also important in measuring other characteristics of an algorithm, This experiment is further divided into the following subcategories:

   (a) Experiment-2.1: Comparing the Global Average Time taken by the algorithms to allocate tasks. An ideal algorithm will have a lower global average time compared to other algorithms.

   (b) Experiment-2.2: Comparing the Local Average Time taken by the algorithm to allocate any singular task. Similar to the global average time, an ideal algorithm will have a lower local average allocation time compared to other algorithms.

   (c) Experiment-2.3: Comparing the Average time consumed by the algorithms during reallocation process.This is done to show that the algorithms don‘t spend too much time looking for the perfect allocation. An ideal algorithm will have low reallocation time values.

   (d) Experiment-2.4: Comparing the average quality of an allocation made by the algorithms. The focus of this experiment is to see how the algorithms will compare to each other in respect to the average utility value of an allocation which in simpler terms is how good any individual allocation made by the algorithm is. The ideal algorithm will have a higher average allocation utility value compared to the other algorithms.

4.3 Results

From the data retrieved by testing the experiments on the algorithms, different specific metrics were extracted and stored in tables. Using this sorted data in the tables, various visuals were generated as the results of the experiments. These visuals are discussed for both sets of algorithms and key details are highlighted.

Figures-3, 4 show allocation map of the validity testing in the form of basic allocation maps. The purple color in the map represents ground robots/tasks while the red color represents aerial robots/tasks. Tasks are enclosed in circles while the robots are enclosed in square for ground robots and triangle for aerial robots.

The allocation map of the Distributed Bees algorithm visualises a valid set of allocations where robots are allocated to tasks in different zones. The allocation map of the S+T algorithm also visualises a similar set of allocations but also includes a dual task link which represents the combination of a service and task. For this combination to be assigned to a robot, its service is first assigned to the only capable robot.

Figures-5, 6 show the variance in the average total time both the algorithms take to allocate all the tasks when the number of tasks are increased. The difference between the total allocation times is noticeable when the number of tasks are more than 50. While it is clear that the Distributed Bees algorithm is far more efficient, it is expected that because of the attached service tasks in S+T algorithm, the algorithm is more complex and thus takes longer to process allocations.

Figures-7,8 show the variance of the average time both the algorithms take to allocate any random task when the number of tasks are increased. The difference between allocation time is noticeable right as soon as the tasks are more than 50. This makes the Distributed Bees algorithm perform far more efficiently. Similar to the total average times, the single allocation time for S+T algorithm is pretty steep in comparison but for certain scenarios this difference might be negligible due to the service allocation the algorithm supports.

Figures-9,10 show the time consumed in the reallocation process for both the algorithms when the number of tasks are increased. Following the previous trends,

the difference between the times is noticeable right from the very beginning. The Distributed Bees algorithm is very efficient in reallocating tasks while also keeping the time to a minimum. The Times for the S+T algorithm are very inefficient and in some specific cases were found to be causing reallocation loop issues. This makes using the S+T algorithm with reallocation unfeasible in most cases.

Figures-11,12 show the average utility value of any task allocated by both the algorithms when the number of tasks are increased. Unlike the previous trends, the difference in the quality of allocations are comparable. This is due to the fact the both the algorithms use the grid based allocation process which insures that allocation quality is always prioritized over the other discussed factors. Instead of having non-linear trends, the average utility values highly depend on the randomness of the system. Still the Distributed Bees algorithm performs slightly better on average compared to the S+T algorithm. This difference might not always be important due to the advantages offered by the S+T algorithm.

Overall there are clear differences between both the algorithms in all the categories. Looking at the quantitative differences, the Distributed Bees algorithm performs better in all the categories. Still depending on the situation, it might not be the most optimal algorithm to implement. The S+T algorithm allocates the task fairly competitively with the help of the grid based allocation process and its advantages in form of services are clearly very relevant in real-world Multi Robot Task Allocation scenarios. For such a scenario, the increased total and individual average allocation times might be negligible. However, the reallocation times for S+T algorithm cannot be termed as negligible due to the potential issue of reallocation loops. This can be addressed by removing the reallocation support for the algorithm and sacrificing some of the allocation quality.

5 Conclusions and Future work

This paper highlighted the importance of task allocation algorithms as part of a Multi Robot System which solves a complex problems. The important challenges faced by a task allocation algorithm in a communication-constrained environment are presented and discussed in accordance to the rural firefighting and rescuing scenario. Its argued that not every task allocation algorithm can handle such challenges and according to the scenario, a grid based allocation process is presented which tries to address the most important of these challenges.

To test such the allocation process, a 3D simulation environment is presented which has the characteristics of a real-world rural region. Different types of robots and tasks are used in this environment which each have different capabilities and requirements respectively. The

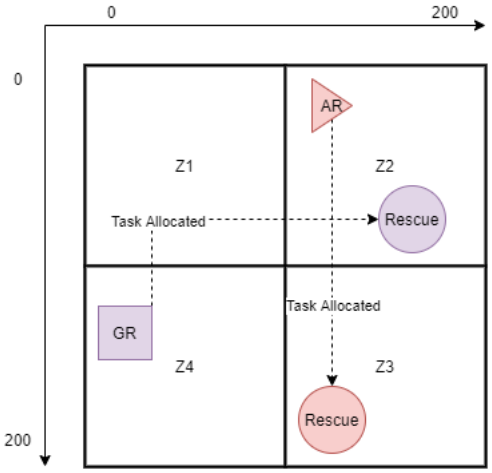

Figure 3: Allocation map generated from the validity testing of DBA.

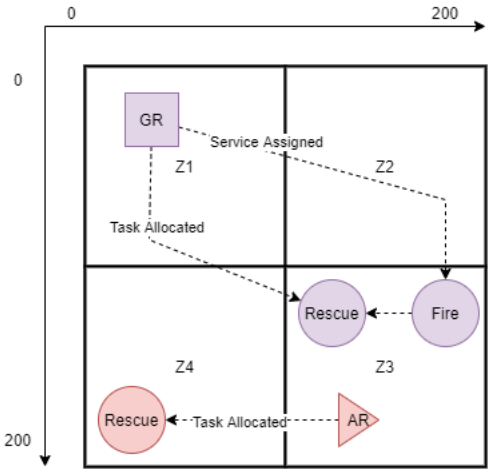

Figure 4: Allocation map generated from the validity testing of SPT.

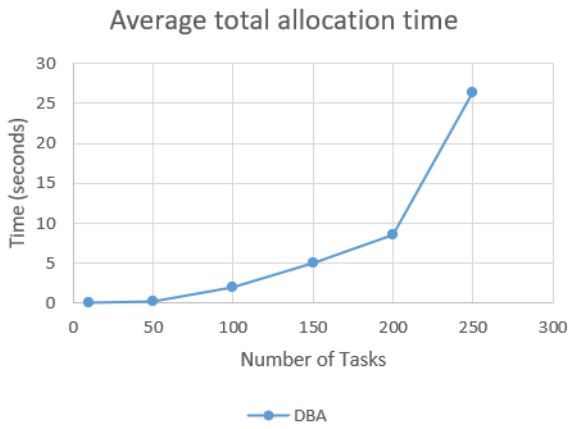

Figure 5: Average time taken to allocate the all tasks in DBA.

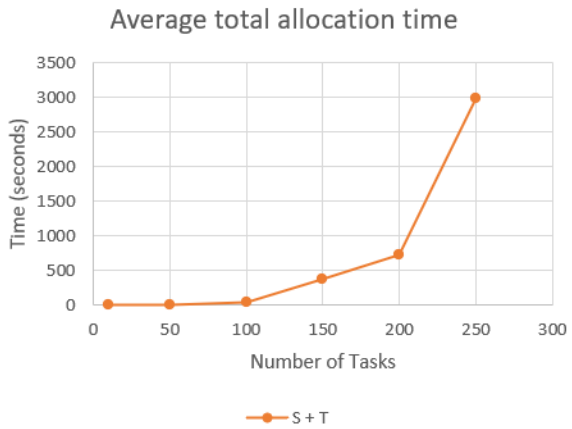

Figure 6: Average time taken to allocate the all tasks in SPT.

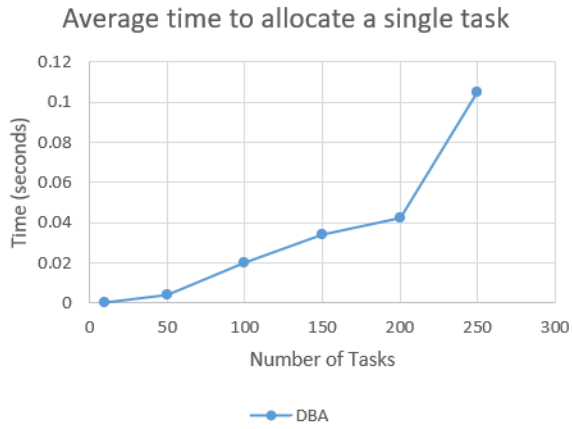

Figure 7: Average time taken to allocate a single task in DBA.

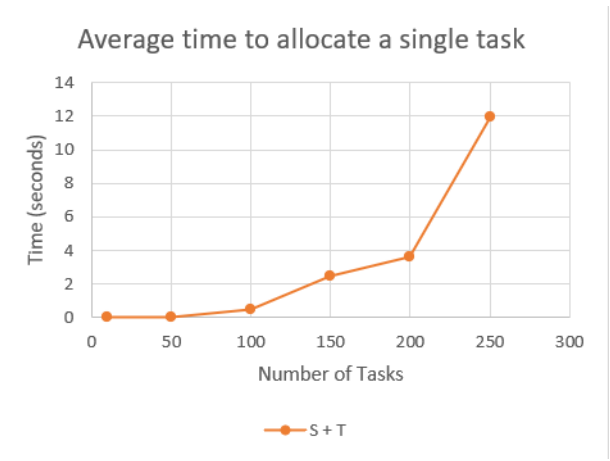

Figure 8: Average time taken to allocate a single task in SPT.

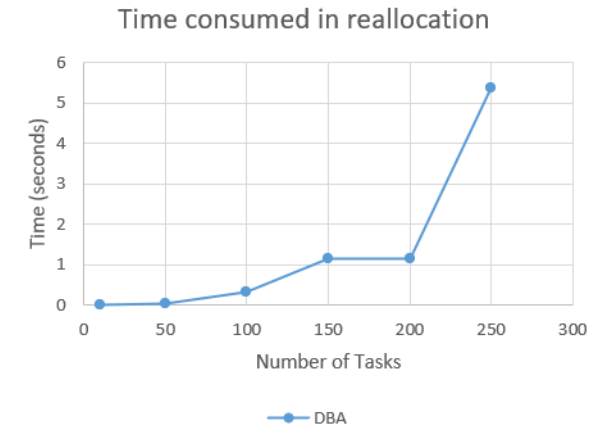

Figure 9: Average time due to the reallocation of tasks in DBA.

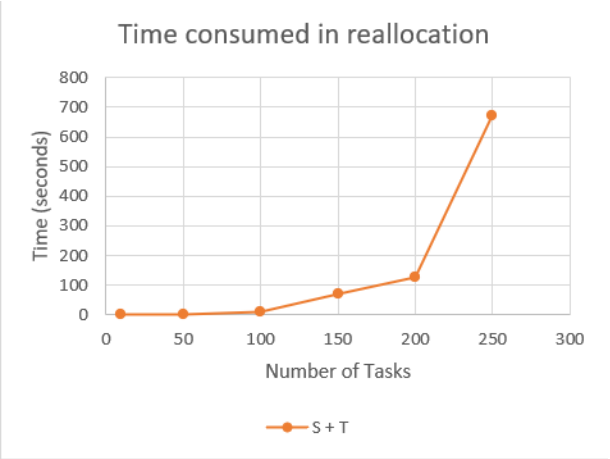

Figure 10: Average time due to the reallocation of tasks in SPT.

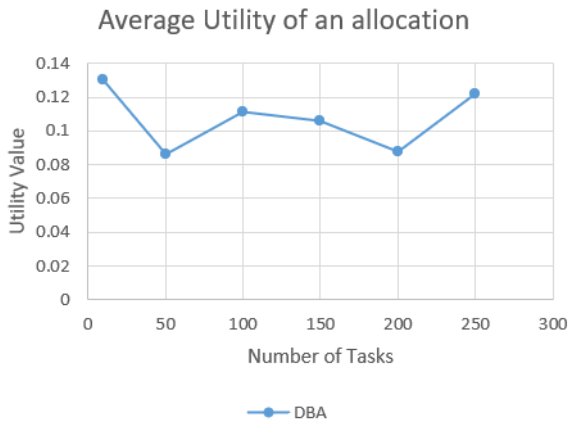

Figure 11: Average utility value of an allocated task in DBA.

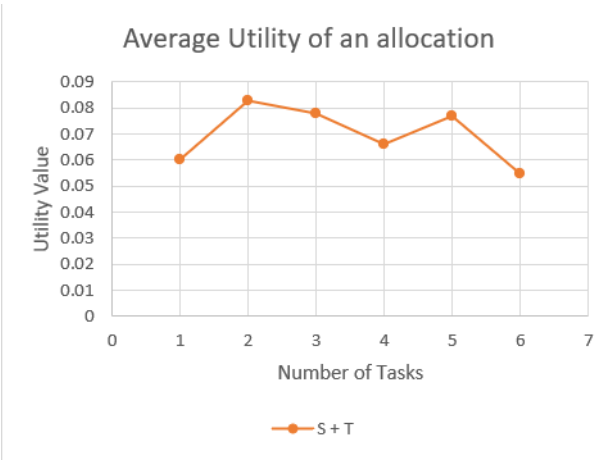

Figure 12: Average utility value of an allocated task in SPT.

grid based allocation process is incorporated in two different types of algorithms and evaluated using different experiments designed to compare the challenges faced in the system. The results from the experiments are discussed according to the comparison metrics and the positives, negatives of each allocation algorithm and the process are highlighted.

With the framework designed for the Grid based allocation process, the work can be expanded to include even more different types of algorithms and varying types of robots and tasks. In addition to that, the allocations themselves can be evaluated by developing a visual simulation which displays the efficiency of such allocations more accurately in terms of completion times and allocation maps.

Acknowledgments

Acknowledgement limited to maximum of 3 sentences.

References

[1] G. Antonelli, F. Arrichiello, and S. Chiaverini, *The null-space-based behavioral control for mobile robots*, in 2005 International Symposium on Computational Intelligence in Robotics and Automation, IEEE, 2005, pp. 15–20.

[2] P. Caloud, W. Choi, J.-C. Latombe, C. Le Pape, and M. Yim, *Indoor automation with many mobile robots*, in EEE International Workshop on Intelligent Robots and Systems, Towards a New Frontier of Applications, IEEE, 1990, pp. 67–72.

[3] A. V. Chandak, B. Sahoo, and A. K. Turuk, *Heuristic task allocation strategies for computational grid*, (2011).

[4] B. Coltin and M. Veloso, *Mobile robot task allocation in hybrid wireless sensor networks*, in 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2010, pp. 2932–2937.

[5] P. Davidsson, S. Johansson, and M. Svahnberg, *Characterization and evaluation of multi-agent system architectural styles*, in International Workshop on Software Engineering for Large-Scale Multi-Agent Systems, Springer, 2005, pp. 179–188.

[6] M. B. Dias, *Traderbots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*, Carnegie Mellon University, 2004.

[7] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, *Market-based multirobot coordination: A survey and analysis*, Proceedings of the IEEE, 94 (2006), pp. 1257–1270.

[8] B. P. Gerkey and M. J. Mataric, *A framework for studying multi-robot task allocation*, (2003).

[9] B. P. Gerkey and M. J. Matarić, *A formal analysis and taxonomy of task allocation in multi-robot systems*, The International journal of robotics research, 23 (2004), pp. 939–954.

[10] M. R. Islam and M. N. Akhtar, *Fuzzy logic based task allocation in ant colonies under grid computing*, in 2017 International Conference on Electrical, Computer and Communication Engineering (ECCE), IEEE, 2017, pp. 22–27.

[11] A. Jevtic, A. Gutiérrez, D. Andina, and M. Jamshidi, *Distributed bees algorithm for task allocation in swarm of robots*, IEEE Systems Journal, 6 (2011), pp. 296–304.

[12] N. Kalra and A. Stentz, *A market approach to tightly-coupled multi-robot coordination: First results*, in Proceedings of the ARL Collaborative Technologies Alliance Symposium, Citeseer, 2003.

[13] A. Khamis, A. Hussein, and A. Elmogy, *Multi-robot task allocation: A review of the state-of-the-art*, Cooperative Robots and Sensor Networks 2015, (2015), pp. 31–51.

[14] K. Konolige, D. Fox, C. Ortiz, A. Agno, M. Eriksen, B. Limketkai, J. Ko, B. Morisset, D. Schulz, and B. Stewart, *Centibots: Very large scale distributed robotic teams*, in Experimental Robotics IX, Springer, 2006, pp. 131–140.

[15] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. J. Kleywegt, S. Koenig, C. A. Tovey, A. Meyerson, and S. Jain, *Auction-Based Multi-Robot Routing.*, in Robotics: Science and Systems, vol. 5, Rome, Italy, 2005, pp. 343–350.

[16] M. T. Long, R. R. Murphy, and L. E. Parker, *Distributed multi-agent diagnosis and recovery from sensor failures*, in Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453), vol. 3, IEEE, 2003, pp. 2506–2513.

[17] A. R. Mosteo and L. Montano, *A survey of multi-robot task allocation*, Instituto de Investigacin en Ingenier\la de Aragn (I3A), Tech. Rep, (2010).

[18] L. E. Parker, D. Rus, and G. S. Sukhatme, *Multiple mobile robot systems*, in Springer Handbook of Robotics, Springer, 2016, pp. 1335–1384.

[19] D. T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, and M. Zaidi, *The bees algorithm—a novel tool for complex optimisation problems*, in Intelligent Production Machines and Systems, Elsevier, 2006, pp. 454–459.

[20] A. Pustowka and E. F. Caicedo, *Market-based task allocation in a multi-robot surveillance system*, in 2012 Brazilian Robotics Symposium and Latin American Robotics Symposium, IEEE, 2012, pp. 185–189.

[21] Y. Rizk, M. Awad, and E. W. Tunstel, *Cooperative heterogeneous multi-robot systems: A survey*, ACM Computing Surveys (CSUR), 52 (2019), pp. 1–31.

[22] J. Schneider, D. Apfelbaum, D. Bagnell, and R. Simmons, *Learning opportunity costs in multi-robot market based planners*, in Proceedings of the 2005 IEEE International Conference on Robotics and Automation, IEEE, 2005, pp. 1151–1156.

[23] A. Stentz and M. B. Dias, *A free market architecture for coordinating multiple robots*, tech. rep., CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1999.

[24] P. Stone and M. Veloso, *Multiagent systems: A survey from a machine learning perspective*, Autonomous Robots, 8 (2000), pp. 345–383.

[25] A. Viguria, I. Maza, and A. Ollero, *SET: An algorithm for distributed multirobot task allocation with dynamic negotiation based on task subsets*, in Proceedings 2007 IEEE International Conference on Robotics and Automation, IEEE, 2007, pp. 3339–3344.

[26] ——, *S+ T: An algorithm for distributed multirobot task allocation based on services for improving robot cooperation*, in 2008 IEEE International Conference on Robotics and Automation, IEEE, 2008, pp. 3163–3168.

[27] P. Yu, R. Xu, M. J. Abramson, S. Li, and Y. Guo, *Bushfires in Australia: A serious health emergency under climate change*, The Lancet Planetary Health, 4 (2020), pp. e7–e8.

[28] R. Zlot and A. Stentz, *Market-based multirobot coordination for complex tasks*, The International Journal of Robotics Research, 25 (2006), pp. 73–101.