Admissible Landmark Heuristic for Multi-Agent Planning

Michal Štolba and Daniel Fišer and Antonín Komenda
{stolba,fiser,komenda}@agents.fel.cvut.cz
Department of Computer Science, Faculty of Electrical Engineering,
Czech Technical University in Prague, Czech Republic

Abstract

Heuristics are a crucial component in modern planning systems. In optimal multiagent planning the state of the art is to compute the heuristic locally using only information available to a single agent. This approach has a major deficiency as the local shortest path can arbitrarily underestimate the true shortest path cost in the global problem. As a solution, we propose a distributed version of a state-of-the-art LM-Cut heuristic. We show that our distributed algorithm provides estimates provably equal to estimates of the centralized version computed on the global problem. We also evaluate the algorithm experimentally and show that on a number of domains, the distributed algorithm can significantly improve performance of a multiagent planner.

Introduction

One of the most commonly used models for deterministic multiagent planning with cooperative agents is MA-STRIPS (Brafman and Domshlak 2008), which introduces the notion of privacy to the classical STRIPS model. We assume, that while the agents can communicate during the planning process, the private knowledge should not be exchanged.

Since the introduction of the MA-STRIPS formalism for multi-agent planning and the publication of the Multi-Agent Distributed A* (MAD-A*) (Nissim and Brafman 2012), there has not been as much progress in the cost-optimal multiagent planning as in the satisficing variant. Apart from a theoretical work on weighted automata (Fabre and Jezequel 2009), cost-optimal factored planning (Fabre et al. 2010) and A# (Jezequel and Fabre 2012), most published work focused on satisficing multiagent planning.

One of the prerequisites for cost-optimal planning is the existence of an admissible heuristic. In MAD-A*, the search is guided by well-known LM-Cut (Helmert and Domshlak 2009) or Merge&Shrink (Helmert, Haslum, and Hoffmann 2007) heuristics, computed on a factor of the problem visible to a single agent (a projected problem). The projected problem consists of all actions of the agent and publicly known parts of actions of other agents as defined in MA-STRIPS (will be formalized later).

A fundamental flaw of computing the heuristic on a projected problem is that the shortest path in the projected problem may be arbitrarily shorter than is the global one, which may result in an arbitrarily bad heuristic estimate. Consider an example, where each agent can achieve the (global) goal \( g \) by a sequence of \( n \) internal (or private) actions followed by a single public action (Figure 1a).

\[
\begin{align*}
\begin{array}{c}
\text{i} \\
\hline
a_1 \\
\vdots \\
a_n \\
\vdots \\
q_1 \\
\vdots \\
q_{n-1} \\
a'_{n-1} \\
\vdots \\
a'_{1} \\
\hline
\end{array}
\end{align*}
\]

\[
\begin{align*}
\begin{array}{c}
\text{p}_1 \\
\vdots \\
\text{p}_{n-1} \\
\text{p}_n \\
\hline
\end{array}
\end{align*}
\]

Figure 1: The full (a) and projected (b) problem.

In Figure 1a, \( a_1, \ldots, a_n \) and \( a'_1, \ldots, a'_{n} \) are internal to respective agents and \( a_g, a'_g \) are visible to all agents (public). If we consider unit cost actions, the cost of an optimal solution is \( n + 1 \).

The projected problem looks quite different (Figure 1b). The agent knows only about the action \( a'_i \) of the other agent. Moreover it does not know about its precondition and considers \( a'_i \) applicable in the initial state \( i \). In the projected problem, the cost of an optimal solution is 1. Therefore no admissible heuristic computed on the projected problem can give estimate higher than 1. This example can be scaled to any number of agents and enlarged to bound the estimate of the projected heuristic arbitrarily far from the real optimal solution cost.

In this paper we solve this problem by providing distributed versions of the admissible \( h_{max} \) (Bonet and Geffner 1999) and LM-Cut heuristics (based on the \( h_{max} \) heuristic). Similar treatment was applied on heuristics for satisficing planning, e.g., (Štolba and Komenda 2013; 2014; Torreño, Onaindia, and Sapena 2014; Maliah, Shani, and Stern 2014). We first describe the algorithms for a distributed \( h_{max} \) and LM-Cut and show that they give the same estimates as the centralized versions (which also implies admissibility). We conclude with experimental evaluation of the distributed LM-Cut heuristic.
Multi-Agent Planning

Now, we formally define the MA-STRIPS problem and several other notions necessary for the rest of the paper.

**Definition 1.** A MA-STRIPS problem for a system of $N = \{1, \ldots, n\}$ agents $A = \{\alpha_i\}_{i \in N}$ is a quadruple $\Pi = \langle P, \{A_i\}_{i \in N}, I, G \rangle$, where:

- $P$ is a set of atoms (facts), where $s \subseteq P$ is a state
- $I \subseteq P$ is the initial state
- $G \subseteq P$ is the goal condition

For all agents other than $\alpha_i$, that is all $\alpha_j \in A \setminus \{\alpha_i\}$, we will use a abbreviated notation $\alpha_i$ (in general $\alpha_i \neq j$ will be a shorthand notation for $j \in N \setminus \{i\}$).

For an agent $\alpha_i$, we define the following:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_i$</td>
<td>union of facts $\cup_{a \in A_i} \text{facts}(a)$ set of facts</td>
</tr>
<tr>
<td>$P_{i}^{\text{priv}}$</td>
<td>union of facts $\cup_{j \neq i} P_j$ set of private facts</td>
</tr>
<tr>
<td>$P_{i}^{\text{pub}}$</td>
<td>union of facts $\cup_{j \neq i} P_j \setminus P_i$ set of public facts</td>
</tr>
<tr>
<td>$A_{i}^{\text{pub}}$</td>
<td>set of pub. actions $A_i \setminus P_{i}^{\text{priv}}$ set of priv. actions</td>
</tr>
<tr>
<td>$A_{i}^{\text{priv}}$</td>
<td>set of priv. actions $A_i \cup A_{i}^{\text{pub}}$ set of pub. actions</td>
</tr>
<tr>
<td>$s^{\alpha_i}$</td>
<td>$s \cap P_i$ $\alpha_i$-projection of state $s$</td>
</tr>
<tr>
<td>$a^{\alpha_i}$</td>
<td>$\alpha_i$-projection of action $a \in A$</td>
</tr>
<tr>
<td>$a_{\text{LO}}^{\alpha_i}$</td>
<td>$\alpha_i$-local action corresponding to $a \in A_i$</td>
</tr>
<tr>
<td>$P^{\text{all}}_{i}$</td>
<td>$P_i \cup \bigcup_{j \neq i} {p</td>
</tr>
<tr>
<td>$A_{i}^{\text{all}}$</td>
<td>$A_i \cup \bigcup_{j \neq i} {a^{\alpha_j}</td>
</tr>
</tbody>
</table>

Additionally, we assume, that all goals are public, that is $G \subseteq \bigcup_{i \in N} P_{i}^{\text{pub}}$.

STRIPS problems derived from the MA-STRIPS problem $\Pi$ are a global problem $\Pi^G = \langle P, \bigcup_{i \in N} A_i, I, G \rangle$, an $\alpha_i$-projected problem $\Pi^{\alpha_i} = \langle P_{i}^{\text{priv}}, A_{i}^{\text{all}}, I \cap P_{i}^{\text{pub}}, G \rangle$ and an $\alpha_i$-local problem $\Pi^{\text{LO}}_{\alpha_i} = \langle P_{i}^{\text{priv}}, A_{i}^{\text{pub}}, I \cap P_{i}^{\text{pub}}, G \rangle$. All the defined problems can be extended with a cost function $\epsilon : A \rightarrow \mathbb{N}^{\geq 0}$ (restricted to the used subset of actions). The costs of an action and its projection are equal.

An $\alpha_i$-projected heuristic is a heuristic computed on the $\alpha_i$-projected problem $\Pi^{\alpha_i}$. Any admissible heuristic computed on an $\alpha_i$-projected problem $\Pi^{\alpha_i}$ is admissible also for the global problem $\Pi^G$.

For a proof sketch observe that a path in $\Pi^{\alpha_i}$ can be constructed from a path in $\Pi^G$ by removing private actions and replacing public actions with respective projections. The constructed path has lower or equal cost (actions were only removed). For contradiction, let us assume there is a state $s$ for which the $\alpha_i$-projected heuristic gives higher estimate than the global heuristic. The least-cost path $\pi^{\alpha_i}$ from $s$ to goal in $\Pi^{\alpha_i}$ has to have higher cost than the least-cost path $\pi$ from $s$ to goal in $\Pi^G$ (follows from admissibility of both heuristics in their respective problems). But that is not possible, because the cost of path $\pi^{\alpha_i}$ in $\Pi^{\alpha_i}$ constructed from $\pi$ is less or equal to the cost of $\pi$. The least-cost path in $\Pi^{\alpha_i}$ cannot have higher cost than $\pi^{\alpha_i}$.

**Distributed $h_{\text{max}}$ Heuristic**

Both $h_{\text{max}}$ and LM-Cut heuristics fall into the class of relaxation heuristics. Relaxation heuristics base the estimate on a solution of a relaxed problem $\Pi^+$, in which all delete effects are ignored (for all actions, $a^+ = \langle \text{pre}(a), \text{add}(a), \emptyset \rangle$). Although the cost of optimal solution to $\Pi^+$ is an admissible estimate, it is NP-hard to compute. Thus various approximations are used, one of the best is the LM-Cut heuristic which computes the estimate by iteratively searching for landmark actions and updating the cost function. To do so the $h_{\text{max}}$ heuristic is computed in each iteration. In the following, we introduce the $h_{\text{max}}$ heuristic and provide a method for its distributed computation. A distributed $h_{\text{max}}$ was already described in (Štola and Komenda 2014) but without provable equality with the centralized version, as is required here.

Let $O(p)$ be a set of actions achieving $p$ in $\Pi^G$, formally $O(p) = \{a \in A|p \in \text{add}(a)\}$. Similarly $O^{\alpha_i}(p) = \{a \in A_{i}^{\text{all}}|p \in \text{add}(a)\}$ in $\Pi^{\alpha_i}$ and $O^{\text{LO}}(p) = \{a \in A_{i}^{\text{pub}}|p \in \text{add}(a)\}$ in $\Pi^{\text{LO}}_{\alpha_i}$.

The $h_{\text{max}}$ heuristic is defined by a set of recursive equations:

$$h_{\text{max}}(P, p) = \max_{p \in P} \{h_{\text{max}}(P, s)\} \quad (1)$$

$$h_{\text{max}}(p, s) = \begin{cases} 0 & p \in s \\ h_{\text{max}}(\arg\min_{a \in O(p)} (h_{\text{max}}(a, s)), s) & \text{ otherwise} \end{cases} \quad (2)$$

$$h_{\text{max}}(a, s) = \text{cost}(a) + h_{\text{max}}(\text{pre}(a), s) \quad (3)$$

where $P$ is a set of facts (goal or action preconditions).

Throughout the text, projected $h_{\text{max}}^{\alpha_j}$ is $h_{\text{max}}$ computed on $\Pi^{\alpha_j}$, local $h_{\text{max}}^{\alpha_j}$ is $h_{\text{max}}$ computed on $\Pi^{\alpha_j}$ and distributed $h_{\text{max}}^{\text{LO}}$ is $h_{\text{max}}$ computed on $\Pi$, but estimating $\Pi^G$.

Initiator agent is the agent $\alpha_i$ which starts the computation of distributed heuristic (e.g., $h_{\text{max}}^{\alpha_i}$) for a given state $s$. All other agents $\alpha_j \neq i$ participating on the computation will be called participant agents.
Distributed \( h_{\text{max}} \) Algorithm

The distributed \( h_{\text{max}} \) algorithm is shown in Algorithm 5. The computation is initiated by agent \( \alpha_i \). The values of \( h_{\text{max}} \) are first initialized to the values of \( \alpha_i \)-projected \( h_{\text{max}} \) (line 2). The algorithm then steps into a loop. In each iteration, the initiator sends requests to all other agents \( \alpha_j \neq \alpha_i \), containing current heuristic values for \( \alpha_i \)-projections of all \( a \in A^\text{pub}_j \) (and for the facts in \( I \)).

The participant \( \alpha_j \) receives the request and computes heuristic values for all \( a \in A^\text{pub}_j \) from the received values. Afterwards, \( \alpha_j \) sends reply to \( \alpha_i \) containing \( h_{\text{max}}(a^{\alpha_i}, s) \) for all \( a \in A^\text{pub}_j \) (in fact only values which changed must be sent).

When received (line 6), the \( h_{\text{max}} \) values are updated based on the received reply (line 7). If no actions are updated, the loop exists, the algorithm terminates and returns heuristic value for the goal.

Algorithm 1: Distributed \( h_{\text{max}} \) heuristic

1. Algorithm \( \text{computeDistHmax} \) \( (\alpha_i, s, G) \)
2. initialize \( h_{\text{max}} \) to \( h_{\text{max}}(a^{\alpha_i}, s) \) for all \( a \in A^\text{all} \)
3. while \( h_{\text{max}}(a^{\alpha_i}, s) \) changed for some \( a \in A^\text{all} \)
4. for each agent \( \alpha_j \) \( \in A \setminus \{\alpha_i\} \)
5. send \( h_{\text{max}}(a^{\alpha_i}, s) \) for all \( a \in A^\text{pub}_j \) to \( \alpha_j \)
6. receive \( h_{\text{max}}(a^{\alpha_i}, s) \) for all \( a \in A^\text{pub}_j \)
7. update \( h_{\text{max}}(a^{\alpha_i}, s) \)
8. return \( h_{\text{max}}(G, s) \)

Equality of \( h_{\text{max}} \) and \( h_{\text{max}} \)

In this section we show that the distributed \( h_{\text{max}} \) algorithm returns the same value as the centralized \( h_{\text{max}} \) for any fact or action in any state. In summary, the proof proceeds as follows. First, a relation between the \( \alpha_i \)-projected and distributed \( h_{\text{max}} \) is established, stating that \( h_{\text{max}}(a^{\alpha_i}, s) \leq h_{\text{max}}(a, s) \) for all \( a \in A^\text{all} \) (see Lemma 2). This is necessary because in the algorithm, all facts are initialized to the \( \alpha_i \)-projected \( h_{\text{max}} \) values and iteratively refined until the global \( h_{\text{max}} \) values are reached.

Computing the \( h_{\text{max}} \) updates based on the public action and the initial state is shown to be sufficient (see Lemma 3). As a consequence, each \( a \in A^\text{pub}_i \) such that \( h_{\text{max}}(a^{\alpha_i}, s) < h_{\text{max}}(a, s) \) is updated to \( h_{\text{max}}(a^{\alpha_i}, s) = h_{\text{max}}(a, s) \) after finitely many steps (see Lemma 4).

From this fact and from the relation of \( \alpha_i \)-projected and distributed \( h_{\text{max}} \) follows that eventually all facts are updated to the desired value of centralized \( h_{\text{max}} \) (see Theorems 5 and 6).

**Lemma 2.** For each state \( s \) and fact \( a \in A^\alpha_i \), \( h_{\text{max}}(a^{\alpha_i}, s) \leq h_{\text{max}}(a, s) \).

The proof of the lemma follows from the observations that \( O^\alpha_i(\cdot) \) and \( O(\cdot) \) contains the same actions (or their projections) for any \( p \in P^\text{all}_i \) and that a projected action \( a^{\alpha_i} \) has lower or equal number of preconditions compared to the action \( a \). Therefore \( h_{\text{max}}(a^{\alpha_i}, s, a) \leq h_{\text{max}}(a, s, a) \).

To correctly update a fact or action value in the \( \alpha_j \)-local \( h_{\text{max}} \), it is enough to provide correct \( h_{\text{max}} \) values for all preceding public actions of agent \( \alpha_j \). An action \( a \) is preceded by action \( a' \) if \( a' \cap \text{pre}(a) \neq 0 \) or \( a' \) exists such that \( a \) is preceded by \( a'' \) and \( a'' \) is preceded by \( a' \). We say that \( a \) is succeeding \( a' \).

**Lemma 3.** If \( h_{\text{max}}(a^{\alpha_i}, s) = h_{\text{max}}(a, s) \) for all \( a \in A^\text{pub}_i \), then \( h_{\text{max}}(a^{\alpha_i}, s) = h_{\text{max}}(a, s) \).

The proof of the lemma is based on the following facts. If \( a^{\alpha_i} \) is preceded only by actions in \( A^\text{priv}_i \) the lemma holds trivially. Similarly if the precondition \( p \) maximizing \( h_{\text{max}} \) is an effect of such \( a^{\alpha_i} \). If the maximizing precondition \( p \) is an effect of some \( a' \in A^\text{pub}_i \) the lemma holds because of its assumption and because \( a' \) is preceding \( a \).

To conclude the proof of equality of \( h_{\text{max}} \) and \( h_{\text{max}} \) we show that each fact with incorrect value is eventually updated and that the algorithm terminates with the desired values for all facts (and all actions).

**Lemma 4.** Each \( p \in \mathcal{P} \) such that \( h_{\text{max}}(p, s) < h(p, s) \) is updated to \( h_{\text{max}}(p, s) = h(p, s) \) after finitely many steps.

**Proof.** Distributed \( h_{\text{max}} \) is initialized to \( h_{\text{max}} \) for all facts and actions. As already shown in the proof of Lemma 2, \( h_{\text{max}}(a, s) \leq h_{\text{max}}(a, s) \) holds for some action \( a \in A_i \), it is caused by some projected action preceding \( a \) and missing the (private) precondition maximizing \( h_{\text{max}} \). Let \( a^{\alpha_i} \) (where \( a_0 \neq A^\text{priv}_i \)) be such a projected action preceding \( a \) for which \( h_{\text{max}}(a^{\alpha_i}, s) \leq h_{\text{max}}(a_0, s) \). Let for all \( p \in \text{pre}(a^{\alpha_i}) \) hold \( h_{\text{max}}(p, s) = h_{\text{max}}(p, s) \), which means that all actions preceding \( a^{\alpha_i} \) already have \( h_{\text{max}} \) equal to \( h_{\text{max}} \). Such action \( a_0 \) always exists, because as \( h_{\text{max}} \) is initialized to \( h_{\text{max}} \) all actions applicable in \( I \) or preceded only by actions in \( A^\text{priv}_i \) have already \( h_{\text{max}} \) equal to \( h_{\text{max}} \).

The inequality \( h_{\text{max}}(a^{\alpha_i}, s) \leq h_{\text{max}}(a_0, s) \) is caused by a fact \( p \in \text{pre}(a_0) \setminus \mathcal{P}_i \) maximizing \( h_{\text{max}}(a_0, s) \). The action \( a_0 \) is sent alongside all other actions in \( A^\text{pub}_i \) to agent \( \alpha_j \) in order to obtain an update. Agent \( \alpha_j \) computes the updated heuristic for all actions from the local problem \( \Pi^{\alpha_j} \), and sends the information back.

From Lemma 3 and because we assumed that, for all actions \( a \in A^\text{pub}_i \), the equality \( h_{\text{max}}(a^{\alpha_i}, s) = h_{\text{max}}(a_0, s) \) holds, it holds also for the returned value of \( a_0 \). Subsequently, \( h_{\text{max}} \) is updated so that \( h_{\text{max}}(a^{\alpha_i}, s) = h_{\text{max}}(a_0, s) \). In the next iteration, for some other action \( a_j \) preceding \( a \) holds \( h_{\text{max}}(a^{\alpha_i}, s) \leq h_{\text{max}}(a_j, s) \) while for all actions preceding \( a_j \) the equality holds. The same reasoning can be applied to \( a_j \). Because there is only a finite number of actions and in each iteration one of the actions is updated to the correct value, action \( a \) is also updated after a finitely many steps.

**Theorem 5.** Algorithm 1 terminates with \( h_{\text{max}}(p, s) = h_{\text{max}}(p, s) \) for any given state \( s \) and all \( p \in \mathcal{P} \).

**Proof.** For each \( a \in A^\alpha_i \), when \( h_{\text{max}}(p, s) = h_{\text{max}}(p, s) \), the heuristic value for fact \( p \) is never changed again. Due
to the finite number of facts in a problem and Lemma 4, all facts are updated to $h_{max}^{DM}(p, s) = h(p, s)$ after a finite number of iterations. After that, no fact and therefore no action is updated and the algorithm terminates.

In the next sections, the heuristic values computed by participant agents $\alpha_{j}\neq i$, that is the $h_{max}^{L_{i,j}}(p, s)$ for the given state $s$ and all facts $p \in P_{1}^{\pr}$, will be preserved throughout the computation. For $h_{max}^{L_{i}}$, the equality holds as well.

**Theorem 6.** Algorithm 1 terminates with $h_{max}^{Lo_{j}}(p, s) = h_{max}(p, s)$ for all $j \neq i, p \in P_{1}$ and any given state $s$.

**Proof.** From Theorem 5 the equality of $h_{max}^{DM}$ and $h_{max}$ holds for all fact and thus for all actions. It holds also for all projections of public actions $a \in A_{pub}^{i}$. From Lemma 3, $h_{max}^{Lo_{j}}(a, s) = h_{max}(a, s) \forall a \in A_{j}$ (and for all $p \in P_{1}$).

**Distributed Landmark Heuristic**

The LM-Cut heuristic (will be denoted as $h_{lm}^{i}$) provides an admissible estimate of the optimal path for a relaxed problem $\Pi^{+}$ by utilizing the idea of disjunctive landmarks. Here, we present a distributed version $h_{lm}^{D}$ for which we show $h_{lm}^{D}(s) = h_{lm}^{i}(s)$ for any state $s$.

**The LM-Cut Heuristic**

**Definition 7. Disjunctive landmark** (or landmark) is a set of actions $L \subseteq A$ s.t. each plan must contain at least one $a \in L$. Cost of landmark $L$ is $c^{m}(L) = \min_{a \in L}c(a)$, where $c(a)$ is the cost of action $a$.

The $h_{lm}^{i}$ heuristic is obtained from a sequence $(L_{k}, c_{k})_{k=0}^{m}$ of landmarks and cost functions, $h_{lm}^{i} = c_{0}^{m}(L_{0}) + c_{1}^{m}(L_{1}) + ... + c_{m}^{m}(L_{m})$. Initially, $c_{0} = c$ and in each iteration $k$, landmark $L_{k}$ is computed using $c_{k}$ and a new cost function $c_{k+1}$ is determined (Algorithm 2, Step 4).

We assume that there is a single fact $i$ representing the initial state and a single fact $g$ representing the goal. If it is not the case, the problem can be transformed by adding zero-cost action $a_{i}$ for each $p \in I$ s.t. $pre(a_{i}) = \{i\}$ and $add(a_{i}) = \{p\}$. The goal $G$ can be treated analogously. Moreover, we assume that each action has at least one precondition and one effect, again, general problem can be transformed by setting $i$ as precondition of actions for which $pre(a) = \emptyset$ and adding a dummy effect $\bot$ for actions for which $add(a) = \emptyset$.

The algorithm will be illustrated on a STRIPS example with a set of 5 actions $\{a_{1}, ..., a_{5}\}$ (later, in a MA-STRIPS formulation, the actions will be divided among two agents $a_{1}, a_{2}$):

$A_{1}: \quad a_{1}: i \rightarrow p_{1}, p_{2} \quad a_{2}: p_{1}, p_{4} \rightarrow g$

$A_{2}: \quad a_{3}: i \rightarrow p_{3}, p_{4} \quad a_{4}: p_{3} \rightarrow p_{5}$

The facts are $\{i, g, p_{1}, ..., p_{5}\}$, where $i$ is the initial fact and $g$ is the goal fact. The initial cost function $c_{0} = c$ is defined as $c(a_{1}) = 3, c(a_{2}) = 1, c(a_{3}) = 1, c(a_{4}) = 1$ and $c(a_{5}) = 1$.

In **Step 1** the $h_{max}$ heuristic is computed for all facts based on $c_{0}$, that is $h_{max}^{0} = \{p_{1} : 3, p_{2} : 5, p_{3} : 1, p_{4} : 1, p_{5} : 2, g : 3\}$.

In **Step 2** a justification graph $J_{0}$ is constructed. A **Justification graph** $J$ is a directed graph with a vertex for each $p \in P$ and an edge $(p, q)$ labeled $a$ if there exists an action $a$ s.t. $p = pcf(a)$ and $q \in add(a)$. Function $pcf$ (a pre-condition choice function) assigns to a given action $a$ one of its preconditions. In $h_{lm}^{i}$, the $pcf$ assigns a precondition maximizing $h_{max}$, ties broken arbitrarily. In the example, $pcf = \{a_{1} \rightarrow i, a_{2} \rightarrow p_{1}, a_{3} \rightarrow i, a_{4} \rightarrow p_{3}, a_{5} \rightarrow p_{2}\}$, resulting in $J_{0}$:

```
 i   p1   a1   p2   a2   g
 a1   p1   p2   a2   p2   g
 a3   p3   a4   p4   a5   p5
 a3   p3   a4   p4   a5   p5
```

In **Step 3** the landmark $L_{k}$ is constructed. (a) All facts $p$ from which the goal $g$ is reachable through a path on which each edge has a label $a$ s.t. $c_{k}(a) = 0$ (is 0-reachable from $p$) are added to $V_{k}^{*}$. In the iteration $k = 0$ of the example it is $V_{0}^{*} = \{g\}$. (b) Find all fact reachable from $i$ without visiting any fact from $V_{k}^{*}$. In the example it is all facts except for $g$. (c) Find all fact reachable from $i$ without visiting any fact from $V_{k}^{*}$. In the example, it is all facts except for $g$. If an edge crossing to $V_{k}^{*}$ (that is $e = (p, q)$ and $q \in V_{k}^{*}$) is reached, label of the edge is added to $L_{k}$. In the example this includes all edges leading to $g$, resulting in $L_{0} = \{a_{2}, a_{3}\}$.

In **Step 4** new cost function $c_{k+1}$ is defined. The costs of all actions in $L_{k}$ is reduced by the cost of $L_{k}$, that is the cost of the least-cost action in $L_{k}$. In the example, $c_{1}(a_{2}) = 0$ and $c_{2}(a_{3}) = 0$, for all other actions it is the same as $c_{0}$.

The computation continues by Step 1. of iteration $k + 1$, until $h_{max}(g) = 0$.

**Algorithm 2: LM-Cut Heuristic**

1. Compute $h_{max}^{k}$ based on $c_{k}$ for every $p \in P$. If $h_{max}^{k}(g) = 0$ terminate and return $h_{lm}^{i}$.
2. Construct a justification graph $J_{k}$
3. Construct a disjunctive landmark $L_{k}$
   (a) Find all facts $p$ s.t. $g$ is 0-reachable from $p$, add $p$ to $V_{k}^{*}$
   (b) Find all facts reachable from $i$ without visiting a fact in $V_{k}^{*}$
   i. If an edge cross to $V_{k}^{*}$, add its label to $L_{k}$

4. Let $c_{k+1}(a) = \{c_{k}(a) - \max_{a \in L_{k}}c_{k}(L_{k}) \quad a \notin L_{k}$
5. Continue with Step 1. for $k = k + 1$

**Distributed LM-Cut Heuristic**

In the following, we assume that the participant agents $\alpha_{j}\neq i$, keep the result of computation (context) of $h_{max}^{DM}$, that is the
heuristic values for all $p \in P_{priv}$ and $a \in A_{\alpha}$. Moreover, we will modify the tie-breaking behavior of the $pcf$ function so that if the tie is between a public and private fact, the public fact will be preferred. We assume that the $pcf$ always chooses the same precondition in both $h_{lmc}$ and $h_{D_{lmc}}$.

To compute the distributed version of the heuristic we introduce a projected version of landmarks:

**Definition 8.** An $\alpha_{ij}$-projected disjunctive landmark (or $\alpha_{ij}$-projected landmark) $L_{\alpha_{ij}}$ corresponding to disjunctive landmark $L$ is $L_{\alpha_{ij}} = (L \cap A_{\alpha_{ij}}) \cup \{\bar{a}\}$, where $\bar{a}$ is a placeholder action.

The placeholder action represents the cost of private actions of other agents in $L$, so when the landmark is completed, $c(\bar{a}) = c^{m}(L)$.

Using the $h_{max}$ values computed by the distributed algorithm for each fact, the justification graph for the global problem can be reconstructed. The resulting **distributed justification graph** $J^{D_{\alpha_{ij}}} = (J^{\alpha_{ij}}, \{J^{\alpha_{ij}}\}_{\neq \psi})$ consists of an $\alpha_{ij}$-projected justification graph $J^{\alpha_{ij}}$ and a set of $\alpha_{ij}$-local justification graphs $\{J^{\alpha_{ij}}\}_{\neq \psi}$.

An $\alpha_{ij}$-**projected justification graph** $J^{\alpha_{ij}}$ is a justification graph over $P \cup \{\bot\}$ with labels from $A_{\alpha_{ij}}$. The $pcf$ is modified so that for each projected action $a_{\alpha_{ij}}$, $pcf(a_{\alpha_{ij}}) = p$ if $p \in P_{\alpha_{ij}}$ maximizes $h_{max}$ and $h_{max}(a_{\alpha_{ij}}, s) = h_{max}(p, s)$, otherwise $pcf(a_{\alpha_{ij}}) = \bot$ if $h_{max}(a_{\alpha_{ij}}, s) > h_{max}(p, s)$. This means that the maximizing fact is private to some other agent. If $add(a_{\alpha_{ij}}) \cap P_{pub} = \emptyset$, we treat the action as if $add(a_{\alpha_{ij}}) = \{\bot\}$. Edges are not connected via $\bot$. An $\alpha_{ij}$-**local justification graph** $J^{\alpha_{ij}}$ is similarly defined over $P_{priv} \cup \{\bot\}$ with labels from $A_{\alpha_{ij}}$.

The distributed justification graph is a distributed graph where the partitions have pairwise disjunctive sets of vertices. Each edge in $J^{\alpha_{ij}}$ with label containing a projected action $a_{\alpha_{ij}}$ s.t. $a \in A_{\neq \psi}$ can be seen as an edge shared with $J^{\alpha_{ij}}$, where the corresponding edge has a label containing the respective $a_{\alpha_{ij}}$.

The distributed version of $h_{lmc}$, denoted as $h^{D_{lmc}}$ follows the same major steps as the centralized version - it differs in that the computation is distributed in some of the steps. To illustrate we will use the previous example in a MA-STRIPS formulation in which $\{p_{2}, p_{4}, g\}$ are public facts and $\{a_{1}, a_{2}, a_{3}, a_{5}\}$ public actions.

In **Step 1** of the $k$-th iteration, distributed version of $h_{max}$ is computed based on the cost function $c_{k}$. The initiator agent $a_{i}$ computes $h^{D_{\alpha_{ij}}}_{max,k}$ for all facts in $P_{i}$ while all other agents $a_{j \neq i}$ compute $h^{L_{\alpha_{ij},k}}$ for all facts in $P_{priv}$. The computed values are identical to the values of centralized $h_{max}$ (Theorems 5 and 6).

In **Step 2** the initiator agent $a_{i}$ builds an $\alpha_{ij}$-projected justification graph $J^{\alpha_{ij}}$ based on the values of $h^{D_{\alpha_{ij}}}_{max,k}$ whereas all other agents build $\alpha_{ij}$-local justification graph $J^{\alpha_{ij}}_{k}$ based on the values of $h^{L_{\alpha_{ij},k}}$, together forming a distributed justification graph $J^{D_{\alpha_{ij}}}$. In our example, the justification graphs are the following:

For $J^{\alpha_{ij}}$:

Notice, that $a_{5}$ has $\bot$ as its precondition. This is because $p_{5}$ maximizes $h_{max,0}^{L_{\alpha_{ij}}}$ and $h_{max,0}^{L_{\alpha_{ij}}}(a_{5}, s) = 3 > h_{max,0}^{L_{\alpha_{ij}}}(p_{5}, s) = 2$ (the globally maximizing fact is $p_{2} \notin J^{D_{\alpha_{ij}}}$).

In **Step 3** the $\alpha_{ij}$-projected landmark $L^{\alpha_{ij}}_{k}$ must be determined in a distributed manner. To obtain the same heuristic estimate as in the centralized version, the cost of $L_{\alpha_{ij}}^{\alpha_{ij}}$ must be equal to the centralized landmark $L_{\alpha_{ij}}$. This will be achieved by the place-holder action $\bar{a}$. But first, all facts from which $g$ is 0-reachable must be found.

**Step 3.1 Find all facts $p$ such that $g$ is $0$-reachable from $p$.** The algorithm starts as in $h_{lmc}$ and puts all facts from which $g$ is 0-reachable into $V_{k,i}$. As $g$ is public, all actions achieving $g$ are also public and the initiator $a_{i}$ knows about them (they or their projections are in $A_{\alpha_{i}}$), therefore the algorithm can be initiated by $a_{i}$. When the algorithm reaches some projected action $a_{\alpha_{i}}$ a request is sent to $a_{j}$ to determine all facts from which $a$ is 0-reachable. Agent $a_{j}$ finds all such facts, places them in $V_{k,j}$ and sends all such public facts $V_{k} = V_{k,j} \cap P_{pub}$ back to $a_{i}$. When received, $a_{i}$ finds all facts $p'$ s.t. some $g \in V_{k}^{a}$ is 0-reachable from $p'$ and adds $p'$ to $V_{k,i}^{a}$. This ensures that all facts $p$ from which $g$ is 0-reachable are found and added either to $V_{k,i}$ if $p \in P_{i}$ or to $V_{k,j}^{a}$ if $p \in P_{j}^{priv}$.

We illustrate this step on the iteration $k = 2$ of the example, where the cost of actions has already been modified so that $c_{2}(a_{1}) = 0, c_{2}(a_{2}) = 0, c_{2}(a_{5}) = 0$ and $h_{D_{lmc}}^{max,2} = \{p_{1} : 0, p_{2} : 0, p_{3} : 1, p_{4} : 1, g : 1\}$ and $h_{max}^{\alpha_{ij},2} = \{p_{3} : 3, p_{5} : 2, g : 1\}$. In this situation the justification graphs are:

Since the cost of $a_{2}$ and $a_{5}$ is 0, $p_{4}$ is added to $V_{2,1}^{a}$ and a request for $a_{5}$ is sent to $a_{2}$. Agent $a_{2}$ starts the reachability analysis from $a_{5}$ and finds that $a_{5}$ is 0-reachable only from $p_{5}$. Because $p_{5}$ is internal, the reply $V_{2,2}^{a}$ is empty, but $p_{5}$ is added to $V_{2,2}^{a}$ and will be used in the next step.
The distributed 0-reachability algorithm ensures, that $g$ is 0-reachable from a fact $p$ in $J$ if and only if it is 0-reachable in $J^{\alpha_i}$. When all such facts are stored in respective $V^*_{k,i}$ or $V^*_{k,j}$, the next step of the $h_{\text{tmc}}$ algorithm can be performed.

Step 3.2 Find all facts reachable from $i$ without visiting a fact in $V^*_{k,i}$ or any $V^*_{k,j}$. Again, the algorithm starts as in $h_{\text{tmc}}$. Similarly to the previous case, a fact $p$ may be reachable from $i$ via some agent $c_{k,i}$. To find all such facts, it is enough, to find all edges which contain a projected actions $\alpha^p_i$ in the label reachable from $i$ and for each such action send a request to $c_{k,i}$. Agent $\alpha^p_i$ then finds all facts reachable from all $q \in \text{add}(\alpha^p)$ without visiting any fact in $V^*_{k,i}$. All public actions in labels of edges visited in the process are added to $A^0_k$ and sent back in reply. When received, agent $\alpha^p_i$ finds all facts $p'$ reachable from all $i' \in \text{add}(\alpha^p)$ for all $\alpha^p \in A^0_k$ without visiting any fact in $V^*_{k,i}$.

Unlike the previous case, $i$ is not public and therefore additional request has to be sent for the initial fact $i$. The request and respective reply are handled the same way as in the case of a projected action.

Recall, that in the example, iteration $k = 2$, $V^*_{k,1} = \{p_1\}$ and $V^*_{k,2} = \{p_3\}$. The facts reachable in $J^{\alpha_i}$ without visiting $p_j$ are $\{i, p_1, p_2\}$. Requests are sent for $\alpha^p_3$ and for $i$. In $J^{\alpha_i}$, the facts reachable without visiting $p_j$ are $\{i, p_3\}$.

The distributed reachability algorithm ensures, that a fact $p$ is reachable from $i$ in $J_k$ if and only if it is reachable in $J^{\alpha_i}$.

Step 3.3 Find landmarks. In $h_{\text{tmc}}$, the purpose of the reachability analysis is to find actions forming the disjunctive landmark $L_k$. Those are all actions in labels of edges in $J_k$, starting from a fact reachable from $i$ and ending in fact in $V^*_k$. The distributed algorithm aims for the same.

In Step 3.2 performed by $\alpha^p_i$ on $J^{\alpha_i}$, action $a$ is added to $L^0_k$ if $a$ is in a label of edge reachable from $i$ ending in some $p \in V^*_{k,i}$, as in $h_{\text{tmc}}$. When the reply in Step 3.2 is computed by agent $\alpha^p_i$, for some requested projected action, landmark actions are added to $L^0_k$ private to $\alpha^p_i$, as in $h_{\text{tmc}}$. To capture the cost of private actions (which may possibly be the lowest cost actions), a place-holder action $\bar{a}$ is created and its cost set to $c_{\bar{a}}(\bar{a}) = c_{\alpha^p_i}(L^\alpha_k)$. The public part of the landmark $L^\text{pub}_k \leftarrow (L^\alpha_k \cap A^\text{pub}_k) \cup \{\bar{a}\}$ is sent in reply alongside the set of reached public actions $A^0_k$. When received, it is merged with $L^{\text{pub}}_k$ while keeping only the lowest-cost place-holder action $\bar{a}$.

In the $k = 0$ iteration of the example, the found landmarks are the following $L^0_k = \{a_2, a^p_2, a_3\}$ and $L^0_k = \{a_3\}$ where $c_0(\bar{a}) = c_0(a_2) = 1$. In this case $\bar{a}$ has no influence in the cost of $L^0_k$ which is 1. In the $k = 2$ iteration, $L^\alpha_k = \{a^\alpha_2, \bar{a}\}$ and $L^\alpha_k = \{a_4\}$ where $c_2(\bar{a}) = c_2(a_4) = 1$, whereas $c(a^\alpha_2) = 2$ and the information encoded in $\bar{a}$ is crucial.

In Step 4 of the distributed algorithm the cost function for the next iteration $k + 1$ is constructed. Thanks to the use of place-holder action $\bar{a}$ which stores the cost of the lowest-cost action over all $\alpha_j$-local landmarks, the same update formula as in $h_{\text{tmc}}$ can be used also in the distributed version. The only difference is that when the $c^0_m(L^\alpha_k)$ value is computed it is sent to all participating agents $\alpha_j$ so that the cost of actions in $L^\alpha_k$ can be locally updated as well.

Notice, that in the example iteration $k = 2$ the $\alpha_j$-projected landmark $L^{\alpha_j}_2 = \{a_4\}$ is missing the action $a_3$. It is not a problem for the computation of the cost of $L^\alpha_k$ because it contains $a_3$, but the cost of $a_3$ will not be updated. This issue can be handled in various ways, in $h_{\text{tmc}}$, it is handled in the computation of $h_{\text{tmc}}^{\alpha_i}$ where the updated cost of projected actions is sent from the initiator to the participants.

Equality of $h_{\text{tmc}}$ and $h_{\text{tmc}}^{\alpha_i}$

To show the equality of the centralized $h_{\text{tmc}}$ and distributed $h_{\text{tmc}}^{\alpha_i}$ heuristic, it is crucial to have the distributed $h_{\text{tmc}}^{\alpha_i}$ equal to the centralized $h_{\text{tmc}}$. This has been shown in Theorems 5 and 6. Then, a distributed justification graph has to be constructed, such that a reachability relation is preserved.

From the definition of $J^{\alpha_i}$ and the presented algorithms directly follows:

Lemma 9. Fact $q$ is reachable (0-reachable) from fact $p$ in a justification graph $J$ if and only if $q$ is reachable (0-reachable) from $p$ in a distributed justification graph $J^{\alpha_i}$.

We proceed by showing that in each iteration, the union of the set of projected landmarks constructed by the distributed algorithm is equal to the landmark constructed by the centralized algorithm (see Lemma 10) and its cost is equal to the cost of the projected landmark constructed by the initiator agent (see Lemma 11). We conclude the proof by showing that the heuristic estimate obtained by the distributed version is equal to the centralized estimate (see Theorem 12).

Lemma 10. For each step $k$, landmark $L_k$ constructed by the centralized algorithm on $J_k$ and landmarks $L^{\alpha_1}_k, ..., L^{\alpha_j}_k$, constructed by the distributed algorithm on $J^{\alpha_i}_k$ holds $L_k = \bigcup_{j=1}^{\alpha_i} L^{\alpha_j}_k \setminus \{\bar{a}\}$.

Proof. In each step $k$, $V^{\alpha}_k = \bigcup_{j=1}^{\alpha_i} V^{\alpha}_k$, holds (from Lemma 9). In the centralized search for landmarks, an action $a$ is added to $L_k$ if and only if $p \in \text{add}(a)$ exists s.t. $p \in V^*_k$ and $p$ is reachable from $i$. From the previously stated, for such $p$ must hold $p \in V^{\alpha}_k$ for some $j$ and from Lemma 9, $p$ is reachable from $i$ in $J^{\alpha_j}_k$. If $p \in P^{\alpha}_i$, $a$ is in $A^0_k$ and is added to $L^{\alpha}_k$, otherwise, $p$ is in some $P^{\text{priv}}_i$ and $a \in A^L_k$ and $a$ is added to $L^{\alpha}_k$. Therefore the lemma holds (the place-holder action $\bar{a}$ introduced by the distributed algorithm, is ignored).

The constructed $\alpha_i$-projected landmark represents the cost of the centralized landmark, formally:

Lemma 11. For each step $k$, landmark $L_k$ constructed by the centralized algorithm on $J$ and $\alpha_i$-projected landmark $L^{\alpha_i}_k$ constructed by the distributed algorithm initiated by agent $\alpha$, on $J^{\alpha_i}$ holds $c_m(L^{\alpha_i}_k) = c_m(L_k)$.
Proof. From proof of Lemma 10, for each \( \alpha \neq j \), \( L_{k}^{\alpha j} = L_{k} \cap A_{j} \). When \( L_{k}^{\alpha j} \) is finished, the public part \( I_{k}^{\text{pub}} = (L_{k}^{\alpha j} \cap \mathcal{A}_{j}^{\text{pub}}) \cup \{ a \} \) of \( L_{k}^{j} \) is sent from \( \alpha_{j} \) to \( \alpha_{i} \). For the place-holder action \( \bar{a} \) holds \( c_{\alpha_{j}}(\bar{a}) = c_{\alpha_{j}}^{lmc}(L_{k}^{j}) \). This ensures, that \( c_{k}^{lmc}(L_{k}^{j}) = c_{k}^{lmc}(L_{k}^{\alpha j}) \) even if the least-cost action is not public. When \( I_{k}^{\text{pub}} \) is received, \( L_{k}^{\alpha j} \leftarrow I_{k}^{\alpha j} \cup I_{k}^{\text{pub}} \), retaining the least-cost \( \bar{a} \). From the definition of \( c_{k}^{lmc} \) follows \( c_{k}^{lmc}(L_{k}^{j}) = \min(c_{k}^{lmc}(L_{k}^{\alpha j}), c_{k}^{lmc}(L_{k}^{\text{pub}})) \). Therefore, when \( I_{k}^{\text{pub}} \) is received from all agents \( \alpha \neq j \) and \( L_{k}^{\alpha j} \) is completed, \( c_{k}^{lmc}(L_{k}^{j}) = \min_{0 \leq j \leq n}(L_{k}^{\alpha j}) = \min_{0 < j \leq n}(L_{k}^{\alpha j}) \).

Finally we conclude that:

**Theorem 12.** For any state \( s \) and any agent \( \alpha \), \( h_{lmc}(G, s) = h^{\alpha_{i}}_{lmc}(G, s) \).

Proof. From Theorems 5 and 6 the result of distributed computation of \( h^{\alpha_{i}}_{lmc}(G, s) \) is equal to the centralized \( h^{\alpha_{i}}_{max}(G, s) \) for any state \( s \), any agent \( \alpha_{i} \) and for all facts \( p \in P \) and therefore also for all actions \( a \in A \). For each step \( k \) of the algorithm, a distributed justification graph \( J^{\alpha_{i}} = \{ J^{\alpha_{i}}, \{ L^{\alpha_{i}} \} \}_{j \neq i} \) can be constructed such that Lemma 9 holds for reachability and 0-reachability. Also, from Lemma 11 the cost of the projected landmark \( L_{k}^{\alpha_{i}} \) constructed by the distributed algorithm initiated by \( \alpha_{i} \) equals the cost of the landmark \( L_{k} \) constructed in step \( k \) by the centralized algorithm. The cost is then shared with all agents \( \alpha_{j} \neq i \) and all actions in \( L_{k}^{\alpha_{j}} \) and all \( L_{k}^{\alpha_{j}} \), which are all actions in \( L_{k} \) (from Lemma 10), have their costs updated. Therefore, the updated cost function in the \( k + 1 \) step of the centralized algorithm equals the cost function in the \( k + 1 \) step of the distributed algorithm for all agents and all actions.

**Experimental Evaluation**

We have evaluated the presented heuristics using a planner based on the MAD-A*(Nissim and Brafman 2012) search algorithm on a set of benchmarks commonly used in the MA planning literature, derived from the classical IPC benchmarks. Each run (per problem) of the planner was limited to 60 min. and 4GB of memory (total for all agents) on a 16 core machine.

The used benchmarks are blocksworld (multiple hands treated as agents), depot (trucks, depots and distributors are agents), driverlog (drivers are agents), ma-sokoban (sokoban with multiple robots as agents), woodworking (the instruments are agents) and elevators, logistics, rovers, satellites and zenotravel (elevators, planes and trucks, rovers, satellites and planes are the respective agents).

The results of the experiments are summarized in Table 1. The coverage results (the number of problems solved for each domain) show that except for three domains, the distributed \( h^{\alpha_{i}}_{lmc} \) solves more (or the same) problems and solves also a more of problems in total. The depot, driverlog and ma-sokoban domains are tightly coupled (as in (Brafman and Domshlak 2008)) and most of the information is public.

<table>
<thead>
<tr>
<th>domain</th>
<th>( h^{\alpha_{i}}_{lmc} )</th>
<th>( h^{\alpha_{i}}_{lmc} )</th>
<th>( h^{\alpha_{i}}_{lmc} )</th>
<th>( t_{lmc} )</th>
<th>( t_{lmc} )</th>
<th>( t_{lmc} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>elevators08 (20)</td>
<td>2</td>
<td>2</td>
<td>0.18</td>
<td>39.9</td>
<td>0.15</td>
<td>0.01</td>
</tr>
<tr>
<td>logistics00 (20)</td>
<td>6</td>
<td>12</td>
<td>0.26</td>
<td>2521.7</td>
<td>4.78</td>
<td>0.01</td>
</tr>
<tr>
<td>zenotravel (18)</td>
<td>6</td>
<td>10</td>
<td>0.41</td>
<td>142.4</td>
<td>0.74</td>
<td>0.03</td>
</tr>
<tr>
<td>rovers (18)</td>
<td>6</td>
<td>6</td>
<td>0.52</td>
<td>33.7</td>
<td>0.45</td>
<td>0.15</td>
</tr>
<tr>
<td>blocksworld (30)</td>
<td>17</td>
<td>20</td>
<td>0.54</td>
<td>45.5</td>
<td>0.36</td>
<td>0.09</td>
</tr>
<tr>
<td>satellites (18)</td>
<td>5</td>
<td>10</td>
<td>0.55</td>
<td>63.9</td>
<td>0.52</td>
<td>0.03</td>
</tr>
<tr>
<td>driverlog (20)</td>
<td>13</td>
<td>12</td>
<td>0.8</td>
<td>4.4</td>
<td>0.22</td>
<td>0.12</td>
</tr>
<tr>
<td>depot (20)</td>
<td>7</td>
<td>4</td>
<td>0.88</td>
<td>1.4</td>
<td>0.13</td>
<td>0.11</td>
</tr>
<tr>
<td>woodworking08 (20)</td>
<td>6</td>
<td>8</td>
<td>0.88</td>
<td>12</td>
<td>1.11</td>
<td>0.35</td>
</tr>
<tr>
<td>ma-sokoban (10)</td>
<td>8</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0.15</td>
<td>0.14</td>
</tr>
<tr>
<td><strong>total</strong> (194)</td>
<td><strong>76</strong></td>
<td><strong>89</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 2: Per-problem \( h^{\alpha_{i}}_{lmc} / h^{\alpha_{i}}_{lmc} \) ratios for initial state.

which means that the projected \( h^{\alpha_{i}}_{lmc} \) has the same information as \( h^{\alpha_{i}}_{lmc} \) Moreover, \( h^{\alpha_{i}}_{lmc} \) has to handle a lot of projected actions.

As expected, both variants of the \( h^{\alpha_{i}}_{lmax} \) heuristic perform significantly worse (total coverage of 59 for the projected and 53 for the distributed version) and are not presented in the table. Except for the ma-sokoban domain (coverage 7 for \( h^{\alpha_{i}}_{lmax} \) and 0 for \( h^{\alpha_{i}}_{lmax} \)) and elevators domain (coverage 0 for \( h^{\alpha_{i}}_{lmax} \) and 2 for \( h^{\alpha_{i}}_{lmax} \)), the difference between \( h^{\alpha_{i}}_{lmax} \) and \( h^{\alpha_{i}}_{lmax} \) is not significant.

To understand the cause of the behavior of \( h^{\alpha_{i}}_{lmc} \) and \( h^{\alpha_{i}}_{lmc} \), better, we have extracted the heuristic values computed for the initial state by both heuristics (for \( h^{\alpha_{i}}_{lmc} \). taking average for all agents), computed a ratio \( h^{\alpha_{i}}_{lmc} / h^{\alpha_{i}}_{lmc} \) for each problem and averaged the ratios per domain. The results are in Table 1 in the column labeled \( h^{\alpha_{i}}_{lmc} / h^{\alpha_{i}}_{lmc} \) and for each problem and averaged the ratios per domain. The results are for coverage show that, in the tightly coupled domains, the distributed evaluation does not improve the heuristic estimate enough to justify the communication overhead. On the other hand, as the ratio drops below approx. 0.8, the improved heuristic accuracy overweight the communication overhead.

For more detailed view, the heuristic ratios aggregated in...
the column labeled \( \hat{h}_{\text{lmc}} \) are plotted per-problem in Figure 2. The plot raises a question whether the distributed heuristic should not dominate the projected one as was shown for \( h_{\text{max}} \) in Lemma 2. Against intuition, the answer is no. The reason lies in the inherent variance of the heuristic depending on the tie-breaking behavior of the precondition choice function \((\text{pef})\). Although in the proofs of equality of \( h_{\text{lmc}} \) and \( h_{\text{lmc}}^{\alpha} \) it was possible to fix the tie-breaking behavior (thanks to use of distributed \( h_{\text{max}} \)), it is not the case with \( h_{\text{lmc}}^{\alpha} \). The fact \( p \) which maximizes \( h_{\text{max}} \) in \( h_{\text{lmc}} \) may not maximize it in \( h_{\text{lmc}}^{\alpha} \) (or vice versa) therefore the same fact \( p \) could not be chosen in both. If most of the actions in the problem are public, this may lead to situation that for some state \( h_{\text{lmc}}^{\alpha} > h_{\text{lmc}}^{\alpha} \), as can be seen in Figure 2 for some of the depot problems.

The quality of heuristic estimates can be assessed by the number of expanded states. In Table 1, the column \( \epsilon_{\text{lmc}} \) shows the average ratio of expanded states, restricted to problems solved by both heuristics. The most significant improvement is in the logistics domain, where \( h_{\text{lmc}}^{\alpha} \) expands over \( 2500 \times \) more states than \( h_{\text{lmc}}^{\alpha} \) followed by zenotravel and satellites with approx. \( 140 \times \) and \( 60 \times \) increase of expanded states over the distributed heuristic respectively. The limiting factor is \( 30 \times \) in rovers domain where the quality of the distributed heuristic just eliminates the overhead of the distributed heuristic, below this factor the projected heuristic exhibits better performance.

The total planning time (\( t_{\text{lmc}} \) in Table 1) and time per expanded state (\( t_{\text{lmc}}^* \) in Table 1) were treated similarly, computed only from problems solved by both heuristics. The results show that, except for the logistics and woodworking domains, the projected heuristic leads to approx. \( 2 \times 10 \) faster solution, which was not unexpected. The average time spent on an expanded state shows that the projected heuristic is \( 10 \times \) faster. Even though, the added value of better heuristic estimates is crucial in many domains, most notably logistics. Notice that in domains with the largest difference in the number of expanded states, the projected heuristic is significantly faster. This suggests that the projected problem is much simpler, but ignores a lot of important information thus makes the resulting heuristic estimate much less accurate.

A specific case is the woodworking domain. Even though the distributed heuristic estimates are only slightly better than the projected ones, the distributed heuristic solves more problems as the projected heuristic is only \( 3 \times \) faster and ignores important information.

The structural properties causing success of the \( h_{\text{lmc}}^{\alpha} \) heuristic are closely related to the motivation example in the introduction. An example is the logistics domain, where trucks and planes are moving packages from starting to goal locations. In the MA-STRIPS formulation, the location of a package is public only when it is at an intermediate (or goal) location, it is not known when loaded onto some vehicle. The unload action of a vehicle is seen by other agents with a precondition only on the location of the vehicle, having the package actually loaded is not required. The cost of getting the package to a location where it can be loaded and load-

ing it is lost in the projected problem, enabling the agents to have a package cheaply unloaded at their loading site by a projected unload action of some other agent. This is exactly the principle demonstrated in the motivation example. Similar situation occurs in the elevators domain and other loosely coupled domains.

Conclusions & Future Work

We have presented an algorithm to distributively compute a global LM-Cut heuristic estimate in a MA-STRIPS problem. We have shown both its theoretical properties and practical applicability. It is clear that the pilot implementation is competitive with the state-of-the-art projected version, and outperforms it on the class of loosely coupled problems. It is imaginable that further optimizations of the algorithm may decrease the communication overhead so that the improved heuristic estimate pays off in more domains, but there are limits which can hardly be overcome - problems in which the projected heuristic estimate pays off in more domains, but there are limits which can hardly be overcome - problems in which the projected heuristic estimate pays off in more domains, but there are limits which can hardly be overcome - problems in which the projected heuristic estimate pays off in more domains, but there are limits which can hardly be overcome - problems in which the projected heuristic estimate pays off in more domains, but there are limits which can hardly be overcome.

Acknowledgments

This research was supported by the Czech Science Foundation (grant no. 13-22125S) and by the Grant Agency of the CTU in Prague (grant no. SGS14/202/ OHK3/3T/13). Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum, provided under the program “Projects of Large Infrastructure for Research, Development, and Innovations” (LM2010005), is greatly appreciated.

References


