Trial-Based Heuristic Tree-Search for Distributed Multi-Agent Planning

Tim Schulte  
Institut für Informatik  
Albert-Ludwigs-Universität  
Freiburg, Germany  
schultet@cs.uni-freiburg.de

Bernhard Nebel  
Institut für Informatik  
Albert-Ludwigs-Universität  
Freiburg, Germany  
nebel@cs.uni-freiburg.de

Abstract
We present a novel search scheme for privacy-preserving multi-agent planning, inspired by UCT search. We compare the presented approach to classical multi-agent forward search and evaluate it based on benchmarks from the CoDMAP competition.

Introduction
In collaborative multi-agent planning multiple agents attempt to achieve a common goal by planning and coordinating their actions appropriately. In this paper, we introduce a novel search technique for privacy preserving distributed multi-agent planning based on trial-based heuristic tree-search (THTS) (Keller and Helmert 2013), a general scalable framework for solving different types of planning tasks. Our main contribution is the definition of the resulting search framework, which we call distributed multi-agent trial-based heuristic tree-search (DMT). This framework extends the way of how distributed plans can be generated and so might be useful for portfolio approaches to multi-agent planning. We exemplify two DMT algorithms and provide preliminary empirical evaluation.

Background
A multi-agent planning task consists of a finite set of agents \( n \) for each agent \( i \), for each agent \( i \), each action \( a \) is specified via precondition \( \text{pre}(a) \) and effect \( \text{eff}(a) \), both being partial variable assignments over \( V \). An action \( a \) is applicable in state \( s \) if \( s \) agrees with \( \text{pre}(a) \) wherever \( \text{pre}(a) \) is defined. Application of action \( a \) in state \( s \) yields the successor state \( s' \) which agrees with \( \text{eff}(a) \) where \( \text{eff}(a) \) is defined, and agrees with \( s \), elsewhere. The solution to a planning task is a sequence of actions \( \pi = (a_1, \ldots, a_k) \) such that \( a_1 \) is applicable in \( s_0 \), every subsequent action is applicable in the state generated by its preceding action, and \( a_k(\ldots(a_1(s_0))\ldots) = s_n \).

Privacy constraints are defined in terms of private variables and private actions. The private variables of an agent \( \varphi_i \) can only be observed and be affected by actions of \( \varphi_i \). Private actions of \( \varphi_i \) are only known to \( \varphi_i \) and only depend on or affect private variables of \( \varphi_i \). Public actions can affect or depend on both public and private variables of the agent. Therefore, other agents can only access projections of \( \varphi_i \)’s public actions, where private variables are hidden.

Trial-based Heuristic Tree-Search (Schulte and Keller 2014; Keller and Helmert 2013) algorithms maintain a tree of search nodes and select one of its leaf nodes for expansion in each search step. Three phases are executed repeatedly; their specific behaviour must be defined to derive concrete algorithms. Selection is the first phase of the algorithm with the objective to select one of the leaf nodes for expansion. Beginning from the root, a selection strategy recursively selects a child, until a leaf node is reached. In the initialization phase, the previously selected leaf node is initialized. Successor nodes are generated and integrated into the tree. During the backpropagation phase new information, like value estimates or the number of times a node has been visited during selection, is propagated through the tree. After the backpropagation phase, the algorithm starts again with the first phase. This process is repeated until a goal state is generated, or some limit is reached. By alternatingly executing selection and initialization phase, multiple nodes can be initialized in each search step. The number of nodes that get initialized in a single search step is denoted as trial length.

Distributed Multi-Agent THTS
We now present a complete and privacy preserving scheme for the distributed application of trial-based heuristic tree-search. The concept is similar to MAFS (Nissim and Braffman 2014), where forward-search is concurrently executed while state information is exchanged between the planning agents according to a specific message passing scheme. Each agent performs THTS locally, using its own actions only. Whenever agent \( \varphi_i \) expands a state \( s \) in which a public projection of an action of \( \varphi_j \) is applicable, \( \varphi_i \) will send a message to \( \varphi_j \) containing \( s \). \( \varphi_j \) then integrates \( s \) into its search tree, such that it can prospectively select \( s \) for expansion. To accomplish this, \( \varphi_j \) identifies a suitable parent and adds \( s \) as a child to it. In principle, any node can be used as a parent without soundness or completeness being compromised. However, since the tree structure is crucial to the success of THTS algorithms, it is important where new states are inte-
grated. Let $s$ be the result of applying the sequence of actions $(a_1, \ldots, a_k)$ in the initial state, i.e. $a_k(\ldots(a_1(s_0))\ldots) = s$. If $\phi_j$ has no action in the sequence, it adds $s$ as a child to the root. Otherwise, let $a_j$ be the last action of $\phi_j$ in the sequence. Then, $\phi_j$ adds $s$ as a child to $s' = a_j(\ldots(a_1(s_0))\ldots)$. Note that $\phi_j$ is not aware of all actions in the sequence leading to $s$ and hence cannot compute $s'$. We enable $\phi_j$ to identify $s'$ by using a special message type.

**Definition 1 (State message).** A state message from $\phi_i$ to $\phi_j$ for state $s$ is a tuple $m = (s, h_i, g_i, T)$, where $s$ is a state, $h_i$ is a value estimate of $\phi_i$, for $s$, $g_i$ is the cost of $\phi_i$ to establish state $s$ from the nearest ancestor state contributed by $\phi_j$ (or the root if no such state exists), and $T$ is a set of state tokens. Private components of $s$ are encrypted, such that each agent can only decrypt its own private components.

Each state token belongs to an agent $\phi_k$ and contains a state identification number. This number references a node in the local search space of $\phi_k$ and is meaningless to all other agents. Figure 1 illustrates how tokens are used to integrate states. Numbers next to nodes depict state IDs that correlate with states for which the other agent has an applicable public projection are rendered in bold. When $\phi_2$ initializes the node with state ID 3, it transmits message $m_1$ to $\phi_1$, containing a token that enables $\phi_2$ to identify the node. When $\phi_1$ receives $m_1$, it creates a new search node for $s$. Because $m_1$ contains no token for $\phi_1$, the new node is attached as a child to the root. Later on, $\phi_1$ initializes the node labelled with 5 and sends message $m_2$ to $\phi_2$. Because the state with ID 5 was generated in a branch to which $\phi_2$ contributed an ancestor state, the token $\phi_2 \rightarrow 3$ is attached to the message, along with the new token $\phi_1 \rightarrow 5$. On receiving $m_2$, $\phi_2$ looks up its token $\phi_2 \rightarrow 3$, creates a new node for $s'$, and attaches it as a child to the node with state ID 3.

**Algorithms.** We propose two DMT algorithms, which only differ in the selection strategy used. Both algorithms generate all successors in the initialization phase and propagate the best (minimum) value estimate in the backup phase. DMT-BFS selects the successor node with the best value estimate. DMT-UCB uses a balanced selection strategy based on UCB1 (Kocsis and Szepesvári 2006). Both algorithms were shown to be sound and complete (Schulte and Nebel 2016) when privacy constraints of MA-STRIPS (Brafman and Domshlak 2013) apply.

<table>
<thead>
<tr>
<th>mafs</th>
<th>dmt-bfs$1$</th>
<th>dmt-ucb$1$</th>
<th>dmt-bfs$100$</th>
<th>dmt-ucb$100$</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>69</td>
<td>52</td>
<td>92</td>
<td>84</td>
</tr>
</tbody>
</table>

Table 1: Coverage. DMT algorithms use a trial length according to their indices (1 or 100).

**Evaluation**

DMT and MAFS were both implemented in a distributed multi-agent planning system using the FF heuristic (Hoffmann and Nebel 2001). Experiments were run on a PC with a quad-core CPU and 4 GB of RAM. Table 1 shows coverage results on 240 planning tasks from the CoDMAP competition (Štola, Komenda, and Kovacs 2015) with a time limit of two minutes per task. The results show that both DMT configurations have significantly higher coverage when a trial length of 100 is used and could even outperform the MAFS approach. Increasing the trial length causes additional exploration and encourages faster escape from local minima. A portfolio planner running dmt-ucb$100$, dmt-bfs$100$, and mafs for 2 minutes each solves 117 instances, which shows that MAFS and DMT complement each other well.

**Conclusion**

In this paper we presented DMT, a novel and privacy preserving scheme for distributing THTS algorithms. We proposed two concrete DMT algorithms and evaluated them empirically. Results have shown DMT and MAFS to complement each other in a promising way. In future work we will create and analyze new DMT algorithms to further exploit such complementary strengths.

**References**


