2019 IEEE 58th Conference on Decision and Control (CDC)
Palais des Congrès et des Expositions Nice Acropolis
Nice, France, December 11-13, 2019

On-the-Fly Decentralized Tasking of Autonomous Vehicles

Tadewos G. Tadewos, Laya Shamgah, and Ali Karimoddini

Abstract— **This paper proposes a cooperative task allocation and execution strategy for a group of agents with different capabilities to accomplish a mission autonomously. For each local agent, a hierarchical and modular Behavior tree (BT) is synthesized to coordinate a sequence of actions to accomplish a task either individually or in collaboration with other agents. To facilitate the coordination among agents and the task assignment process, a market-based auction algorithm is embedded in the developed framework. The details of the developed algorithm are illustrated through different examples, verifying the effectiveness of the proposed approach.**

I. Introduction

Due to recent technological advancements, the adoption of multi-agent systems to accomplish a task is becoming a common trend in many application domains [1]–[3]. While using multi-agent systems provides clear advantages in terms of resilience, cost, speed, and coverage, it creates its own challenges [4]. A major challenge for the deployment of multi-agent systems is the task allocation problem, i.e., assigning tasks to agents with the objective of minimizing the overall cost while maximizing resource utilization.

Different approaches exist in the literature for tasking multi-agent systems including, but not limited to, formal specification-guided tasking [5]–[7], event-based supervisory control [8]–[11], and mixed-integer linear programming (MILP) [12]. Most of these task allocation methods rely on offline computations, requiring information about the environment and all tasks in advance. Therefore, if new tasks are introduced or if the environment changes, these methods require redoing the entire process to include the new changes.

To address these challenges, in this paper, by adopting a market based auction [13], [14] algorithm, we develop an on-the-fly tasking mechanism by synthesizing Behavior Trees (BTs) as local coordinators for autonomous vehicles. By definition, BTs are graphical mathematical models for the execution of tasks with inherent hierarchical, modular, and reactive properties [15], [16], and hence, can serve as building blocks for autonomous decision making. These features make BTs as ideal candidates to construct a modularized, reactive, and scalable control structures to meet the goal of a mission. In [17], assuming that a decomposable task and the associated central (global) BT exist, a heuristic algorithm has been used to generate local BTs for each agent to meet the goal of a task. However, the proposed mechanize does not explain how the global BT is generated. In [18], a procedure is provided to obtain a BT for a single robot that meets a mission specification. Combining these two algorithms, it is possible to use the algorithm proposed in [18] to generate the global BT, and then, employ the method in [17] to decompose the global BT to distribute the tasks among available agents. However, this approach is not salable, it does not consider the capacity and capability of the available agents beforehand, and it is not computationally efficient as the process consists of extra steps for obtaining the global BT followed by decomposition stages.

To address these challenges, in this paper, we develop a novel approach to directly synthesize the local BTs in a distributed setting. The proposed method adopts a two-level market based auctioning mechanism to distributively synthesize BTs for each agent with the objective of minimizing the overall cost. In the proposed method, collaboration among agents is needed only if a single robot cannot do the assigned task alone, thus resource utilization is maximized leaving other available robots for handling new tasks. The developed method is illustrated and verified via several examples.

The rest of the paper is organized as follows. The background and necessary preliminaries are provided in Section II. In Section III, the BT-based decentralized multi-agent coordination problem is formulated. Section IV describes our proposed approach for synthesizing decentralized BTs in detail. Section V illustrates the proposed method using several case studies. Finally, Section VI concludes the paper.

II. Background

Behavior Tree (BT) is an effective tool to capture the decision making mechanism for an autonomous vehicle. As the name implies, the structure of a BT is based on a tree that can be represented with a directed acyclic graph (DAG) to demonstrate control flows from top to bottom (parent to child) among different types of nodes. At the top of the tree, a *root* node exists that provides the activation clk for all other nodes. In addition to a *root* node, a BT may contain *leaf* and *composite* nodes.

Leaf nodes are terminal nodes that could act as a sensing unit (*condition* nodes) or as a computing/actuation unit (*action* nodes). A *condition* node checks the state of the robot or the environment and return *success* only if the condition is true. An action node performs an operation that modifies/change the state of the robots or the environment. Similarly, the action node returns success only if the operation is completed. Fig. 1.b, shows activation of action A_1 if condition C_1 is true.

The authors are with the Department of Electrical and Computer Engineering, North Carolina Agricultural and Technical State University, Greensboro, NC 27411 USA.

Corresponding author: A. Karimoddini. Address: 1601 East Market Street, Department of Electrical and Computer Engineering North Carolina A&T State University Greensboro, NC, US 27411. Email: akarimod@ncat.edu, Tel: (+1)336-285-3313.

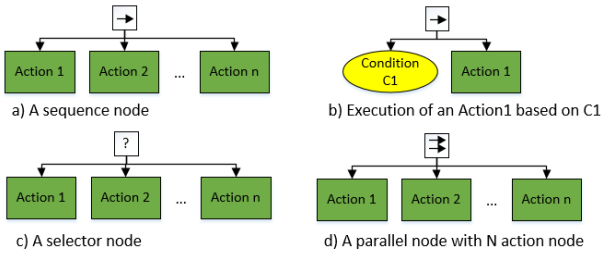

Fig. 1: Building blocks of Behavior Trees

Composite nodes provide the capability to compose multiple child nodes under a single parent. A *sequence* node composes actions or sub-trees in an ordered fashion, where activation is passed from one child to the next only if the current node is completed with *success*. Otherwise, a failure status is returned by the *sequence* node. A *selector* node composes actions or sub-trees with priority where activation of the next child is possible if the current node returns a *failure* status. A selector node return *success* if only one child node succeeds, otherwise it returns *failure*. A *parallel* node provides the capability to execute actions/sub-trees simultaneously. The success of a *parallel* node is determined by a natural number N, which specifies how many children are needed to succeed for the node to return *success*. If N number of children succeed, then the node returns *success*, otherwise it returns *failure*. Figures 1.a, 1.c, and 1.d show the graphical representation of *sequence*, *selector* and *parallel* nodes, respectively.

Generally, the execution of a BT is initiated by the root node which sends a tick (enabling signal) with a certain frequency to its children. Then, the enabled child activates another child or returns its execution status as *running*, *failure*, or *success* to its immediate parent. In this way, the actions are executed from the bottom left of the BT, returning *success/failure/running* to their parents.

By the proper combination of leaf nodes (actions and condition nodes), and composite nodes (*sequence, selector or parallel* nodes), a complex BT structure can be constructed that is both modular and reactive that can effectively meet the goal of a mission.

III. Problem Formulation

In this section, we use BTs to formulate the coordination and tasking for multi-agent systems over the following components:

1) The set R which includes a team of robots $R = \{R_1, \cdots, R_M\}$, where $M \in \mathbb{N}$ is the number of agents. Here, the terms agents, robots, and vehicles are used interchangeably.
2) The set A is the global action bank and contains a set of actions A_k, $k = 1, \cdots, L$, where $L \in \mathbb{N}$ is the total number of actions. We define a set of action capability indicators $\hat{a}_{ik}$, $i = 1, \cdots, M$, $k = 1, \cdots, L$, for which $\hat{a}_{ik} = 1$ if the robot R_i can accomplish Action A_k, otherwise $\hat{a}_{ik} = 0$. Here, the robots are assumed to perform single action at a time.
3) The set T which includes a set of complex Tasks (a task can be decomposed into multiple set of actions that could satisfy the same task goal in different ways [19]) T_j, $j = 1, \cdots, N$, where $N \in \mathbb{N}$ is the number of tasks. The accomplishment of each task, T_j, can be captured by meeting a condition C_j. For example, if the task T_1 is to "reach a goal region", then C_1 is "being at the goal region." We also define a set of task indicators x_{ij}, $i = 1, \cdots, M$, $j = 1, \cdots, N$, for which $x_{ij} = 1$ if the task T_j is assigned to R_i to handle it individually or in collaboration with other robots, otherwise $x_{ij} = 0$. Similarly, we define a set of action assignment indicators $\hat{x}_{ijk}$, $i = 1, \cdots, M$, $j = 1, \cdots, N$, $k = 1, \cdots, L$, for which $x_{ijk} = 1$ if action A_k of R_i is assigned for task completion of T_j. To reach the "goal" of a task T_j, depending on the agent that is responsible to handle the task, a series of actions from the action bank A should be completed, where the last action should meet C_j. In our proposed framework, only a robot that can accomplish an action which meets C_j, can be a candidate for being selected to handle T_j. Such a robot can complete an action to meet C_j, and may delegate the prerequisite actions to other agents if necessary. Further, we define the indicators $\hat{a}_{ijk}$, $i = 1, \cdots, M$, $j = 1, \cdots, N$, $k = 1, \cdots, L$, for which $\hat{a}_{ijk} = 1$ if action A_k from robot R_i is needed to complete the task T_j, otherwise $\hat{a}_{ijk} = 0$.
4) The set F includes a set of values $f_{ij} : R \times T \to \mathbb{R}^+$ to describe the cost of handling the task T_j by R_i based on performance, energy, and proximity. Robot R_i can accomplish the actions in T_j individually or delegate the actions to other robots if necessary. We define a cost function $\hat{f}_{ik} \in \mathbb{N}$, which indicates the cost of accomplishing an individual action A_k by the agent R_i.
5) We define the set $\hat{C}$ which includes a set of preconditions $\hat{c}_{ikp}$, $i = 1, \cdots, M$, $k = 1, \cdots, L$, and $p = 1, \cdots, P_k$, where P_k is the number of preconditions for action A_k, and $\hat{c}_{ikp}$ specifies pth preconditions for completing action A_k by robot R_i. We also define action status indicator $\hat{c}_{ik}$ where $\hat{c}_{ik} = 1$ if action A_k is executed and completed by R_i, otherwise $\hat{c}_{ik} = 0$.
6) Consider a discrete clock clk with a granularity of $1sec$, i.e., $clk = clk + 1$ (this can be of smaller step sizes if needed). The clock clk represents the elapsed time starting from the first task announcement. Then, we define $\triangle t_{ik}$, $i = 1, \cdots, M$, $k = 1, \cdots, L$, which represents the duration the agent R_i needs to complete the action A_k. We also define an action timeline indicator t_{io}, $i = 1, \cdots, M$, $o = 1, \cdots O$, where $O \in N$ is the last sample time, and $t_{io} = 1$ during the time that R_i is assigned to perform one of the actions A_*, which takes R_i for $\triangle t_{i*}$ time units.
7) We define an operation $R_i \models con$ which checks if the agent $R_i, i = 1, \cdots, M$, satisfies the condition con at its current state, where the condition con can be a

condition for a task, i.e., C_j, or a precondition for an action, c_{ikp}.

Further, to do automatic tasking for multi-agent systems, similar to [18], we need to make the following assumptions:

$Assumption$ 1: Each agent can verify if an action has succeeded, failed or if it is running.

$Assumption$ 2: Each agent can verify if a condition is true or false.

$Assumption$ 3: For each goal and for each initial configuration of the agents, there exists a sequence of actions that can be taken by the agents leading to the achievement of the goal. This assumption guarantees that each goal is achievable at least by one of the agents.

$Assumption$ 4: The effect of the dynamic environment can void the accomplishment of the actions at most a finite number of times. This assumption is made to avoid sticking in a live-lock of repeating an action and being voided by the environment over and over, preventing the agent to achieve its goal.

$Assumption$ 5: Given two actions A_i and A_j, if the execution of A_i requires the execution of A_j, A_j must not require the execution of A_i. This assumption prevents deadlocks due to cyclic dependency.

$Assumption$ 6: All actions are ultimately reversible. That is, each action can be undone through a finite sequence of actions.

$Assumption$ 7: For each action, there exists at least one agent to achieve it, which can be accomplished by a low-level controller embedded in the agent in a finite time.

Now, given $R,\ T,\ F,\ A, C$, and $\hat{C}$, and making assumptions $1-7$, the tasking problem for multi-agent systems can be stated as:

Problem 1: Consider a $Mission$ consists of several tasks $T_j,\ j = 1, \cdots, N$, to be completed by a set of robots $R_i,\ i = 1, \cdots, M$, that (some of them) are capable of accomplishing the actions $A_k,\ k = 1, \cdots, L$, within $\triangle t_{ik}$ time units to achieve the mission. Also, consider that there is no order and dependency among the tasks, other than the order in which tasks are issued (one at a time). Synthesize decentralized BT_i to coordinate the individual robots R_i to collectively achieve a set of tasks T_k.

IV. Automatic Behavior Tree Synthesis

To address Problem 1, we propose a decentralized method for generating the local BTs by combining a market-based auctioning algorithm with a reactive BT synthesis technique, so that the generated local BTs can collectively satisfy the mission specification.

A. Task Assignment for Coordination of Multi Agent Systems

To fairly assign tasks and avoid conflicts, we adopt a two-level market-based auctioning algorithm. Generally in a market-based auctioning, even-though there is a collaboration among agents we assume each agent acts on its own interest, i.e. to maximize the reward or to minimize cost. An auctioning process has four steps, starting with a task announcement by the coordinator (announcement stage), followed by the bidding stage where capable agents send a bid. Based on the cost, the auctioneer selects the best agent (the selection stage) and finalize the auction by forming a contract with the selected agent (contract stage). In the proposed framework the $Mission\ Controller$ (MC) announces a task T_j, where capable agents (agents that can meet C_j), $R_i,\ i = 1, \cdots, M$, participate in the bid. To complete the task T_{ij}, the candidate agent has to identify the sequence of actions either from the local action bank or by delegation, where these actions are used to estimate the total cost f_{ij} before issuing the bid. Based on the estimated cost f_{ij} from each agent, the MC selects an agent and form a contract. Mathematically, this is equivalent to:

$$\begin{aligned} &\min_{x_{ij}} \sum_{i}^{M} f_{ij} x_{ij},\ \forall j \\ &subject\ to \sum_{i}^{M} x_{ij} = 1\ \forall j \\ &x_{ij} \in \{0, 1\},\ \forall i, j \end{aligned} \tag{1}$$

where f_{ij} is the cost of task T_j when handled by R_i and x_{ij} is an indicator that task T_j is assigned to R_i.

If an agent delegate an action to complete a task, then the agent has to act as the auctioneer and perform a second level auctioning to identify a suitable agent. Therefore, the total cost for a task is the sum of local and delegated actions: $f_{ij} = \sum_{k=1}^{L} \hat{a}_{ijk}(\hat{a}_{ik}\hat{f}_{ik} + (1-\hat{a}_{ik}) f_D(ijk)), \forall i, j$, where $f_D(ijk)$ is the cost of the delegated action A_k for the task T_j by R_i, provided that the involved robots are available to complete the actions at the time they are needed. To check availability of the robot, we introduce the function $\nabla(t_{io}, T_j, A_k)$ where $clk(T_j, A_k)$ represents the time that the action A_k is needed for the task T_j. If $t_{io} = 0$ for $clk \geq clk(T_j, A_k) + \triangle t_{ik}$, then $\nabla(t_{io}, T_j, A_k) = 1$, otherwise it returns 0. In addition, $\nabla(t_{io}, T_j, A_k)$ returns the nearest time slot that the agent R_i can accomplish an action. This indeed is equivalent to the following minimization:

$$\begin{aligned} f_D(ijk) = \min_{\hat{x}_{djk}} &\sum_{d}^{M} \hat{x}_{djk}\hat{f}_{dk}, \forall k \\ &d = 1 \cdots M,\ d \neq i, \\ subject\ to\ &\sum_{d}^{M} \hat{x}_{djk} = 1\ \forall j, k, \\ &\nabla(t_{do}, T_j, A_k) = 1 \end{aligned} \tag{2}$$

where $\hat{f}_{dk}$ is the cost of action A_k when done by R_d and $\hat{x}_{djk}$ indicates if action A_k of task T_j is assigned to agent R_d or not.

Once an action or a task is assigned to an agent, the availability indicator t_{*o} is updated from 0 to 1 for $\triangle t_{*o}$ using the function $\hat{\nabla}(t_{*o}, \triangle t_{*k})$ to avoid double assignment.

B. Decentralized Behavior tree synthesis algorithm

The overall procedure to generate the BTs for individual agents is explained in Algorithms $1-3$. First, the mission

controller announces a task T_j (level-I auctioning). Then, any capable agent estimates the task cost and sends a bid. The estimation of the cost is calculated starting from the goal and recursively identifying the precondition of the successor action until the action can be done at the current state of the robot. Upon receiving the bid from the agents, the MC selects the best agent and form a contract (Algorithm 3). The winning agent R_i synthesizes a BT using Algorithm 1 while Algorithm 2 is used to identify actions locally or by delegation (level-II auctioning) to meet the conditions needed to complete the task.

Algorithm 1: *Main BT Synthesis and Execution*

```
1  function MainBTSynthesisandExecution (C_j) ;
   Input : C_j : Condition for assigned task of agent i
   Output: T_ij = Syntheisized BT
2  T_ij ← C_j
     // Start the BT for task T_j from the condition C_j, which is
     used to cheek if the task is completed or not
3  T_all_i ← Parallel(T_ij, T_all_i)
     // T_all_i represents all BTs of an agent running in parallel
     to execute multiple tasks including bidding and auctioning
4  while True do
5      do
6          r, ĉ_ik ← Execute(T_ij)
7          if R_i ⊨ C_j then
8              Set x_ij = 0  // Task T_j is completed
9              break // End execution of T_ij
10         end
11     while r = Executable;
12     c_if ← GetConditionToExpand(T_ij)
         //Identify the the reason why T_i is not executable
13     T_ij, T_subtree_ij ← ExpandTree(T_ij, c_if)
         //Resolve the cause by Algorithm 2
14     while Conflict(T_ij) do
15         T_ij ← IncreasePriority(T_subtree_ij)
16     end
17 end
```

Assume that the task T_j is assigned to the robot R_i as it can meet the condition C_j. Algorithm 1 then synthesizes the local BTs. Algorithm 1 starts from the "goal" input, which describes the condition for the accomplishment of a "task" indicated by the condition C_j (Line 1). By first assigning the condition C_j to the BT (Line 2) (this condition will be used to determine if the task is completed or not), the algorithm iteratively updates the BT until a sequence of actions is obtained which as a whole realizes the task and achieves the goal (Lines 4-17). Since each task requires its own BT, to execute multiple tasks, the BTs for each task are composed in parallel with the existing BTs, T_{all_i} (Line 3). In a *do while* loop, the BT actions are tested to determine whether they are executable (Lines 5-11). If the condition $R_i \models C_j$ is satisfied by the execution of the BT, the agent is free to accept a new task (Lines 7-10). Otherwise, if the BT is not executable, Line 12 identifies the *cause* of failure, c_{if}. The identified *cause* will become a condition in a subtree to resolve the problem by finding alternative actions or other agents (Line 13), as will be described in Algorithm 2. After updating the BT, due to the addition of a new subtree, $\mathcal{T}_{subtree_{ij}}$, a conflict could arise. To resolve the conflict, the function $conflict(\mathcal{T}_{ij})$ increases the priority of $\mathcal{T}_{subtree_{ij}}$ by moving the subtree toward the left. As an example, in response to avoid an obstacle the robot decides to pick an obstacle (object), but picking up an object has to be done if the robot arm is free.

Algorithm 2 essentially synthesizes a subtree that satisfies the condition c_{if}. In Line 2 of Algorithm 2, the function $GetlocalActionwithPrecondtion(.)$ returns the optimal action, which satisfies the condition c_{if}. If the returned action is not empty, then the identified action A_k along with its preconditions, $\hat{c}_{ikp}$, are composed by a sequence node to form $\mathcal{T}_{seq_{ij}}$ (Lines 5-9). Further, $\mathcal{T}_{seq_{ij}}$ is composed with $\mathcal{T}_{sel_{ij}}$, defined as c_{if} (Line 3), by a selector node, to enforce the execution of $\mathcal{T}_{seq_{ij}}$ only in situations where c_{if} is not satisfied (Line 10). To avoid double assignment, the time-line and availability indicators for R_i are also updated (Lines 11-12). However, if no local action exists, the $AuctioningModule$ (similar to Algorithm 3) is activated to conduct an auction in pursuit of finding an agent that can accomplish c_{if} (Lines 14-16). Finally, the condition c_{if} is replaced with a sub-tree that can meet c_{if} (Line 17).

Algorithm 2: *Expand Behavior Tree Module For R_i*

```
1  function ExpandBT (T_ij, c_if);
   Input : c_if = condition (cause) for T_j not being executable
   Output: T_ij = Expanded BT
2  A_k ← GetLocalActionwithPrecondtion(c_if)
     // Identify local actions that satisfy c_if
3  T_sel_ij ← c_if
4  if GetLocalActionwithPrecondtion(c_if) ≠ ∅ then
5      c_ik = GetPrecondtionforAction(A_k)
6      for c_ikp in c_ik do
7          T_seq_ij → Sequence(T_seq_ij, c_ikp)
             // sequence BT with the condition of action
8      end
9      T_seq_ij ← Sequence(T_seq_ij, A_k)
         // Generate a sequence subtree containing action
           A_k and its preconditions
10     T_sel_ij ← Selector(T_sel_ij, T_seq_ij)
11     ∇̂(t_io, T_j, A_k)  // t_io is set to 1 for △t_ik time units
12     set x̂_ijk = 1 // Action A_k of T_j is assigned to R_i
13 end
14 else
15     AuctionModule_L2(c_if)
         //If there is no action to meet a condition,
          initialize the Level IIAuction Module for delegation
16 end
17 T_ij ← Substitute(T_ij, c_if, T_sel_ij)
     // add the subtree T_sel_ij to T_ij replacing c_if
18 return T_ij, T_sel_ij
```

Algorithm 3 performs an auction to find a suitable agent following a standard market based auctioning mechanism (Lines 2-6). The auction terminates with a contract (Line 6).

Algorithm 3: *Auctioning Module*

1 function AuctioningModule (c_{if}) ;
  Input : c_{if} : *condition to be delegated*
2 $selected_f \leftarrow \emptyset$
3 $Announcing(c_{if})$ //broadcasting condition c_{if}
4 s = $ReceiveSubmission()$
  // agents with the spesfied action replies
5 $selected_f = Selection(s)$
  // choose the agent that minimizes cost $\hat{f}_{if}$
6 $Contract(selected_f, c_{if})$

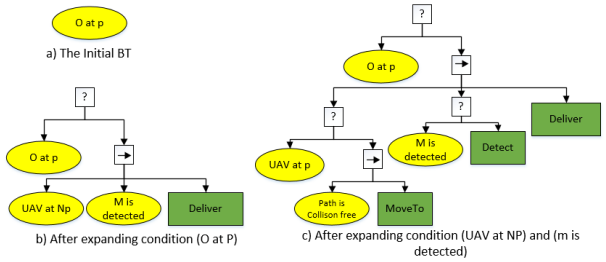

Fig. 2: Synthesizing a BT for a UAV to search and deliver an object to a particular position

V. CASE STUDY

A. Single Agent: Search and Delivery UAV mission

The mission objective is to deliver an object o at a specific place marked by m near position p. The UAV has to search for the marking m in close vicinity of p, N_p, before delivering the object o. Then, the problem is given the action bank in Table I, generate a BT using Algorithms 1 & 2 to achieve the task.

Algorithm 1 starts from the goal, "o at p", i.e., the object o should be at position p, as shown in Fig. 2a. Since initially the goal is not satisfied yet and the generated BT (Line 6 of Alg1) is not executable, the function $GetCondtionsToExpand$ is called to identify the preconditions (Line 12). From Table I, the $Deliver$ action can meet the precondition of Algorithm1 and hence, the $ExpandBt$ function (Line 13) uses this action to update the BT by composing the conditions of $Deliver$ action via a sequence node and the goal by a selector node (Lines 4-13 of Alg2) as shown in Fig. 2b. Again since the preconditions, $uav\ at\ N_p$ and $m\ is\ detected$, are not true, they have to be expanded, following the same procedure, by their corresponding actions $MoveTo$ and $Detect$ as shown in Fig. 2c.

Global Action Template			
No	Action	Precondition	Effect
1	MoveTo($p, path$)	$path\ is\ collsionfree$	$uav\ at\ p$
2	Detect(m)	$uav\ at\ N_p$	$m\ is\ detected$
3	Deliver(i, m)	$uav\ at\ Np$ $m\ is\ detected$	$o\ at\ p$

TABLE I: Action templates for case study V-A

Mission			
No	Task	Condition	Sequence of actions
1	T_1	C_1	A_1, A_3, A_2
2	T_2	C_2	A_6, A_4, A_3
3	T_3	C_3	A_5, A_1
4	T_4	C_4	A_4
5	T_5	C_5	$A_2, A_8\ or\ A_3, A_7$

TABLE II: Mission tasks expanded using Algorithm 1

Resource		
No	Agent	Agent capability - $\{A_k,(\hat{f}_{ij}, \triangle t_{ik}\})$
1	UAV_1	$\{A_1,(0.1,3)\},\{A_2,(0.2,2)\ \{A_8,(0.6,2)\}$
2	UAV_2	$\{A_3,(0.5,2)\},\{A_5,(0.4,3)\},\{A_7,(0.4,2)\}$
3	UAV_3	$\{A_3,(0.7,1)\},\{A_4,(0.7,1)\},\{A_6,(1,4)\}$

TABLE III: Agents capability along with the cost ,$\hat{f}_{ij}$, and duration, $\triangle t_{ij}$, of an action A_k

B. Multiple Agent Multiple Task

Given multiple, $R = \{UAV1_1, UAV_2, UAV_3\}$, along with their capabilities described by the action bank in Table III, our aim is to synthesize BTs in a decentralized way to satisfy the tasks listed in Column 2 of Table II. To avoid repeating the procedure of generating a sequence of actions for each task, explained in section V-A, each task is expanded to a sequence of actions which is described in Column 3 of Table IV. Then, following Algorithms 1 - 3, the details of the BT generation are given in Table IV. As an example, consider the expanded task $T1$ with the sequence A_1, A_3, A_2 (Row 1 of table IV), where UAV_1 is the only candidate and winner of T_1 (since only UAV_1 can do the last action, i.e., A_2). However, in task T_1, action A_2 cannot be executed by UAV_1 because UAV_1 cannot perform action A_3 which precedes action A_2. Hence UAV_1 initiates a level-two auctioning to assign action A_3 (Row 2 of table IV), where UAV_2 wins the auction with minimum cost. Now the final action A_1 in T_1 can be handled by UAV_1. This concludes the action assignment for the task T_1 with a total cost of 0.8 in the time interval $[1, 8]$ (Row 3 of Table IV). The assignment of all tasks follows the same procedure. Sometimes it may be the case that agents are not available at the time a task is requested, like in T_3. When T_3 is assigned, even though UAV_2 is free, the available time before A_3 of T_1 starts execution is not enough to complete the action A_5 of T_3 completely. So action A_5 is deferred to a later time (Row 9 of Table IV). This can be seen more clearly in Figure 3 which shows the tasks and UAVs' assignments along with time axis. The final task, T_5, can be accomplished by UAV_1

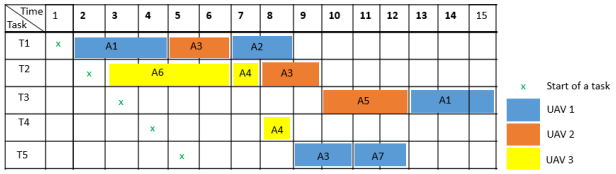

Fig. 3: Task assignment along a timeline

Auctioning steps for assigning the tasks T1, $\cdots$, T5					
Step	Task/ Action	time	Auctioneer	Candidates	Contract
1	T_1	1	$MC,\ L1$	$\{UAV_1\ : A_2,\ \hat{f}_{12} = 0.2\}$	-
2	A_3	4	$UAV_1,\ L2$	$\{UAV_2:\ \hat{f}_{23} = 0.5,\ clk(T_1, A_3) = [4-6]\}$ $\{UAV_3:\ \hat{f}_{33} = 0.7,\ clk(T_1, A_3) = [4-5]\}$	UAV_2
3	T_1		$MC,\ L1$	-	$\{UAV_1 : f_{11} = 0.8,\ clk(T_1, A_*) = [1-8])\}$
4	T_2	2	$MC,\ L1$	$\{UAV_2\ : A_3, \hat{f}_{23} = 0.5\}$ $\{UAV_3\ : A_3, \hat{f}_{33} = 0.7\}$	-
5	A_6	2	$UAV_2,\ L2$	$\{UAV_3\ : \hat{f}_{36} = 1,\ clk(T_2, A_6) = [2-6]\}$	UAV_3
6	A_4	6	$UAV_2,\ L2$	$\{UAV_3\ : \hat{f}_{34} = 0.7,\ clk(T_2, A_4) = [6-7]\}$	UAV_3
7	T_2		$MC,\ L1$	$\{UAV_2 : f_{22} = 2.2,\ clk(T_2, A_*) = [2-9]\}$ $\{UAV_3\ : f_{32} = 2.4,\ clk(T_2, A_*) = [2-8]\}$	$\{UAV_2 : f_{22} = 2.2,\ clk(T_2, A_*) = [2-9])\}$
8	T_3	3	$MC,\ L1$	$\{UAV_1\ : A_1, \hat{f}_{11} = 0.1\}$	-
9	A_5	3	$UAV_1,\ L2$	$\{UAV_2:\ \hat{f}_{15} = 0.5,\ clk(T_2, A_5) = [9-12]\}$	UAV_2
10	T_3		$MC,\ L1$	$\{UAV_1\ : f_{13} = 0.6,\ clk(T_3, A_*) = [10-15]\}$	$\{UAV_1 : f_{13} = 0.6,\ clk(T_3, A_*) = [3-15]\}$
11	T_4	4	$MC,\ L1$	$\{UAV_3\ : A_4, \hat{f}_{34} = 0.7\}$	$\{UAV_3 : f_{34} = 0.7,\ clk(T_4, A_*) = [4-8])\}$
12	T_5	5	$MC,\ L1$	$\{UAV_1\ : f_{15} = 0.8,\ clk(T_5, A_8) = [9-12]\}$ $\{UAV_2\ : f_{25} = 0.9,\ clk(T_5, A_*) = [13-16])\}$	$\{UAV_2 : f_{25} = 0.9,\ clk(T_5, A_*) = [5-12])\}$

TABLE IV: Task Assignment: T_1 represents task 1, "$MC,\ L1$" represents level-one auctioning by the mission controller and "$UAV_*, L2$" represents level-two auctioning by an agent

and UAV_2 in a non-unique way. This shows that tasks are not necessarily a fixed sequence of actions rather multiple capable agents can do a task in different ways to meet the goal.

VI. CONCLUSION

This work developed a new BT-based automatic tasking, synthesis, and execution framework for the coordination of heterogeneous agents with different capabilities to meet the goal of a series of tasks. In the proposed framework there are two-levels of auctioning where agents compete to win either a task (has to be expanded into a sequence of actions) or an action. Further, collaboration among agents is on a need basis, i.e., if an agent lacks the capability to perform an action, that action could be completed by delegation. Future work includes implementation of the proposed framework using the Robot Operating System (ROS) on real robots, and performing complexity analysis.

ACKNOWLEDGMENT

The authors would like to acknowledge the support from the National Science Foundation under the award number 1832110 and Air Force Research Laboratory and OSD under agreement number FA8750-15-2-0116.

REFERENCES

[1] A. Macwan, J. Vilela, G. Nejat, and B. Benhabib, "A multirobot path-planning strategy for autonomous wilderness search and rescue," *IEEE transactions on cybernetics*, vol. 45, no. 9, pp. 1784–1797, 2014.

[2] K. Vinh, S. Gebreyohannes, and A. Karimoddini, "An area-decomposition based approach for cooperative tasking and coordination of uavs in a search and coverage mission," in *2019 IEEE Aerospace Conference*, March 2019, pp. 1–8.

[3] L. Shamgah, T. G. Tadewos, A. Karimoddini, and A. Homaifar, "Path planning and control of autonomous vehicles in dynamic reach-avoid scenarios," in *2018 IEEE Conference on Control Technology and Applications (CCTA)*, Aug 2018, pp. 88–93.

[4] R. M. Murray, "Recent Research in Cooperative Control of Multivehicle Systems," *Journal of Dynamic Systems, Measurement, and Control*, vol. 129, no. 5, pp. 571–583, 05 2007. [Online]. Available: https://doi.org/10.1115/1.2766721

[5] A. Ulusoy, S. L. Smith, X. C. Ding, C. Belta, and D. Rus, "Optimality and robustness in multi-robot path planning with temporal logic constraints," *The International Journal of Robotics Research*, vol. 32, no. 8, pp. 889–911, 2013.

[6] L. Shamgah, T. G. Tadewos, A. Karimoddini, and A. Homaifar, "A symbolic approach for multi-target dynamic reach-avoid problem," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, June 2018, pp. 1022–1027.

[7] I. Filippidis, D. V. Dimarogonas, and K. J. Kyriakopoulos, "Decentralized multi-agent control from local ltl specifications," in *2012 IEEE 51st IEEE Conference on Decision and Control (CDC)*, Dec 2012, pp. 6235–6240.

[8] Y. Liu, M. Ficocelli, and G. Nejat, "A supervisory control method for multi-robot task allocation in urban search and rescue," in *2015 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Oct 2015, pp. 1–6.

[9] P. Ramadge and W. Wonham, "The control of discrete event systems," vol. 77, no. 1, pp. 81–98, 01 1989.

[10] M. Karimadini, A. Karimoddini, and H. Lin, "Modular cooperative tasking for multi-agent systems," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, June 2018, pp. 618–623.

[11] M. Karimadini, H. Lin, and A. Karimoddini, "Cooperative tasking for deterministic specification automata," *Asian Journal of Control*, vol. 18, no. 6, pp. 2078–2087, 2016.

[12] M. Darrah, W. Niland, and B. Stolarik, "Multiple uav dynamic task allocation using mixed integer linear programming in a sead mission," in *Infotech@ Aerospace*, 2005, p. 7164.

[13] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 1, Sep. 2004, pp. 698–705 vol.1.

[14] N. Kalra, R. Zlot, M. B. Dias, and A. Stentz, "Market-based multirobot coordination: A comprehensive survey and analysis," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 2005.

[15] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ögren, "Towards a unified behavior trees framework for robot control," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 5420–5427.

[16] T. G. Tadewos, L. Shamgah, , and A. Karimoddini, "Automatic safe behaviour tree synthesis for autonomous agents," in *Proc. of 58th IEEE Conference on Decision and Control (CDC)*, 2019.

[17] M. Colledanchise, A. Marzinotto, D. V. Dimarogonas, and P. Oegren, "The advantages of using behavior trees in mult-robot systems," in *Proceedings of ISR 2016: 47st International Symposium on Robotics*. VDE, 2016, pp. 1–8.

[18] M. Colledanchise, D. Almeida, and P. Ogren, "Towards blended reactive planning and acting using behavior trees," *arXiv preprint arXiv:1611.00230*, 11 2016.

[19] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International Journal of Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013. [Online]. Available: https://doi.org/10.1177/0278364913496484