Solving Hierarchical Auctions with HTN Planning

Antoine Milot,[1,2,3] Estelle Chauveau,[2] Simon Lacroix,[1] Charles Lesire[3]
[1] LAAS-CNRS, Université de Toulouse, CNRS, 7, Avenue du Colonel Roche, 31031 Toulouse, France
[2] Naval Research, Naval Group, 99 Avenue Pierre-Gilles de Gennes, 83190 Ollioules, France
[3] ONERA/DTIS, Université de Toulouse, 2 avenue Edouard Belin, 31055 Toulouse, France
antoine.milot@laas.fr, estelle.chauveau@naval-group.com, simon.lacroix@laas.fr, charles.lesire@onera.fr

Abstract

This paper presents a preliminary approach to solve the Multi-Robot Task Allocation problem through hierarchical auctions combined with the use of HTN planning. We present the global approach and the challenges arisen by partially-ordered HTNs through some examples. We finally outline some options to integrate such constraints in the allocation scheme.

Introduction

Deploying multi-robot systems for complex missions, such as post-catastrophic situation assessment or submarine mine-hunting, requires to reason about the tasks that the robots must perform, depending on their capabilities and the current situation. We then need to solve a Multi-Robot Task Allocation (MRTA) problem (Gerkey and Matarić 2004). Given a set of n robot $R = (r_1, \dots, r_n)$ and a set of s tasks $Q = (t_1, \dots, t_s)$, solving the MRTA problem consists in finding an allocation $A : Q \rightarrow R$, i.e. allocate each task $t \in Q$ to a robot $r \in R$.

When the environment is highly dynamic, tasks have to be reallocated regularly, due to the impossibility for some robot to achieve allocated tasks, or to the arrival of new tasks to achieve. The auction-based approaches have been extensively considered to approximately solve the MRTA problem in such environments (Dias et al. 2006).

While simple auction schemes only allocate tasks one after the other, hence not yielding optimal allocations, near-optimal allocation schemes need to resort to *combinatorial auctions*, i.e., auctions where each robot can bid over any subset of the tasks to allocate. However, solving the Winner Determination Problem (WDP), which is done by the auctioneer based on robots' bids, becomes untractable for combinatorial auctions. A good balance between expressing combinatorial auctions and solving the corresponding WDP has been proposed through *hierarchical auctions*, where subsets of tasks on which robots can bid are limited to nodes of the task decomposition.

Hierarchical auctions have been used to allow robots to bid on abstract tasks, which has the advantage for the bidders to account for local constraints on primitive tasks (Zlot and Stentz 2006; Liu et al. 2013; Khamis, Elmogy, and Karray 2011). However, in these approaches, greedy Breadth-First-Search (BFS) based algorithms have been proposed to solve the WDP. Consequently, they cannot handle causal constraints between tasks nor correctly manage ordering constraints.

In a former work, we laid the groundwork of an auction-based approach that allocates hierarchical tasks (Milot et al. 2021). We implemented a first prototype and tested it on totally-ordered coverage problems for multiple underwater robots. We obtained good performances in term of solution quality and computation time with respect to the state-of-the-art. Encouraged by the results of this proof of concept, we formalize in this paper the approach with causal and ordering constraints.

This approach relies on HTN planning (Bercher, Alford, and Höller 2019) to both estimate individual bids and solve the WDP. While integrating HTN planning in the hierarchical auction scheme is quite straightforward when HTN problems are totally ordered, managing partial-order problems is more challenging.

The next section summarizes related work. We then describe the general approach and detail the steps of the process for totally-ordered problems. Finally, we illustrate the current limits of the approach for partially-ordered tasks and we draw the first ideas to integrate these constraints into the proposed approach.

Related Work

Auction-based approaches to handle the MRTA problem have been explored for a long time (Dias et al. 2006), mainly because of their simplicity and their ability to handle dynamic events and unreliable communications (Otte, Kuhlman, and Sofge 2019).

The most basic scheme involves Single-Item (SI) auctions (Koenig, Keskinocak, and Tovey 2010): the *auctioneer* agent, which is responsible of the allocation, has only one item to allocate. This scheme then follows these steps: (1) the *announcement*, when the auctioneer opens the auction by broadcasting the information on the item for sale to the *bidders*; (2) the *bids estimation*, in which each bidder estimates the cost associated to the item and sends back its bid to the auctioneer; (3) the *Winner Determination*, where the auctioneer decides to which bidder the item is allocated; and (4) the *reward* announcement, in which the auctioneer announces which bidder won the auction. In the SI scheme, each bidder produces a single bid for the auctioned item,

and the winner determination simply consists of selecting the best bid.

A direct improvement of this scheme are Sequential Single-Item (SSI) auctions: the auctioneer announces a list of items for sale, and each bidder produces a bid for each item of this list. If some items remain not allocated after solving the WDP, the auctioneer starts a new round with the remaining items. The process goes on until all items are allocated or a stop criterion is reached. While SSI allows robots to prioritize some items over others (Kalra, Ferguson, and Stentz 2005; Dias, Ghanem, and Stentz 2005; Botelho and Alami 1999), it does not allow to express the dependencies between the bids. Indeed, a strong assumption of SI and SSI schemes is the independence of the bids, which results in allocating at most one item per robot at each round.

However, such dependencies may be mandatory to express in complex problems, when the tasks (i.e. the items) have temporal or causal constraints between them. Nunes and Gini (2015) have proposed a sequential auction scheme for temporally constrained tasks, where the auctioneer maintains a Simple Temporal Network (STN) of the items for sale. However, this approach is used for the allocation of tasks of an already computed STN, while in our approach we aim at solving both the allocation and the selection of the tasks to perform in order to fulfill a mission objective.

Zlot and Stentz (2006) proposed an auction scheme for *hierarchical tasks*, in which an item consists of an AND/OR tree that decomposes a complex task into sub-tasks. Bidders can then express bids on any node of this tree, and the auctioneer can decide to allocate a complete sub-tree to a robot in a single round. A direct benefit of this approach is to better interleave task decomposition and allocation. Consequently, by selling sub-trees, this approach allows bidders to take account of tasks' dependencies in their bids (e.g. by placing a more interesting bid on an abstract task than on the sum of its individual tasks). In addition, OR nodes in the tree induce bidders to make choices. Therefore, the MRTA problem to solve (by allocating nodes in the tree) becomes also a planning problem. This feature allows to handle difficult problems where choices are needed. Finally, by constraining hierarchically the possible sets of tasks, the hierarchical auctions reduce the burden of classic combinatorial auctions where a bid can be expressed on any subset of tasks.

This approach has been extended to allow bidders to buy and resell tasks to others, possibly proposing a new decomposition (Liu et al. 2013; Khamis, Elmogy, and Karray 2011). However, in these approaches, the WDP is solved by a greedy BFS-like algorithm, where task allocations are locally decided, without reasoning on the global task decomposition tree. While it provides an efficient algorithm in terms of computation time, the quality of the allocations are questionable. Moreover, these approaches do not consider temporal nor causal constraints between tasks, and due to the specific WDP algorithm, these constraints cannot be easily integrated in the approach.

Prerequisites

Notations

In this paper, we use the following notations, inspired from the ones used by Erol, Hendler, and Nau (1994); Höller et al. (2020). By $\mathcal{L}$, we denote a first-order language composed of finite sets of constants, predicate, primitive and compound task symbols, an infinite set of variable symbols, and an infinite set of labels denoted by $\mathbb{L}$. Given a set of terms $x_1, \dots, x_k$ issued from this language, and s a task symbol, we denote by $t = s(x_1, \dots, x_k)$ a *task* (also called *task instance*). A task t can be decomposed by a method $m = (t, tn)$, where $tn = (L, \prec, \alpha)$ is a task network where $L \subset \mathbb{L}$ is a set of labels, $\prec$ is a strict partial order over L and $\alpha : L \to X$ maps labels to the method sub-tasks. We then denote a planning domain $\mathcal{D}$ as $(\mathcal{L}, T_P, T_C, M)$ where $\mathcal{L}$ is the underlying language, T_P and T_C are the sets of primitive and compound tasks, and M the set of decomposition methods. Finally we write a planning problem as $\mathcal{P} = (\mathcal{D}, s_I, tn_I)$ where s_I is the initial state, and tn_I is the initial task network.

Solving a problem $\mathcal{P} = (\mathcal{D}, s_I, tn_I)$ consists in finding a *solution* task network tn such that tn is *primitive* and *executable* in s_I, i.e., there is a sequence of tn tasks, that respects the ordering constraints, in which the preconditions of a task are valid in the state resulting from applying the previous task.

Illustrative example

To illustrate our approach, we consider a *BorderDelivery* problem, inspired from the *transport* and *logistics* problems of the IPC2020 (Behnke et al. 2019). The goal of the *BorderDelivery* problem is to move two packages from an area *Ext* to an area *Storage* and check one of these packages in an area *Check* before bringing it back to *Storage*. To move the packages from *Ext* to *Storage*, robots can bring both packages at the same time or bring each package one by one. The locations of packages are included in a predicate *at(pkg, location)*. Thus, preconditions of the tasks to allocate verify this predicate and their effects change it, leading to causal relations between tasks.

The corresponding domain and problem formulated in Hierarchical Domain Definition Language (HDDL) are shown respectively on Listings 1 and 2. This minimal example allows to highlight the key points of our approach and to shine a light on challenges with partially ordered problems. We first explain the approach working under total order assumption.

General Approach

Our approach uses a SSI auctions scheme, similarly to (Zlot and Stentz 2006), but we use HTN planning at critical steps of the process, and then build our approach on HTN structures. This process is depicted in Figure 1.

Our approach relies strongly on a duality *global/local*. The global part corresponds to the multi-robot level, it consists in describing *what* to allocate and to *who*. This part is embodied by the task tree sent by the auctioneer and shared with all robots participating to the auction. On the other

```
(define (domain BorderDelivery)
   (:requirements :typing)
   (:types
     location locatable - object
     package - locatable)
   (:constants
     storage ext check - location
     package-0 package-1 - package)
   (:predicates
     (at ?x - locatable ?v - location))
   (:task random-check :parameters ())
   (:task store-packages :parameters (?p1
     ?p2 - package ?l1 ?l2 - location))
   (:method m-random-check
     :parameters (?p - package)
     :task (random-check)
     :ordered-subtasks (and
       (bring-new-package ?p storage check)
       (bring-new-package ?p check storage)
       ))
   (:method m-store-packages-one
     :parameters (?p1 ?p2 - package
       ?l1 ?l2 - location)
     :task (store-packages ?p1 ?p2 ?l1 ?l2)
     :ordered-subtasks (and
       (bring-new-package ?p1 ?l1 ?l2)
       (bring-new-package ?p2 ?l1 ?l2)))
   (:method m-store-packages-all
     :parameters (?p1 ?p2 - package
       ?l1 ?l2 - location)
     :task (store-packages ?p1 ?p2 ?l1 ?l2)
     :ordered-subtasks (and
       (bring-all-packages ?p1 ?p2 ?l1 ?l2)
       ))
   (:action bring-new-package
     :parameters (?p - package
       ?l1 ?l2 - location)
     :precondition (and
         (at ?p ?l1))
     :effect (and
         (not (at ?p ?l1))
         (at ?p ?l2)))
   (:action bring-all-packages
     :parameters (?p1 ?p2 - package
       ?l1 ?l2 - location)
     :precondition (and
         (at ?p1 ?l1)
         (at ?p2 ?l1))
     :effect (and
         (not (at ?p1 ?l1))
         (not (at ?p2 ?l1))
         (at ?p1 ?l2)
         (at ?p2 ?l2))))
```

Listing 1: HDDL description of the *BorderDelivery* domain.

hand, the local part corresponds to the *how* a particular robot can accomplish these tasks. Also, we assume that all multi-robot effects are included in the global part and local actions do not impact them.

An auction is initialized by the definition of an *Auction problem* that corresponds to the root task to be decomposed

```
(define (problem pb)
   (:domain BorderDelivery)
   (:htn
      :subtasks (and
         (t1 (store-packages package-0
            package-1 ext storage))
         (t2 (random-check)))
      :ordering (and (< t1 t2))
   (:init
      (at package-0 ext)
      (at package-1 ext)))
```

Listing 2: HDDL description of the *BorderDelivery* problem.

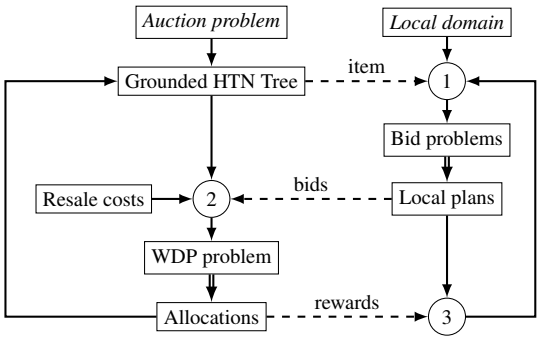

Figure 1: Protocol description. Rectangles represent information or data structures managed by the agents. Blocks on the left are managed by the auctioneer, blocks on the right by each bidder. Dashed arrows represent data exchanged between the auctioneer and the bidders. Circles indicate processes that aggregate information to build new structures. Double arrows indicate calls to a HTN planner.

and allocated. In order to start the allocation process, we first have to determine all the possible tasks to be allocated. To do so, we build a *Grounded HTN tree* $\mathcal{H}$ from the auction problem.

This *Grounded HTN tree* is a structure representing the grounded tasks and methods of the problem similarly to the Task Decomposition Graph (TDG) (Bercher et al. 2017). However, the TDG does not allow to differenciate task instances (i.e. a task symbol with associated parameters) that occur several times while in the *Grounded HTN tree* each task instance is labeled. This need for labeling the task instances comes from both ensuring that our robots reason over the same elements and that the mission is entirely fulfilled.

The auctioneer then sends $\mathcal{H}$ as an item for sale to the bidders. $\mathcal{H}$ represents the task decomposition of the *Multi-robot problem*. Each robot can bid on each task of the decomposition, depending on its ability to perform them.

In order to produce a bid on a labeled task in $\mathcal{H}$, the bidder will plan it to estimate a cost. The solution to this planning problem will be composed of the bidder's own *local* actions (which will increment the cost). To this aim, the bid-

der needs to express how a task in $\mathcal{H}$ can be accomplished regarding its own capacities: this is achieved thanks to HTN planning. The basic idea behind it is that we extend $\mathcal{H}$ with HTNs representing the bidder's local capacities and then use a HTN solver to estimate a bid. The merging of $\mathcal{H}$ with the bidder's *local domain* is done preserving the structure of $\mathcal{H}$ and the associated preconditions and effects on the multi-robot problem. This is accomplished by applying a protocol turning specific primitives tasks of $\mathcal{H}$ into abstract tasks.

Consequently, each bidder produces, for each task to estimate, an estimation problem. This problem is specifically built from the received item and its local domain for the task to estimate and is solved to determine its bid value (process ①). These problems are then solved by a HTN planner to produce the bid values that correspond to the cost of the local plan associated to the task.

Then, bids received from the bidders need to be integrated in order to find an allocation (i.e. a set of winning bids). To this aim, the auctioneer extends $\mathcal{H}$ by including bids as new decompositions of the concerned tasks, which produces a *WDP problem* (process ②). This problem is then solved by a HTN planner to decide which tasks of $\mathcal{H}$ are allocated to which robot. Finally, the result of the allocation is dispatched to the bidders.

Depending on the received bids, it may be possible to still have unallocated tasks at the end of the auction round. In this case $\mathcal{H}$ is modified to account for the allocated tasks. Then a new round is started by sending a new item for sale (with the updated HTN tree) to the bidders.

Hierarchical auctions on totally-ordered HTNs

When the auction problem is totally ordered, the process is sound. We detail the steps in this section. Then, we show the challenges raised when considering a partially ordered auction problem.

As we consider totally ordered problems, we need to add several constraints in our Border Delivery example. These are precedence constraints between every tasks. For example in our *BorderDelivery* problem, we have to store all packages before checking one. These precedence constraints are encoded in the corresponding decomposition methods.

Auction Initialisation

Our approach aims to solve an MRTA problem defined by an auction problem $\mathcal{P}_{auc} = (\mathcal{D}_{auc}, s_I, tn_I)$ formulated by the auctioneer. $\mathcal{D}_{auc}$ is a domain describing the high-level decomposition of tasks at the multi-robot scale. In fact, this domain focuses on the decomposition of tasks that can be allocated, i.e. describing *what* can be allocated without taking into account *how* a robot will accomplish the allocated tasks in terms of its proper actions (*goto, load, unload...*). s_I is the initial state determined by the auctioneer and tn_I is the initial task network of the problem. As a reminder, we consider that all multi-robot effects are included in this problem description and that local actions do not impact them.

Solving $\mathcal{P}_{auc}$ means solving the MRTA problem associated to this auction. However, as the decision scheme is decentralized, when robots bid on a task we need to determine to which instance of this task in the final multi-robot plan this bid is associated. Therefore, we cannot only reason on $\mathcal{P}_{auc}$ and we need to identify each task instance in the problem.

To do so, we use $\mathcal{P}_{auc}$ as an input of the auction scheme. From this auction problem, we build a *grounded HTN tree* $\mathcal{H}$ (algorithm 1). The grounded HTN tree describes hierarchical decomposition of tasks while labels bring unicity and identify them. During an auction round, robots rely on labels to share information relative to the grounded HTN tree.

Algorithm 1: *BuildGroundedHtnTree*

Input: $\mathcal{P}_{auc}$
Output: $\mathcal{H} = (V_T, V_M, E), labels$
$\mathcal{G} \leftarrow TDG(\mathcal{P}_{auc})$
if $\mathcal{G}$ *has cycles* **then return** error;
Let X be an empty First-In-First-Out list
Let l_{top} be a new unique label
$V_T \leftarrow \{(l_{top}, top)\}$, where top is the root task of $\mathcal{G}$
$labels \leftarrow \emptyset$; $V_M \leftarrow \emptyset$; $E \leftarrow \emptyset$
$X.push((l_{top}, top))$
while X *is not empty* **do**
  $(l, t) \leftarrow X.pop()$
  forall *method* $m \in \mathcal{G}$ *such that* $m = (t, (L, \prec, \alpha))$ **do**
    $V_M \leftarrow V_M \cup \{(l, m)\}$
    $E \leftarrow E \cup \{((l, t), (l, m))\}$
    forall $u \in L$ **do**
      Let v be a new unique label
      $V_T \leftarrow V_T \cup \{(v, \alpha(u))\}$
      $E \leftarrow E \cup \{((l, m), (v, \alpha(u)))\}$
      $labels \leftarrow labels \cup \{(l, u, v)\}$
      $X.push((v, \alpha(u)))$

This algorithm first builds a TDG from the auction problem. However, as the TDG does not allow to differenciate multiple occurences of a task instance the algorithm goes through the TDG in a breadth-first search way while labeling each task instance. In order to accomplish this translation from the TDG to the *grounded HTN tree*, the TDG must not have cycles, otherwise the algorithm will never end. We verify this condition with the TDG properties.

We then initialize $\mathcal{H}$ by adding the root task top, with a unique label l_{top} to the set of task vertices. Then, we expand $\mathcal{H}$ by creating, for each labeled task (l, t), and each decomposition method m of t in the TDG, a labeled method (l, m) in the set of method vertices. Finally, for each sub-task u of m, we create a new unique label v, and the corresponding labeled sub-task to $\mathcal{H}$, and store the label mapping to the $labels$ set. $labels$ will contain tuples (l_i, l_j, l_k) representing that sub-task with label l_j in the decomposition method of task labeled l_i has label l_k in $\mathcal{H}$. Figure 2 represents the resulting grounded HTN tree for a *BorderDelivery* auction problem.

At each round, the auctioneer sends $\mathcal{H}$ to each bidder, along with complementary information from the auction problem. An *item for sale* δ is then defined as a tuple

Figure 2: $\mathcal{H}$ for a *BorderDelivery* problem. Rounded rectangles are labeled tasks, hexagons are labeled methods.

$(\mathcal{H}_\delta, s_\delta, L_\delta)$, where $\mathcal{H}_\delta$ is a finite grounded HTN tree representing the hierarchical decomposition between labeled tasks and the associated precedence constraints (included in the decomposition methods); s_δ is a set of atomic formulas on the constants in $\mathcal{H}_\delta$; L_δ is the set of task labels in $\mathcal{H}_\delta$ that are *sellable*, i.e. on which robots can produce bids.

For example at the first round of the *BorderDelivery* problem, the auctioneer sends δ_1 with $\mathcal{H}_{\delta_1}$ (illustrated on Figure 2), s_{δ_1} which contains *at(pkg, location)* predicates specifying the initial locations of the packages, and $L_{\delta_1} = \{l_0, l_1, l_2, l_3, l_4, l_5, l_6, l_7\}$.

Estimating bids with HTN planning

Once an item for sale δ is received, the bidder must compute a bid for each feasible labeled task among L_δ, i.e. each task executable by the bidder. The bid valuation corresponds to the cost of performing this task. Therefore, for each $l \in L_\delta$, we build a planning problem $\mathcal{P}_l = (\mathcal{D}_l, s_l, tn_l)$, and ask a HTN planner to solve it.

To compute this estimation, robots must indicate *how* they can perform primitive tasks of $\mathcal{H}_\delta$. They may indeed need to perform specific actions, like moving to the locations of packages, activating sensors or actuators to grab packages, etc. We consider that the descriptions of the tasks specific to each robot are defined in a *local problem*. This local problem has the following constraints: first, it must share the tasks that the robot can decompose locally with the auction problem (i.e., symbols and constants). Second, we reasonably consider that the local and auction problems do not share any predicate. It allows indeed to consider that a multi-robot task on the auction problem cannot depend on a predicate that could be validated only by a specific local action of a single robot. Conversely, that robot's local actions cannot depend on effects of multi-robot tasks.

The aggregation of the item and the local problem corresponds to step ① in Figure 1. It consists, for each label $l_\delta \in L_\delta$ on which the bidder wants to produce a bid, in extending $\mathcal{H}_\delta$ in the following way: each leaf task l of $\mathcal{H}$ that is either a descendent of l_δ, or was already allocated to the bidder in previous rounds, is replaced by an abstract task with exactly one ordered method, made of (in order) (1) a $start(l)$ action representing the beginning of l, and containing only the preconditions of l, (2) an abstract task that will be decomposed in the bidder local problem, and (3) an $end(l)$ action representing the end of l, and containing only the effects of l. Other primitive tasks are accounted a cost of 0. Essentially, $start(l)$ and $end(l)$ allow to insert the robot's local actions between the preconditions and effects of the task l that were defined in $\mathcal{H}$.

Figure 3 illustrates this process by showing the expansion of the task labeled l_4 from the *BorderDelivery* problem. Tasks in yellow correspond to the decomposition introduced earlier. Orange tasks correspond to the TDG built with a robot's local domain. For example, solving the bid estimation problem can lead to a sequence of actions such as $[start(l_4) \rightarrow get\text{-}to(Ext) \rightarrow \cdots \rightarrow end(l_4)]$.

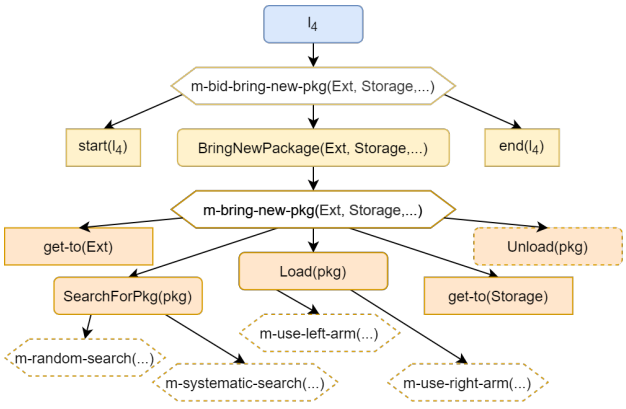

Figure 3: Illustration of the decomposition of labeled task l_4 for the bid estimation of the *BorderDelivery* problem. Hexagons represent methods, rounded rectangles represent compound tasks while sharp rectangles are primitive tasks, dotted lines represent nodes with hidden decompositions. Blue nodes represent elements from $\mathcal{H}$. The specific decomposition for the bid estimation is represented in yellow. Local decomposition nodes are represented in orange.

Moreover, the bidder builds s_l by merging s_δ with its proper initial state including its own local information (e.g. its current position, energy...). tn_l is defined by the root task of $\mathcal{H}_\delta$.

Finally, for each labeled task on which we want to esti-

mate a bid, we synthesize a domain and problem (including the decomposition mentioned earlier, and the initial states both from the item for sale and the local problem). We rely on the HDDL formalism to define these domain and problem. In this problem, primitive actions are either the start and end actions introduced earlier, which have a null cost, or bidder's local actions, which may have a non-unit cost. A HTN planner then solves this problem, producing both a local plan to complete the task and a cost associated to this plan. Actions start and end in the plan allow to check preconditions and trigger effects of the task.

Finally, the bidder returns to the auctioneer a set B containing triplets (l, c, P), where $l \in L_\delta$ is a task label, c the associated bid value, i.e. the cost returned by the planner, and P the set of $\mathcal{H}_\delta$ task labels present in the local plan returned by the planner.

Solving the WDP with HTN planning

At the end of an auction round, the auctioneer solves the WDP to determine the task allocations. Given L_δ the set of labels of the tasks for sale and $R = (r_1, \ldots, r_n)$ the set of bidders participating to the auction, we denote B_{r_i} the set of received bids from the bidder r_i bearing on labels in L_δ. The set of all received bids is denoted by $\mathcal{B} = \bigcup_{r_i \in R} B_{r_i}$. Solving the WDP consists in finding a set of winning bids $\mathcal{B}_w \subset \mathcal{B}$ such that each bidder wins at most one bid and each $l \in L_\delta$ is won by at most one bidder.

In order to determine this set of winning bids, the auctioneer needs to build $\mathcal{D}_{wdp}$ and $\mathcal{P}_{wdp} = (\mathcal{D}_{wdp}, s_{wdp}, tn_{wdp})$ a planning domain and problem dedicated to the WDP solving.

To do so, we complete $\mathcal{H}_\delta$ with the bids received from the bidders (step ② in Figure 1). We do this by adding to every task one method for every bid. Each of these methods corresponds to a unique action whose cost is the bid value. However, we must integrate the SSI scheme constraint on bid independence, that enforces that at most one task (whatever its abstraction level) can be allocated to each robot.

Encoding this property in the actions corresponding to the bids is quite straightforward. Each task on which a bid has been received must then also be decomposed into an action corresponding to not allocating this task (that we call *resell*). The pattern is quite similar to the one used for bidding: we add a new method for each task on which we have bids, decomposed with $start$ and end actions for primitive tasks, and an abstract $AllocateOrResell$ task that will itself be decomposed into one method per bid plus one resell method.

Figure 4 illustrates this decomposition for task with label l_4, for which robot r_1 bids. A new decomposition leading to the task allocation or reselling was added with the task network consisting in doing $start(l_4) \rightarrow AllocateOrResell(l_4) \rightarrow end(l_4)$. And $AllocateOrResell(l_4)$ can be decomposed as a resell action or an allocate action to robots that have bid on this task, here r_1 submitted a bid on task l_4.

Similarly to the estimation process, tn_{wdp} is defined by the root task of $\mathcal{H}_\delta$. However, the initial state s_{wdp} is the same as s_δ.

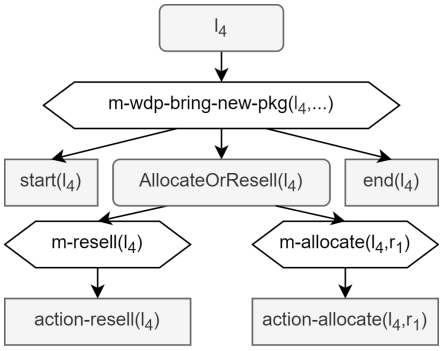

Figure 4: Decomposition of task l_4 integrating received bids for the WDP of the *BorderDelivery* problem. Hexagons represent methods, rounded rectangles represent compound task while sharp rectangles are primitive tasks.

As we cannot force the WDP to allocate exactly one task to each robot, we cannot set the cost of the $resell$ actions to 0: these tasks would systematically be resold when optimizing the allocation plan. Consequently, we need to define in a clever way the *resale costs* to have a sound and efficient allocation behavior. While these costs could be defined per domain, we have proposed two strategies to define them, based on the received bids on a task:

- a pessimistic strategy about the bids that will be received in future rounds for this task, encouraging to allocate the task at this round; the resale cost is then set as the maximum of the bid values for this task plus one;
- an optimistic strategy that hopes for better bids in the future; the resale cost is set as the minimum of the bids values for this task plus one.

In summary, the auctioneer creates, from $\mathcal{H}_\delta$ and the received bids, a new planning problem $\mathcal{P}_{wdp}$ corresponding to the WDP of this auction round. This problem, translated to a HDDL domain and problem, is then solved by a HTN planner, which returns a plan containing either an *allocation* action or a *resell* action for each task. The auctioneer then sends reward messages with winning bids for the allocated tasks. The winners keep traces of their reward and commit the local plan associated to the winning bid (process ③ in Figure 1). Finally, the auctioneer integrates the plans attached to the winning bids into $\mathcal{H}_\delta$[1], and a new round is started with the tasks to resell.

Auctioning with partially-ordered HTNs

The allocation protocol presented earlier is quite simple: a consistent HTN tree is updated all along the rounds with the plans corresponding to the allocated tasks, and is extended respectively by the bidders to integrate their own decomposition when estimating bids, and by the auctioneer to integrate the bids and solve the WDP. While the formulation requires

[1] this step, which also removes the deprecated decomposition from $\mathcal{H}$, is quite straightforward, and is hence not further detailed.

some modelling tricks, the process is sound and simple when tasks are totally ordered.

In this section, we present through an example why partially-ordered tasks cannot be managed by the previous approach, and we give some perspectives towards integrating such constraints.

Illustrative example

Again, we use the *BorderDelivery* problem as a support example. Without losing relevancy, we remove ordering contraints $l_1 \rightarrow l_2$ and $l_4 \rightarrow l_5$ because we do not need to bring all packages before checking one and we do not need to bring a precise package before the other, therefore the problem is now partially-ordered.

Let us consider an auctioneer that tries to allocate the $\mathcal{H}$ of Figure 2 in this partially-ordered case. The auctioneer sends the corresponding item to two robots, r_1 and r_2.

Let us also consider that at the first round, r_1 won task l_4 and r_2 won task l_5. The plans of r_1 and r_2 are then respectively (if we only look at primitive tasks in the multi-robot problem) $[\boldsymbol{l_4}]$ and $[\boldsymbol{l_5}]$.

At the second round, the auctioneer sends an updated $\mathcal{H}_2$, where method $(l_1, m\text{-}store\text{-}pkg\text{-}all)$ has been removed, l_4 and l_5 are already allocated, and l_2, l_6 and l_7 are sellable tasks. At this second round, r_1 wins l_6, for which it has computed the plan:

$$[l_5 \rightarrow \boldsymbol{l_6} \rightarrow \boldsymbol{l_4}] \quad (1)$$

where tasks in bold correspond to tasks executed by r_1 (either already allocated or being bid) and non-bold tasks represents tasks allocated to another robot (or not allocated tasks) that are necessary to the robot's plan, i.e. tasks whose effects are preconditions of robot's tasks.

At the third round, only l_7 is to sell. $\mathcal{H}_3$ integrates that both l_4 and l_6 have been allocated to r_1, and l_5 to r_2. Let us consider that r_2 wins l_7, with the plan:

$$[l_4 \rightarrow \boldsymbol{l_7} \rightarrow \boldsymbol{l_5}] \quad (2)$$

Therefore, at the end of the third round, all tasks have been allocated and the auction is finished. However, the resulting allocation cannot be executed. Both robots locally computed a plan involving a temporal constraint between one of their task and a task of the other robot. These constraints are specifically induced by a causal constraints on the *at* predicate, representing the location of the packages. We can indeed notice that the multi-robot plan issued from equations (1) and (2) contains a mutual dependency between l_4 and l_5.

Such a situation could not arise in totally-ordered problems: ordering constraints are already modeled in $\mathcal{H}$, and the plans computed by the robot cannot add new ordering constraints.

Conversely, in partially-ordered problems, each bidder may decide on its own to fix an order between tasks to solve the bid problem, but this new constraints is not forwarded to the auctioneer, and then to other bidders, then leading to possible inconsistencies in the ordering constraints of each local plan.

Problem statement

As we have seen in the previous example, inconsistencies may arise due to the lack of information on the bidders' local plans when solving the WDP and estimating successive bids. A worse situation may even happen: as a bidder decomposes primitive multi-robot tasks with its local domain, the resulting local plan can *interleave* these tasks, while they are primitive for the auctioneer, i.e. not decomposable.

We then have to not only integrate information about ordering constraints coming from bidders local plans, but also make the approach able to account for the interleaving of tasks, at any level of the hierarchy (as bidders can bid on abstract tasks), integrating causal relations with tasks allocated to other robots.

Solution perspectives

The current approach for totally-ordered problems has already some nice features and assumptions that will help integrating all the considered constraints. First, the constraint to allocate at most one task per robot per round eases the integration, as we will not have to consider simultaneous constraints coming from several bids of the same robot at the same time. The update of $\mathcal{H}$ to integrate allocated tasks will also ensure that constraints from previous allocations are enforced in the new bids. Finally, the proposed decomposition of primitive tasks with a $start$ and end action will help formulating constraints allowing the interleaving of tasks.

Therefore, to keep a sound problem, results (i.e. committed local plans) from the allocation of the previous round must be encoded into the new $\mathcal{H}$. By doing so, they will be taken into account during the bid estimation phase without need of further modification. Moreover, this encoding directly comes out from the WDP solution which must not deny the intended plans of the winners. Thus, the most challenging part is the WDP problem formulation (process ②) which must reflect the bidders' intentions in order to provide a consistent solution.

Consequently, we will integrate every local plan computed during the bid estimation into the WDP problem. More specifically, we will investigate whether integrating this local plan can be done when decomposing the task on which the bid is evaluated, in place of a simple action, as illustrated in Figure 4.

Discussion

In this paper, we proposed a preliminary approach to handle the Multi-Robot Task Allocation problem by solving hierarchical auctions with HTN planning. The approach relies on hierarchically linked tasks and auctions in order to interleave decomposition and allocation. Items for sale are HTNs and HTN planning is used to both formulate the bids and solve the Winner Determination Problem.

We presented a protocol that is sound for totally-ordered problems. However, partially-ordered problems raise some specific challenges, and we outlined some perspectives to handle them and support causal and temporal constraints.

Current work aim at improving the approach with the proposed perspectives. Finally, we want to demonstrate the applicability of the approach to online reparation problems.

References

Behnke, G.; Höller, D.; Bercher, P.; Biundo, S.; Pellier, D.; Fiorino, H.; and Alford, R. 2019. Hierarchical Planning in the IPC. In *Workshop on HTN Planning (ICAPS)*.

Bercher, P.; Alford, R.; and Höller, D. 2019. A Survey on Hierarchical Planning - One Abstract Idea, Many Concrete Realizations. In *Int. Joint Conf. on Artificial Intelligence (IJCAI)*. Macao, China.

Bercher, P.; Behnke, G.; Höller, D.; and Biundo, S. 2017. An Admissible HTN Planning Heuristic. In *IJCAI*.

Botelho, S.; and Alami, R. 1999. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Int. Conf. on Robotics and Automation (ICRA)*. Detroit, MI, USA.

Dias, M.; Zlot, R.; Kalra, N.; and Stentz, A. 2006. Market-Based Multirobot Coordination: A Survey and Analysis. *Proceedings of the IEEE* 94(7).

Dias, M. B.; Ghanem, B.; and Stentz, A. 2005. Improving cost estimation in market-based coordination of a distributed sensing task. In *Int. Conf. on Intelligent Robots and Systems (IROS)*. Edmonton, Canada.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *AAAI Conference on Artificial Intelligence (AAAI)*. Seattle, WA, USA.

Gerkey, B.; and Matarić, M. 2004. A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems. *The International Journal of Robotics Research* 23(9).

Höller, D.; Behnke, G.; Bercher, P.; Biundo, S.; Fiorino, H.; Pellier, D.; and Alford, R. 2020. HDDL: An Extension to PDDL for Expressing Hierarchical Planning Problems. In *AAAI Conference on Artificial Intelligence (AAAI)*. New York City, NY, USA.

Kalra, N.; Ferguson, D.; and Stentz, A. 2005. Hoplites: A Market-Based Framework for Planned Tight Coordination in Multirobot Teams. In *Int. Conf. on Robotics and Automation (ICRA)*. Barcelona, Spain.

Khamis, A. M.; Elmogy, A. M.; and Karray, F. O. 2011. Complex Task Allocation in Mobile Surveillance Systems. *International Journal on Intelligent and Robotic Systems* 64(1).

Koenig, S.; Keskinocak, P.; and Tovey, C. 2010. Progress on Agent Coordination with Cooperative Auctions. In *AAAI Conference on Artificial Intelligence (AAAI)*. Atlanta, GA, USA.

Liu, Y.; Yang, J.; Zheng, Y.; Wu, Z.; and Yao, M. 2013. Multi-Robot Coordination in Complex Environment with Task and Communication Constraints. *International Journal of Advanced Robotic Systems* 10(5).

Milot, A.; Chauveau, E.; Lacroix, S.; and Lesire, C. 2021. Market-based Multi-robot coordination with HTN planning. In *Int. Conf. on Intelligent Robots and Systems (IROS)*.

Nunes, E.; and Gini, M. 2015. Multi-robot auctions for allocation of tasks with temporal constraints. In *AAAI Conference on Artificial Intelligence (AAAI)*. Austin, TX, USA.

Otte, M.; Kuhlman, M. J.; and Sofge, D. 2019. Auctions for multi-robot task allocation in communication limited environments. *Autonomous Robots* 44.

Zlot, R.; and Stentz, A. 2006. Market-based Multirobot Coordination for Complex Tasks. *The International Journal on Robotics Research* 25(1).