A Fast Distributed Auction and Consensus Process for Allocation of Prioritised Tasks in Multi-Robot Systems

Gautham P. Das¹, Martin McGinnity¹, Sonya Coleman¹, and Laxmidhar Behera¹ ²

¹ Intelligent Systems Research Centre, School of Computing and Intelligent Systems, University of Ulster, Magee Campus, Derry, BT48 7JL, N. Ireland, UK; ² Indian Institute of Technology, Kanpur, India

p_das-g@email.ulster.ac.uk, {tm.mcginnity, sa.coleman, l.behera}@ulster.ac.uk

Abstract. The effectiveness of a multi-robot system is directly related to the coordination and cooperation among the robots. Decentralised task allocation algorithms have been proposed by prior researchers as a compromise between the robustness and solution optimality and such algorithms can easily be employed for large teams of robots. We present an auction based decentralised task allocation algorithm, for task allocation in parallel to the task execution, suitable for task allocation in dynamic environments and scalable to large robot groups. The performance of the proposed algorithm under different conditions is compared with other distributed consensus algorithms.

Keywords: multi-robot task allocation, auction based task allocation, decentralised task allocation

1 INTRODUCTION

The performance potential of multi-robot systems (MRS) compared to a single robot has led to increased research interest in MRS. An efficient task allocation algorithm ensures proper cooperation and coordination among the members of the robot group with direct influences on the performance of MRS. Multi-robot task allocation (MRTA) problems can be classified based on the taxonomy proposed by Gerkey and Mataric [1]. Conventional optimisation algorithms and problem formulation in operational research were used in early solutions to MRTA problems, especially for tasks such as foraging and sample collection, where the MRTA problem has been formulated as multiple travelling salesmen problem (mTSP) [2], [3]. Due to the NP hardness [4] of MRTA problems, algorithms are selected based on a trade-off between execution time and solution optimality.

Task allocation can be carried out in a centralised or decentralised manner. Centralised task allocation systems use a central controller to allocate the tasks based on the information from all the robots and sensors in the environment. Decentralised task allocation systems use multiple controllers geographically distributed at different locations. These controllers generate task allocation schemes based on locally available information and exchange the proposed allocation schemes among the
different controllers to reach a collective decision, avoiding any conflicts. Centralised approaches can give an optimal solution as the decision is made using global information, but is not robust as the central controller is the single point of failure. Centralised approaches require large communication networks to connect to the different robots in the environment to collect the information. These drawbacks directly affect the scalability of the centralised approach to large groups of robots. Solutions from decentralised approaches tend to be sub-optimal as the decision is made based on the local information. The distributed task allocation schemes are scalable because of the limited workload on each controller and the limited communication requirements.

Market based approaches such as auctions have received significant attention from robotic research community for MRTA problems [5]. In an auction based system, an auctioneer conducts the auction of all tasks and the robots bid for the tasks based on the estimation of effective returns from execution of the task and costs to be incurred during task execution. The auctions can be single round combinatorial auctions, sequential single item auctions or parallel single item auctions [6]. Based on the allocation time and computation complexity sequential single item auctions and parallel single item auctions are preferred to combinatorial auctions, although combinatorial auctions give optimal solutions.

Market based decentralised task allocation algorithms, namely Consensus Based Auction Algorithm (CBAA) and Consensus Based Bundle Algorithm (CBBA), proposed by Choi et al., [7] have attracted a lot of research interest [8], [9], [10]. CBAA uses parallel single item auctions where each robot bids for a single task and if there are unallocated tasks after allocating one task to each robot, for allocating them the CBAA has to be run iteratively, where as CBBA uses sequential single item auctions to build a bundle of allocated tasks for each robot. Although optimality of solutions given by CBBA is better than the solutions from running CBAA iteratively, in a dynamic environment the allocation may become infeasible as time passes. In CBBA each robot greedily searches for the execution order of task in the bundle of tasks already allocated to the robot, while placing the bid on the task, which is computationally intensive. To accommodate the uncertainties in case of dynamic tasks Bertucelli et al., [8] proposed time depended accuracy of the allocations using CBBA, in which periodic reallocation is carried out. Ponda et al., [9] proposed an alternative approach where the number of tasks in a bundle is limited. This could lower the allocation time when reallocation has to be carried out many times. Mercker et al., [10] proposed an extension to CBBA for multiple tasks of different priorities at the same target location without any dependencies among them and with restricted communication between the robots. All these algorithms face the same problems as CBBA, when applied in a dynamic environment. Moreover, both CBBA and CBAA have not considered prioritised tasks.

In this paper we propose a distributed auction and consensus based task allocation algorithm, with less computational requirements and faster solutions than those of CBBA. The proposed algorithm achieves faster solutions by parallel task allocation and execution and can handle prioritised tasks. The proposed algorithm is suitable for dynamic environment and can be applied to large groups of robots. The rest of the
A Fast Distributed Auction and Consensus Process for Allocation of Prioritised Tasks in Multi-Robot Systems

The paper is organised as follows. Section II defines the task allocation problem and Section III explains the proposed algorithm with the steps involved in the task allocation. In Section IV, results of the proposed algorithm have been compared with those of iterative CBAA for different test cases. Section V concludes the paper, mentioning future research to be carried out to extend this work.

2 Problem Statement

In a multi-robot multi-task situation with a list of tasks $T$ and a list of homogeneous robots $R$, a task allocation algorithm has to allocate tasks to robots without any conflicts; the allocation is conflict free if no more than one robot is allocated any one task. We consider the situation where all tasks are atomic tasks requiring only one robot to execute it. Considering $N_t$ and $N_r$ as the cardinality of $T$ and $R$ respectively, the problem can be stated mathematically as.

Maximise $\sum_{i=1}^{N_t} \sum_{j=1}^{N_r} v_{i,j}(p_i)x_{i,j}$ \hspace{1cm} (1)

Subject to $\sum_{j=1}^{N_r} x_{i,j} \leq L_i, \forall i \in I$ \hspace{1cm} (2)

$\sum_{j=1}^{N_r} x_{i,j} \leq 1, \forall j \in J$ \hspace{1cm} (3)

where $I$ is the set of all robot indices $\{1, 2, ..., N_r\}$ and $J$ is the set of all task $\{1, 2, ..., N_t\}$. $v_{i,j}$, is the non negative reward associated with the execution of task $t_j$ by robot $r_i$ and is a function of the absolute reward from the task and the cost for the traversal of the robot from its current position $p_i$ to the task location.

When $x_{i,j} = 1$, this indicates that task $t_j$ has been allocated to robot $r_i$ and $L_i$ is the maximum number of tasks that can be allocated to the robot $r_i$. The tasks have a static priority ranging from 0 to 5, where 0 denotes the highest priority and 5 the lowest priority. Each robot records the winning bid value of all tasks ($B$), the bid winning robot's index for all tasks ($A$) and the execution status ($E$) of all tasks. If a successful bid is made for any task, the corresponding values are updated. These data are communicated among the robots to reach consensus. During the consensus process, these values are updated with the data from the other robots.

3 Task Allocation

Typically, in consensus based approaches, consensus is reached prior to any tasks being carried out [7], [8], [9], and [10]. In the proposed approach, each robot will bid for its starting task and subsequent tasks are bid for during the execution of the
current task; this readily enables a dynamic multi-robot multi-task allocation algorithm. In our approach the robots bid for the task with the highest bid value, followed by a distributed conflict resolution process. The connected robots exchange the current bidding information; this message passing followed by conflict resolution among robots ensures the system reaches consensus in the task allocation. The different phases involved in the task allocation process are task bidding, conflict resolution and task assignment and execution; each of these are considered in turn in the remainder of this section. The tasks and robots are assumed to be in an environment without any obstacles and the robots move in straight lines to the task location.

3.1 Task Bidding

In this phase each robot independently bids for a task (the one providing maximum reward) from the qualifying tasks. All robots are aware of all the active available tasks. For a task to be considered in the bidding process, it must be active, must not have been previously allocated and the value of the calculated bid must be greater than any current bid on that task by any robot. The earlier mentioned vectors \( B, A \) and \( E \) are referred to and modified during this process. The robots bid for the highest priority tasks initially. If there is more than one task at the highest priority level, the robot will bid for the task with highest reward at that priority level. The bid value \( v_{i,j} \) for task \( t_j \) by robot \( r_i \) is calculated as

\[
v_{i,j} = \lambda^{d_j(p_i)} \cdot C_j \tag{4}
\]

where \( \lambda \) is the discount factor associated with the task to introduce a reward reduction corresponding to the delay in reaching the task location after the bid has been made. \( d_j(p_i) \) is the distance the robot \( r_i \) has to travel to reach the task \( t_j \) from its current position \( p_i \). \( C_j \) is the absolute reward associated with the task.

The bid value vector \( B_i \) of the robot \( r_i \) is the set \( \{ b_{i,j}, \forall i \in I, \forall j \in J \} \) of winning bids of all the tasks as known by robot \( r_i \). Tasks which have a bid by any robot will have a non-negative bid value. The vector \( A_i \) of robot \( r_i \) is the set \( \{ a_{i,j}, \forall i \in I, \forall j \in J \} \) of indices of robots which won the respective tasks known to \( r_i \). The execution status vector \( E_i \) is the set \( \{ e_{i,j}, \forall i \in I, \forall j \in J \} \) containing the execution status of the tasks known to \( r_i \). A value of -1 indicates no winning bid in the case of the task bid value vector, no winning robot in the case of the task allocation vector and the task unallocated status in case of task execution status vector. The other possible values in the task bid value vector are the bids by the robots for the tasks and in the task allocation vector are the robot indices. Within the task execution vector, -2 denotes task completion status and -3 denotes task allocated but not completed status. For a robot to be able to bid for a task it must satisfy equation (2),
which ensures that the robot has not already been allocated its maximum number of tasks. For a task to be biddable, the following conditions must be satisfied.

\[ v_{ij} > b_{ij} ; \quad e_{ij} = -1 \]  

(5)

The conditions in equation (5) ensure that the bid by the robot \( r_i \) for task \( t_j \) is greater than the current bid for the task by any other robot and the task is currently not allocated to any other robot. For any non-negative value of \( b_{ij} \), there will be an \( a_{ij} \), which is the index of the robot with the current maximum bid for task \( t_j \). Among the biddable tasks, the task \( j_{\text{max}} \) is selected using the following criteria.

\[ j_{\text{max}} = \arg \max_j v_{ij} , \forall j \in J \]  

(6)

where \( \arg \max_j \) gives the value of \( j \) corresponding to the maximum value of \( v_{ij} \).

The bidding process is different for the first task than the later tasks. In the Initial bid phase each robot bids to secure only one task and the robots wait until they reach consensus on the current bids. Highest priority tasks among the unallocated tasks are bid initially. During the Later bids, (after having at least one task allocated to each robot), the robot bids in advance for the next task considering the path distance from the current position to the location of the task for which the robot bids for, through the currently executed task location. These bids are continuously updated as the robots are moving towards the current task location and the resultant distance to the next task is getting reduced. During this phase the robots can reach full or partial consensus on the bid, so that it can be easily allocated once the current task is completed. Once the robot successfully identifies its potential next task and bids for it, it moves to the next phase, namely the communication and consensus process.

3.2 Communication and Consensus

The second phase in the proposed algorithm is the communication and consensus process where each robot sends its task vectors to other connected robots and receives other robots’ task vectors in order to clear any conflicts in bids. This process is necessary as the bid value is calculated based only on the local information and there is no central auctioneer to find the highest bidder. The task vectors (task bid value, task allocation and task execution status vectors) are updated during the consensus process, which follows the communication phase. Based on the values which one robot receives from the neighbour robot the possible actions are

1. **Update**: Update values in task vectors \((B, A \text{ and } E)\) corresponding to the task, using the data from the neighbour.
2. **Leave**: No action needs to be taken. Leave without modifying any value in the task vectors.
3. **Reset**: Reset the values in the task vectors \((B, A \text{ and } E)\) corresponding to the task with default values.
4. **Release**: Release the task by adjusting the internal bid tracking.
Assuming robot $r_i$ receives the task vectors from robot $r_k$, the consensus process can be subdivided into two options, one for the task which has been on bid by $r_i$ and the other for all other tasks. In the first consensus process of Bid task, the task which is currently bid on by the robot $r_i$, alone is checked with the task vectors from neighbour robot $r_k$. In the second consensus process of the unbid tasks, all the tasks which are not bid for by the robot $r_i$ are updated by comparing with task vectors from neighbour robot $r_k$. The consensus rules are captured in Table 1 and Table 2. As the proposed algorithm considers bid task and other tasks separately, robots detect and rebid for a new task quickly. The environment information from the other robots is updated regularly to improve decisions during the bidding process.

### Table 1. Consensus Rules for Bid Task

<table>
<thead>
<tr>
<th>Receiver ($\epsilon_{i,j}$)</th>
<th>Sender ($\epsilon_{k,j}$)</th>
<th>Receiver's Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-3</td>
<td>Release and Update</td>
</tr>
<tr>
<td>-2</td>
<td>-2</td>
<td>Release and Update</td>
</tr>
<tr>
<td>-1</td>
<td>if ($b_{i,j} &lt; b_{k,j}$)</td>
<td>Update</td>
</tr>
</tbody>
</table>

### Table 2. Consensus Rules for Unbid Tasks

<table>
<thead>
<tr>
<th>Sender ($\epsilon_{k,j}$)</th>
<th>Receiver ($\epsilon_{i,j}$)</th>
<th>Receiver's Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>-3</td>
<td>-3</td>
<td>Leave</td>
</tr>
<tr>
<td>-2</td>
<td>-2</td>
<td>Leave</td>
</tr>
<tr>
<td>-1</td>
<td>if ($a_{i,j} = -1$)</td>
<td>Update</td>
</tr>
<tr>
<td></td>
<td>if ($a_{i,j} = k$)</td>
<td>Update</td>
</tr>
<tr>
<td>-2</td>
<td>-3</td>
<td>Update</td>
</tr>
<tr>
<td>-2</td>
<td>-2</td>
<td>Leave</td>
</tr>
<tr>
<td>-1</td>
<td>if ($a_{k,j} = -1$)</td>
<td>Update</td>
</tr>
<tr>
<td></td>
<td>if ($a_{i,j} = k$)</td>
<td>Reset</td>
</tr>
</tbody>
</table>

### 3.3 Task Assignment and Execution

Once consensus has been reached by robots regarding the current bids and the robot has completed the previously assigned task, the robot will be assigned the task, after which no other robot will be allowed to bid for that task. The traversal of the robot from its current position at the time of allocation of task to the task location is considered to be the task execution phase. Obstacles in the path and dynamic tasks are not considered at this stage. The robots move along straight lines. During the execution of a task, the robot will actively participate in bidding and consensus to identify the next task to be executed. Only after the completion of the current task, the task identified as the next task to be executed will be assigned to the robot, until then that task can be overbid by another robot.
4 Implementation and Results

The proposed algorithm had been tested extensively using a simulated obstacle free environment of size 100x100m² with randomly placed robots and tasks. Homogeneous robots with constant velocity, fully communicating networks and stationary tasks are considered. Except for the tests for prioritised task allocation, all tasks are of equal priority. Although total distance in case of CBBA is less than that of CBAA, time to reach consensus is more, due to the greedy path building in CBBA. For these reasons iterative CBAA was selected over CBBA for comparison with the proposed algorithm. Results of iterative CBAA and the proposed algorithm have been compared in the following subsections.

4.1 Case 1: All Robots and Tasks Active Initially

Figure 1 (a)–(e) show the comparison of average distance travelled per robot (with at least one task allocated to it) for iterative CBAA and the proposed algorithm for different number of robots and tasks. The average distance for the proposed algorithm is slightly better in the majority of the experiments. Figure 1 (f)–(j) compares the execution time (consensus time + actual execution time) for both algorithms. For large teams of robots the proposed algorithm performs faster than the iterative CBAA because of its parallel task execution and bidding.

4.2 Case 2: Dynamic Environment – Introducing New Tasks

In a real working environment, new tasks can be introduced at any time. As task allocation occurs in parallel to task execution in the proposed approach, newly introduced tasks will be considered alongside the unallocated tasks. In the case of the iterative CBAA, if the system has not already allocated all tasks to the available robots, the new task can be considered in the bidding of remaining tasks. However, if a new task is introduced during execution, then either another bidding and consensus process must be carried out without changing the current allocation or the current allocation can be completely removed and a new bidding process conducted and consensus reached. In the tests, the environment is initialised with 10 robots and 20 tasks. In the tests 10, 20, ..., 80 new tasks were activated in the environment, in groups of 10 tasks, with each group activated after a fixed time interval. Figure 2 (a) compares the average distance travelled and Figure 2(b) compares the execution time in the second set of tests for both algorithms. As the task allocation is carried out dynamically during the task execution, the proposed algorithm performs better than iterative CBAA in these situations and the performance is improved further as the number of added tasks increases.

4.3 Case 3: Dynamic Environment – Introducing New Robots

It is also possible that new robots join the environment. To make the maximum utilisation of the available robots, these new robots also should be allocated some tasks.
Fig. 1. Comparison between proposed algorithm and iterative CBAA when all tasks are known initially: (a)-(e): average distance per active robots for different $N_r$ and $N_t$; (f)-(j): total time of execution for different $N_r$ and $N_t$.

The environment was initialised with 10 robots and 100 tasks. In these tests groups of 10 new robots were activated in the environment. 10, 20, ..., 80 new robots were activated in the environment. Figure 2 (c) compares the average distance travelled and Figure 2 (d) compares the execution time for both algorithms for addition of new robots. In the tests the timing is selected so that new robots are activated before all the
A Fast Distributed Auction and Consensus Process for Allocation of Prioritised Tasks in Multi-Robot Systems

Tasks are executed. As the consensus and bundle creation time is high for the iterative CBAA, its total execution time is high compared to the proposed algorithm, although the average distance travelled in both cases are very similar.

Fig. 2. Comparison between proposed algorithm and iterative CBAA (a) Average distance travelled per robot and (b) Total execution time when new tasks are activated; (c) Average distance per robot and (d) Total execution time when new robots are activated.

Fig. 3. (a) Navigation path to all tasks (b) Task allocation and execution schedule for all robots in the proposed algorithm for prioritised tasks

4.4 Case 4: Prioritised Tasks

As CBAA and CBBA do not consider prioritised tasks, these tests were carried out to test the effectiveness of the proposed algorithm in handling prioritised tasks, rather than comparing the performance with that of another algorithm. A test environment with 5 robots ($r_0\text{--}r_4$) and 20 tasks ($t_0\text{--}t_{19}$) is considered. Tasks $t_5\text{--}t_9$ are the highest priority tasks with priority level 0, tasks $t_{10}\text{--}t_4$ and $t_{10}\text{--}t_{14}$ have priority level 1 and tasks $t_{15}\text{--}t_{19}$ are of priority level 2. The positions of the robots and tasks are randomly initialised. Figure 3 (a) shows the path of each robot and Figure 3(b) shows the schedule of each robot. The first of the two important observations from figure 3(b) is the short progression time to the next task after the completion of each task and the second is the allocation of the highest priority tasks initially.
5 Conclusion & Future Work

In this paper we have proposed a fast and scalable algorithm for the allocation of prioritised tasks in multi-robot multi-task scenarios. The algorithm makes use of a distributed auction and consensus process to allocate the tasks. The performance of the algorithm has been compared with an existing distributed consensus based task allocation algorithm for static and dynamic situations. The proposed algorithm can easily adjust to dynamic environmental conditions and performs better than the other distributed consensus based algorithm. The proposed algorithm can handle allocation of prioritised tasks efficiently.

In future work, heterogeneous robots and tasks, tasks requiring cooperation from multiple robots, dynamic priorities of tasks and task allocation in a cluttered environment will be addressed.

6 References