Solving Hierarchical Auctions with HTN Planning

Antoine Milot[1,2,3], Estelle Chauveau[2], Simon Lacroix[1] and Charles Lesire[3]

[1]LAAS-CNRS, [2]NAVAL GROUP, [3]ONERA

Problematic and approach

Multi-Robot Task Allocation and Planning

When a robot has to complete several tasks it needs to solve a **planning** problem. However, when considering a **multi-robot system** it is necessary to both **allocate** and **plan** these tasks. This results in the **Multi-Robot Task Allocation problem**. In order to solve this problem **we propose an approach interleaving decomposition and allocation of the tasks**.

How to interleave Decomposition and Allocation?

→ by hybridizing **auctions** & **Hierarchical Task Networks**.

Auction-Based Allocation
- Decentralized decision scheme
- Robust to communication failures

Hierarchical Task Network (HTN) Planning
- Integrate expert knowledge to decompose difficult tasks into simpler ones

Allocation with auctions

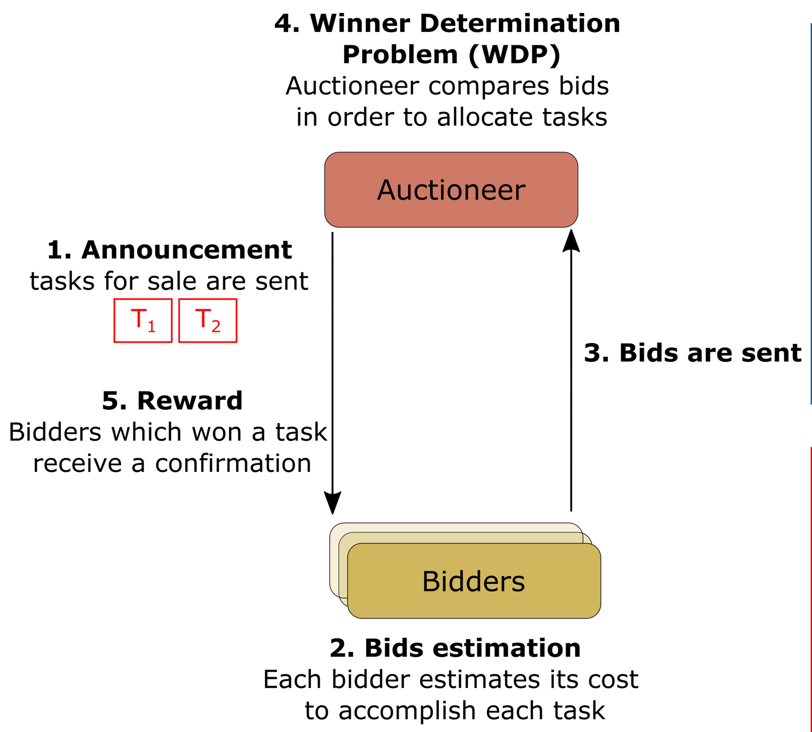


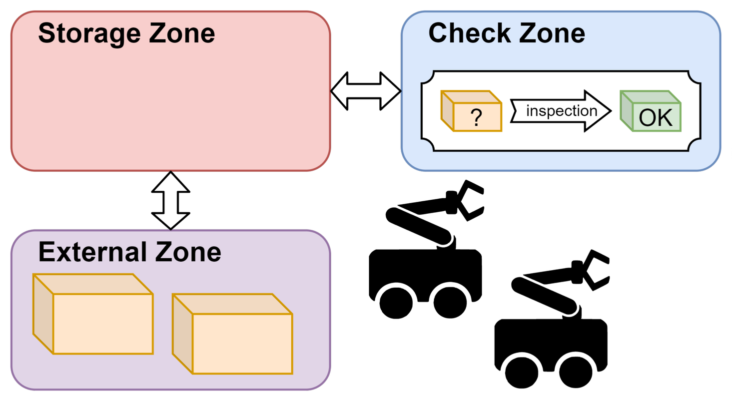
Figure 1: Allocation with auctions process.

Auction key points
- The auctioneer is in charge of tasks allocation
- Bidders formulate bids (ex: a cost)
- Bids are **independant**
- A bidder can win **at most one** bid

Auctioning with HTN planning
- Items for sale are HTNs
- HTN Planning is used to:
  - Estimate bids
  - Solve the WDP

Use case – *BorderDelivery* problem

To illustrate our approach, we consider a *BorderDelivery* problem, inspired from the *transport* and *logistics* problems of the IPC2020.



Objectives:
- **Store** both packages in the *Storage* area
- Randomly **check** one of the packages

Figure 2: BorderDelivery Problem.

This minimal example can be expressed as an HTN planning problem. It allows to highlight the key points of our approach.

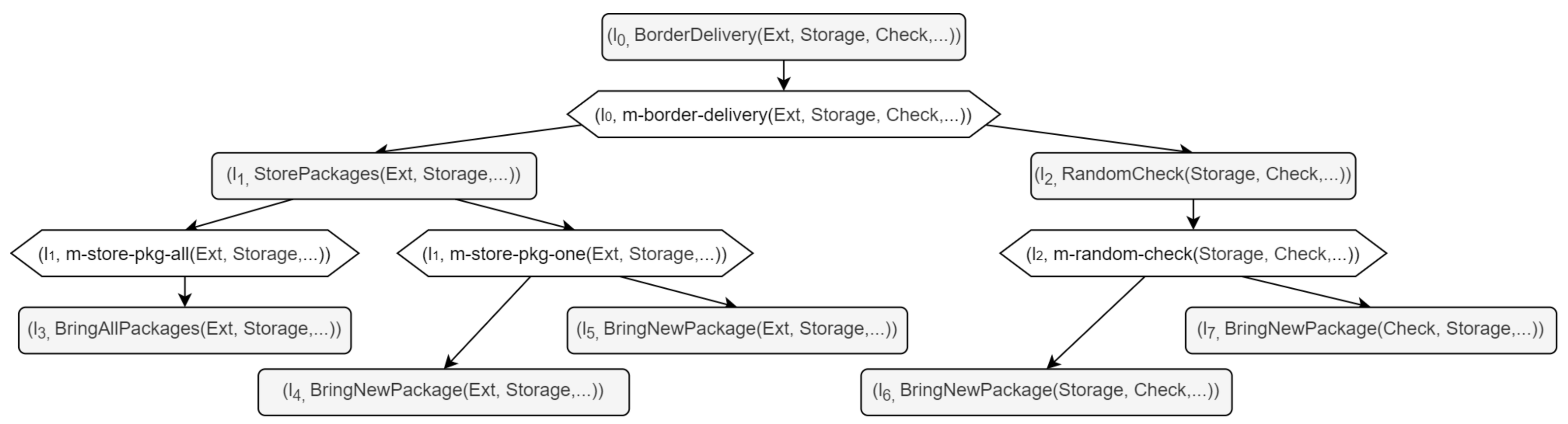


Figure 3: The *Grounded HTN Tree* $\mathcal{H}$ corresponding to the *BorderDelivery* problem.

Overall protocol

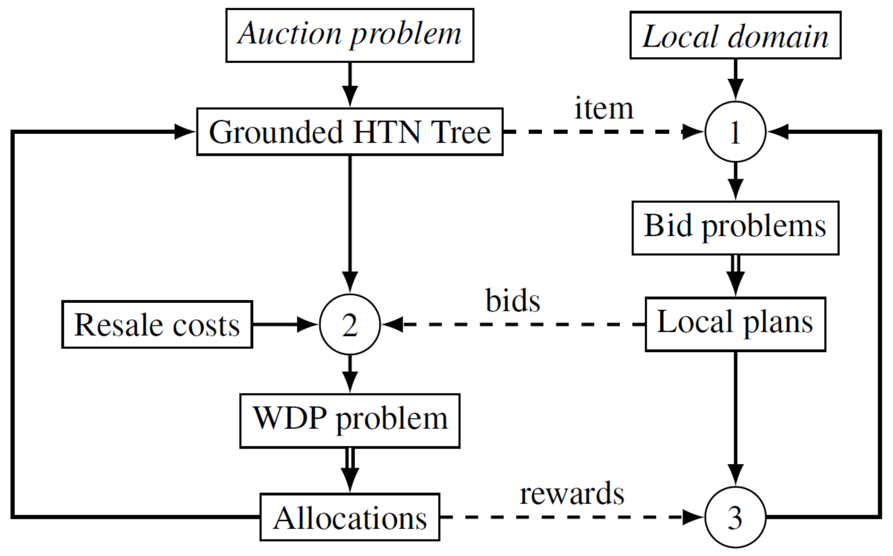


Figure 4: Approach protocol.

The approach features global and local aspects
- Global part: **multi-robot level**, i.e. *what* to allocate and to *who*
- Local part: *how* a **particular robot** can accomplish a task

The tasks in the multi-robot level are **labeled** in order to ensure that each robot reasons on the same elements. The resulting HTN is the *Grounded HTN Tree* $\mathcal{H}$.

1 Bid Estimation
- The bidder **extends**, on the tasks to estimate, $\mathcal{H}$ with its proper actions
- Bidder's capacities are described in a **local HTN domain**
- Only **local actions** can increment the **cost**
- **start** and **end** actions are used to apply **preconditions** and **effects** of the task
- An **HTN solver** is used to find a plan (and its associated cost)

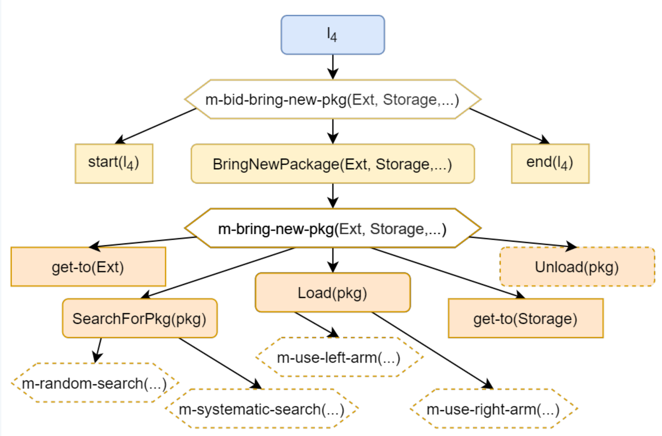


Figure 5: Local decomposition of labeled task l_4.

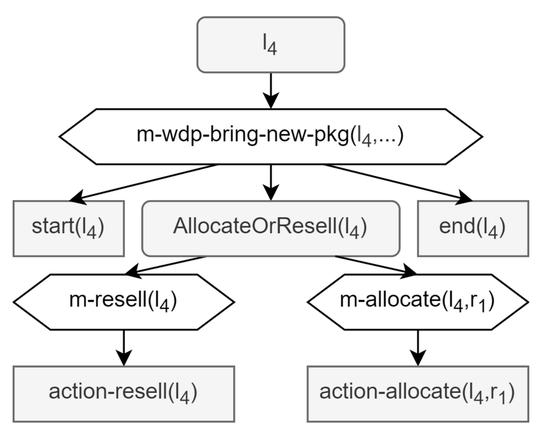


Figure 6: Decomposition of task l_4 integrating received bids.

2 Winner Determination Problem
- For every bid a new method to allocate the task is added
- For every task a new method to resell it is added
- Resale cost is a lever to **control the allocation process**
- An **HTN solver** is used to determine an allocation

3 End of an auction round
- $\mathcal{H}$ is **updated** with the WDP result
- The winners **commit the plans** corresponding to their winning bids
- A **new auction round** begins with the remaining tasks for sale

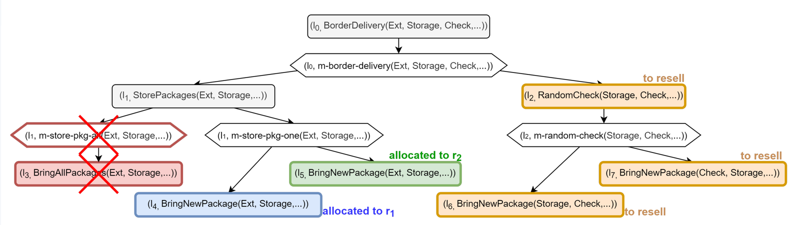


Figure 7: Update example of $\mathcal{H}$ after an auction round.

Totally vs Partially ordered problems
- The approach is sound and complete on **totally ordered problems**
- Improvement are necessary to support **partially ordered problems** with causal constraints

Auctioning with partially-ordered HTNs

Generalities
- **Problems are rarely totally ordered**
- When integrating **causal constraints** they become harder to solve

Numerous applications require to be able to solve partially-ordered or not-ordered problems with causal constraints.

BorderDelivery case
- The presence of a package *at* a specific location is a causal constraint
- There is no particular reason to:
  - move a specific package *first* and *then* the other one
  - wait that both packages are in *Storare* before checking one

By removing the ordering constraints $l_1 \rightarrow l_2$ and $l_4 \rightarrow l_5$ the problem becomes partially-ordered.

Allocation on partially-ordered problems yield execution deadlock

Let's consider two robots, r_1 and r_2, and the following sequence of allocation:
- First round: r_1 wins task l_4 with the plan: $[\boldsymbol{l_4}]$; r_2 wins task l_5 with the plan: $[\boldsymbol{l_5}]$
- Second round: r_1 wins l_6 with the plan: $[l_5 \rightarrow \boldsymbol{l_6} \rightarrow \boldsymbol{l_4}]$
- Third round: r_2 wins l_7 with the plan: $[l_4 \rightarrow \boldsymbol{l_7} \rightarrow \boldsymbol{l_5}]$

All tasks have been allocated. However, the resulting allocation cannot be executed. Both robots **locally** computed a plan involving a **precedence constraint** between one of their task and a task of the other robot. There is a mutual dependency between l_4 and l_5.

Origin of the problem
These inconsistencies may arise due to the **lack of information on the bidders' local plans** when solving the WDP and estimating successive bids.

Necessary improvements
- The results (i.e. committed local plans) from the **allocation of the previous round must be encoded into the new** $\mathcal{H}$.
- The WDP problem formulation (process 2) must **reflect the bidders' intentions** in order to provide a consistent solution.

Ongoing work – Improving the framework
- Transmission of each **local plan associated to a bid** (at multi-robot level)
- **Integration of the received local plans as *possibilities* in the WDP** thanks to new decompositions and causal constraints sets

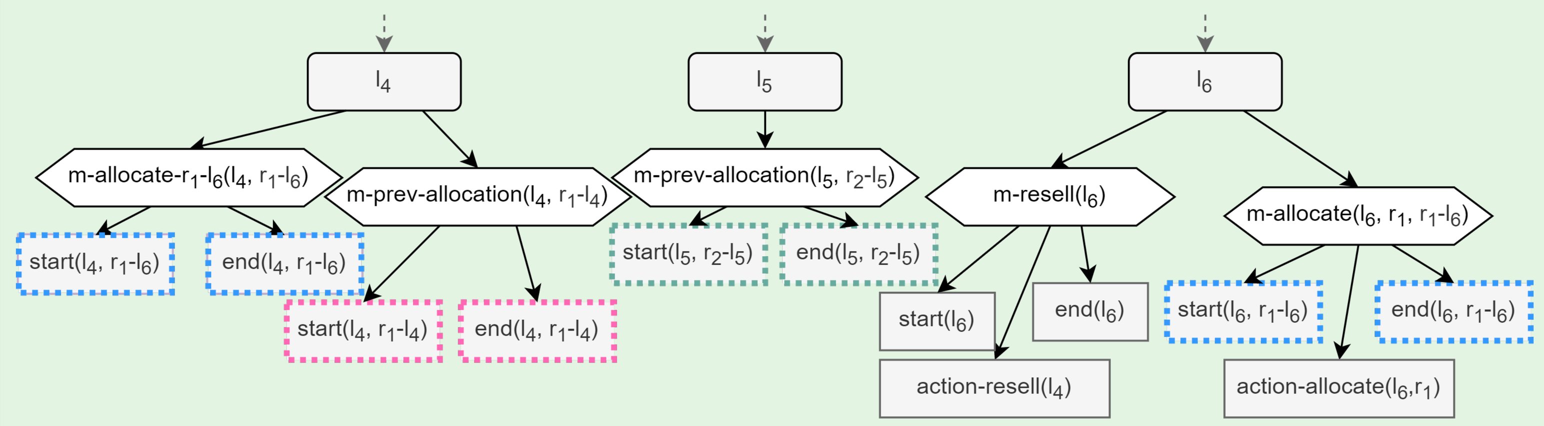


Figure 8: Illustration of the local plans integration in the WDP at the 2nd round.