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 comunicate 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).

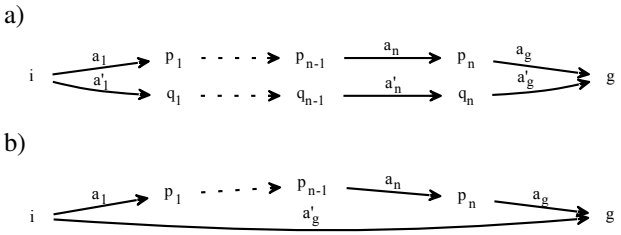


Figure 1: The full (a) and projected (b) problem.

In Figure 1a, $a_1, ..., a_n$ and $a'_1, ..., a'_n$ 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'_g of the other agent. Moreover it does not know about its precondition and considers a'_g 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 $N = \{1, ..., n\}$ agents $\mathcal{A} = \{\alpha_i\}_{i \in N}$ 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 $i \in N$, A_i is the set of actions agent α_i can perform and is planning with. The set of actions are pairwise disjoint, that is $\forall i \neq j : A_i \cap A_j = \emptyset$. Each action $a \in A = \bigcup A_i$ has the standard STRIPS syntax and semantics, i.e. $a = \langle \mathsf{pre}(a), \mathsf{add}(a), \mathsf{del}(a) \rangle$ is given by its preconditions, add effects and delete effects. We will denote $\mathsf{facts}(a) = \mathsf{pre}(a) \cup \mathsf{add}(a) \cup \mathsf{del}(a)$.

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

For an agent α_i, we define the following:

P_i	$\bigcup_{a \in A_i} \mathsf{facts}(a)$	set of facts
P_i^{priv}	$P_i \backslash \bigcup_{j \neq i} P_j$	set of private facts
P_i^{pub}	$P_i \backslash P_i^{\mathsf{priv}}$	set of public facts
A_i^{priv}	$\{a \in A_i \mid \mathsf{facts}(a) \subseteq P_i^{\mathsf{priv}}\}$	set of priv. actions
A_i^{pub}	$A_i \backslash A_i^{\mathsf{priv}}$	set of pub. actions
s^{α_i}	$s \cap P_i$	α_i-projection of state s
a^{α_i}	$\mathsf{pre}(a^{\alpha_i}) = \mathsf{pre}(a) \cap P_i$ $\mathsf{add}(a^{\alpha_i}) = \mathsf{add}(a) \cap P_i$ $\mathsf{del}(a^{\alpha_i}) = \mathsf{del}(a) \cap P_i$	α_i-projection of action $a \in A$
$a^{\mathsf{L}\alpha_i}$	$\mathsf{pre}(a^{\alpha_i}) = \mathsf{pre}(a) \cap P_i^{\mathsf{priv}}$ $\mathsf{add}(a^{\alpha_i}) = \mathsf{add}(a) \cap P_i^{\mathsf{priv}}$ $\mathsf{del}(a^{\alpha_i}) = \mathsf{del}(a) \cap P_i^{\mathsf{priv}}$	α_i-local action corresponding to $a \in A_i$
A_i^{L}	$\{a^{\mathsf{L}\alpha_i} \mid a \in A_i\}$	set of α_i-local actions
P_i^{all}	$P_i \cup \bigcup_{j \neq i} \{p \mid p \in P_j^{\mathsf{pub}}\}$	set of all known facts
A_i^{all}	$A_i \cup \bigcup_{j \neq i} \{a^{\alpha_i} \mid a \in A_j^{\mathsf{pub}}\}$	set of all known actions

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

STRIPS problems derived from the MA-STRIPS problem Π are a **global problem** $\Pi^G = \left\langle P, \bigcup_{i \in N} A_i, I, G \right\rangle$, an **$\alpha_i$-projected problem** $\Pi^{\alpha_i} = \left\langle P_i^{\mathsf{all}}, A_i^{\mathsf{all}}, I \cap P_i^{\mathsf{all}}, G \right\rangle$ and an **α_i-local problem** $\Pi^{\mathsf{L}\alpha_i} = \left\langle P_i^{\mathsf{priv}}, A_i^{\mathsf{L}}, I \cap P_i^{\mathsf{priv}}, \emptyset \right\rangle$. All the defined problems can be extended with a cost function $c : A \to \mathbb{N}^{0+}$ (restricted to the used subset of actions). The costs of an action and of its projection are equal.

An **α_i-projected heuristic** is a heuristic computed on the α_i-projected problem Π^{α_i}. Any admissible heuristic computed on an α-projected problem Π^{α_i} is admissible also for the global problem Π^G.

For a proof sketch observe that a path in Π^{α_i} can be constructed from a path in Π^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 α_i-projected heuristic gives higher estimate than the global heuristic. The least-cost path π^{α_i} from s to goal in Π^{α_i} has to have higher cost than the least-cost path ϖ from s to goal in Π^G (follows from admissibility of both heuristics in their respective problems). But that is not possible, because the cost of path ϖ^{α_i} in Π^{α_i} constructed from ϖ is less or equal to the cost of ϖ. The least-cost path in Π^{α_i} cannot have higher cost than ϖ^{α_i}.

Distributed h_{max} Heuristic

Both h_{max} and LM-Cut heuristics fall into the class of relaxation heuristics. Relaxation heuristics base the estimate on a solution of a relaxed problem Π^+, in which all delete effects are ignored (for all actions, $a^+ = \langle \mathsf{pre}(a), \mathsf{add}(a), \emptyset \rangle$). Although the cost of optimal solution to Π^+ 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_{max} heuristic is computed in each iteration. In the following, we introduce the h_{max} heuristic and provide a method for its distributed computation. A distributed h_{max} was already described in (Štolba 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 Π^G, formally $O(p) = \{a \in A | p \in \mathsf{add}(a)\}$. Similarly $O^{\alpha_i}(p) = \{a \in A_i^{\mathsf{all}} | p \in \mathsf{add}(a)\}$ in Π^{α_i} and $O^{\mathsf{L}\alpha}(p) = \{a \in A_i^{\mathsf{L}} | p \in \mathsf{add}(a)\}$ in $\Pi^{\mathsf{L}\alpha}$.

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

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

$$h_{max}(p, s) = \begin{cases} 0 & p \in s \\ h_{max}(\underset{a \in O(p)}{argmin}(h_{max}(a, s)), s) & \\ & p \notin s \end{cases} \quad (2)$$

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

where P is a set of facts (goal or action preconditions). Throughout the text, projected $h_{max}^{\alpha_i}$ is h_{max} computed on Π^{α_i}, local $h_{max}^{\mathsf{L}\alpha_i}$ is h_{max} computed on $\Pi^{\mathsf{L}\alpha_i}$ and distributed $h_{max}^{\mathsf{D}\alpha_i}$ is h_{max} computed on Π, but estimating Π^G.

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

Distributed h_{max} Algorithm

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

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

When received (line 6), the $h^{\mathsf{D}\alpha_i}_{max}$ values are updated based on the received reply (line 7). If no actions are updated, the loop exits, the algorithm terminates and returns heuristic value for the goal.

Algorithm 1: Distributed $h^{D\alpha_i}_{max}$ heuristic

```
1 Algorithm computeDistHmax(α_i, s, G)
2     initialize h^{Dα_i}_max to h^{α_i}_max for all p ∈ P_i
3     while h^{Dα_i}_max(a, s) changed for some a ∈ A^all_i do
4         for each agent α_j ∈ A\{α_i} do
5             send h^{Dα_i}_max(a^{α_i}, s) for all a ∈ A^pub_j to α_j
6             receive h^{Lα_j}_max(a^{α_i}, s) for all a ∈ A^pub_j
7             update h^{Dα_i}_max
8     return h^{Dα_i}_max(G, s)
```

Equality of h_{max} and $h^{\mathsf{D}\alpha_i}_{max}$

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

Computing the $h^{\mathsf{L}\alpha_j}_{max}$ updates based on the public action and the initial state is shown to be sufficient (see Lemma 3). As a consequence, each $p \in P_i$ such that $h^{\mathsf{D}\alpha_i}_{max}(p, s) < h_{max}(p, s)$ is updated to $h^{\mathsf{D}\alpha_i}_{max}(p, s) = h_{max}(p, s)$ after finitely many steps (see Lemma 4).

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

Lemma 2. *For each state s and fact $p \in P_i$, $h^{\alpha_i}_{max}(p, s) \leq h_{max}(p, s)$.*

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

To correctly update a fact or action value in the α_j-local h_{max} it is enough to provide correct h_{max} values for all preceding public actions of agent α_j. An action a is **preceded** by action a' iff $\mathsf{add}(a') \cap \mathsf{pre}(a) \neq 0$ 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 $h^{\mathsf{L}\alpha_i}_{max}(a^{\mathsf{L}\alpha_i}, s) = h_{max}(a, s)$ for all $a \in A^{\mathsf{pub}}_i$ preceding a', then $h^{\mathsf{L}\alpha_i}_{max}(a'^{\mathsf{L}\alpha_i}, s) = h_{max}(a', s)$.*

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

To conclude the proof of equality of $h^{\mathsf{D}\alpha_i}_{max}$ and h_{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 $p \in P_i$ such that $h^{\mathsf{D}\alpha_i}_{max}(p, s) < h(p, s)$ is updated to $h^{\mathsf{D}\alpha_i}_{max}(p, s) = h(p, s)$ after finitely many steps.*

Proof. Distributed $h^{\mathsf{D}\alpha_i}_{max}$ is initialized to $h^{\alpha_i}_{max}$ for all facts and actions. As already shown in the proof of Lemma 2, if $h^{\alpha_i}_{max}(a, s) \leq h_{max}(a, s)$ holds for some action $a \in A_i$, it is caused by some projected action preceding a and missing the (private) precondition maximizing h_{max}. Let $a^{\alpha_i}_0$ (where $a_0 \in A_{j\neq i}$) be such a projected action preceding a for which $h^{\mathsf{D}\alpha_i}_{max}(a^{\alpha_i}_0, s) \leq h_{max}(a_0, s)$. Let for all $p \in \mathsf{pre}(a^{\alpha_i}_0)$ hold $h^{\mathsf{D}\alpha_i}_{max}(p, s) = h_{max}(p, s)$, which means that all actions preceding $a^{\alpha_i}_0$ already have $h^{\mathsf{D}\alpha_i}_{max}$ equal to h_{max}. Such action a_0 always exists, because as $h^{\mathsf{D}\alpha_i}_{max}$ is initialized to $h^{\alpha_i}_{max}$, all actions applicable in I or preceded only by actions in A_i have already $h^{\mathsf{D}\alpha_i}_{max}$ equal to h_{max}.

The inequality $h^{\mathsf{D}\alpha_i}_{max}(a^{\alpha_i}_0, s) \leq h_{max}(a_0, s)$ is caused by a fact $p_m \in \mathsf{pre}(a_0)\backslash P_i$ maximizing $h_{max}(a_0, s)$. The action a_0 is sent alongside all other actions in A^{pub}_j to agent α_j in order to obtain an update. Agent α_j computes the updated heuristic for all actions from the local problem $\Pi^{\mathsf{L}\alpha_i}$ and sends the information back.

From Lemma 3 and because we assumed that, for all actions $a' \in A^{\mathsf{pub}}_j$ preceding a_0, the equality $h^{\mathsf{D}\alpha_i}_{max}(a'^{\alpha_i}, s) = h_{max}(a', s)$ holds, it holds also for the returned value of a_0. Subsequently, $h^{\mathsf{D}\alpha_i}_{max}$ is updated so that $h^{\mathsf{D}\alpha_i}_{max}(a^{\alpha_i}_0, s) = h_{max}(a_0, s)$. In the next iteration, for some other action a_1 preceding a holds $h^{\mathsf{D}\alpha_i}_{max}(a^{\alpha_i}_1, s) \leq h_{max}(a_1, s)$ while for all actions preceding a_1 the equality holds. The same reasoning can be applied to a_1. 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 $h^{\mathsf{D}\alpha_i}_{max}(p, s) = h_{max}(p, s)$ for any given state s and all $p \in P_i$.*

Proof. For each $a \in A^{\alpha}_i$, when $h^{\mathsf{D}\alpha_i}_{max}(p, s) = h_{max}(p, s)$, 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 $h^{\mathsf{D}\alpha_i}_{max}(p,s) = h(p,s)$ 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 $h^{\mathsf{L}\alpha_j}_{max}(p,s)$ for the given state s and all facts $p \in P^{\mathsf{priv}}_j$, will be preserved throughout the computation. For $h^{\mathsf{L}\alpha_i}_{max}$, the equality holds as well.

Theorem 6. *Algorithm 1 terminates with $h^{\mathsf{L}\alpha_j}_{max}(p,s) = h_{max}(p,s)$ for all $j \neq i$, $p \in P_j$ and any given state s.*

Proof. From Theorem 5 the equality of $h^{\mathsf{D}\alpha_i}_{max}$ and h_{max} holds for all fact and thus for all actions. It holds also for all projections of public actions $a \in A^{\mathsf{pub}}_j$. From Lemma 3, $h^{\mathsf{L}\alpha_j}_{max}(a^{\mathsf{L}\alpha_j}, s) = h_{max}(a,s)$ for all $a \in A_j$ (and for all $p \in P_j$).

□

Distributed Landmark Heuristic

The LM-Cut heuristic (will be denoted as h_{lmc}) provides an admissible estimate of the optimal plan for a relaxed problem Π^+ by utilizing the idea of *disjunctive landmarks*. Here, we present a distributed version h^{D}_{lmc} for which we show $h^{\mathsf{D}}_{lmc}(s) = h_{lmc}(s)$ 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 $c^{lm}(L) = min_{a\in L} c(a)$, where $c(a)$ is the cost of action a.

The h_{lmc} heuristic is obtained from a sequence $\{(L_k, c_k)\}^m_{k=0}$ of landmarks and cost functions, $h_{lmc} = c^{lm}_0(L_0) + c^{lm}_1(L_1) + ... + c^{lm}_m(L_m)$. 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_p for each $p \in I$ s.t. $\mathsf{pre}(a_p) = \{\mathsf{i}\}$ and $\mathsf{add}(a_p) = \{p\}$. 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 $\mathsf{pre}(a) = \emptyset$ and adding a dummy effect $\perp$ for actions for which $\mathsf{add}(a) = \emptyset$.

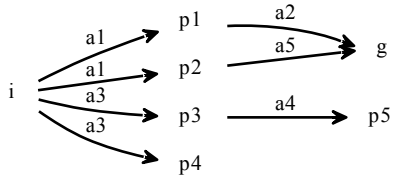
The algorithm will be illustrated on a STRIPS example with a set of 5 actions $\{a_1, ..., a_5\}$ (later, in a MA-STRIPS formulation, the actions will be divided among two agents α_1, α_2):

A_1: $a_1 : \mathsf{i} \to p_1, p_2$ $\quad a_2 : p_1, p_4 \to \mathsf{g}$
A_2: $a_3 : \mathsf{i} \to p_3, p_4$ $\quad a_4 : p_3 \to p_5$
$a_5 : p_2, p_5 \to \mathsf{g}$

The facts are $\{\mathsf{i}, \mathsf{g}, p_1, ..., p_5\}$, 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 $h^0_{max} = \{p_1 : 3, p_2 : 3, p_3 : 1, p_4 : 1, p_5 : 2, \mathsf{g} : 3\}$.

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 $q \in \mathsf{add}(a)$. Function pcf (a precondition choice function) assigns to a given action a one of its preconditions. In h_{lmc}, the pcf assigns a precondition maximizing h_{max}, ties broken arbitrarily. In the example, $pcf = \{a_1 \mapsto \mathsf{i}, a_2 \mapsto p_1, a_3 \mapsto \mathsf{i}, a_4 \mapsto p_3, a_5 \mapsto p_2\}$, resulting in J_0:



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. $c_k(a) = 0$ (g 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. 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 $L_0 = \{a_2, a_5\}$.

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 $c_2(a_5) = 0$, for all other actions it is the same as c_0.

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

Algorithm 2: LM-Cut Heuristic

1. Compute h^k_{max} based on c_k for every $p \in P$. If $h^k_{max}(\mathsf{g}) = 0$ terminate and return h_{lmc}.
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 $c_{k+1}(a) = \begin{cases} c_k(a) & a \notin L_k \\ c_k(a) - c^{lm}_k(L_k) & a \in L_k \end{cases}$
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 $h^{\mathsf{D}\alpha_i}_{max}$, that is the

heuristic values for all $p \in P_j^{\mathsf{priv}}$ and $a \in A_j^{\mathsf{L}\alpha}$. 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 distributed version of the heuristic we introduce a projected version of landmarks:

Definition 8. An α_i**-projected disjunctive landmark** (or α_i-projected landmark) L^{α_i} corresponding to disjunctive landmark L is $L^{\alpha_i} = (L \cap A_i^{\alpha}) \cup \{\bar{a}\}$, 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, $c(\bar{a}) = c^{lm}(L)$.

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** $J^{\mathsf{D}\alpha_i} = (J^{\alpha_i}, \{J^{\mathsf{L}\alpha_j}\}_{j\neq i})$ consists of an α_i-projected justification graph J^{α_i} and a set of α_j-local justification graphs $\{J^{\mathsf{L}\alpha_j}\}_{j\neq i}$.

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

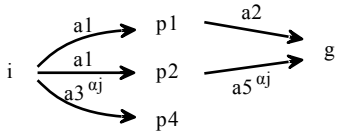
The distributed justification graph is a distributed graph where the partitions have pairwise disjunctive sets of vertices. Each edge in J^{α_i} with label containing a projected action a^{α_i} s.t. $a \in A_{j\neq i}$ can be seen as an edge shared with $J^{\mathsf{L}\alpha_j}$, where the corresponding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

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 $\{p_2, p_4, \mathsf{g}\}$ are public facts and $\{a_1, a_2, a_3, a_5\}$ public actions.

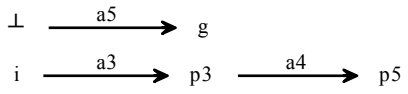
In **Step 1** of the k-th iteration, distributed version of h_{max} is computed based on the cost function c_k. The initiator agent α_i computes $h^{\mathsf{D}\alpha_i,k}_{max}$ for all facts in P_i while all other agents $\alpha_{j\neq i}$ compute $h^{\mathsf{L}\alpha_j,k}_{max}$ for all facts in P_j^{priv}. The computed values are identical to the values of centralized h_{max} (Theorems 5 and 6).

In **Step 2** the initiator agent α_i builds an α_i-projected justification graph $J_k^{\alpha_i}$ based on the values of $h^{\mathsf{D}\alpha_i,k}_{max}$ whereas all other agents build α_j-local justification graph $J_k^{\mathsf{L}\alpha_j}$ based on the values of $h^{\mathsf{L}\alpha_j,k}_{max}$, together forming a distributed justification graph $J_k^{\mathsf{D}\alpha_i}$. In our example, the justification graphs are the following:

$J_0^{\alpha_1}$:



$J_0^{\mathsf{L}\alpha_2}$:



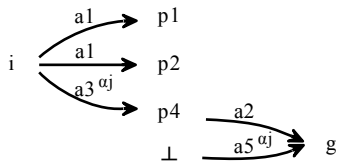
Notice, that a_5 has $\bot$ as its precondition. This is because p_5 maximizes $h^{\mathsf{L}\alpha_2,0}_{max}$ and $h^{\mathsf{L}\alpha_2,0}_{max}(a_5, s) = 3 > h^{\mathsf{L}\alpha_2,0}_{max}(p_5, s)$=2 (the globally maximizing fact is $p_2 \notin P_2^{\mathsf{priv}}$).

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

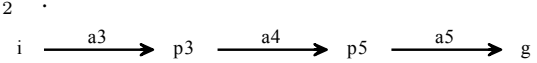
Step 3.1 Find all facts p **such that** g **is 0-reachable from** p**.** The algorithm starts as in h_{lmc} and puts all facts from which g is 0-reachable into $V^*_{k,i}$. As g is public, all actions achieving g are also public and the initiator α_i knows about them (they or their projections are in A_i^{α}), therefore the algorithm can be initiated by α_i. When the algorithm reaches some projected a^{α_i} a request is sent to α_j to determine all facts from which a is 0-reachable. Agent α_j finds all such facts, places them in $V^*_{k,j}$ and sends all such public facts $V_k^a = V^*_{k,j} \cap P_j^{\mathsf{pub}}$ back to α_i. When received, α_i finds all facts p' s.t some $q \in V_k^a$ is 0-reachable from p' and adds p' to $V^*_{k,i}$. 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 $V^*_{k,j}$ if $p \in P_j^{\mathsf{priv}}$.

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 $h^{\mathsf{D}\alpha_1,2}_{max} = \{p_1 : 0, p_2 : 0, p_3 : 1, p_4 : 1, g : 1\}$ and $h^{L\alpha_2,2}_{max} = \{p_3 : 1, p_5 : 2, \mathsf{g} : 1\}$. In this situation the justification graphs are:

$J_2^{\alpha_1}$:



$J_2^{\mathsf{L}\alpha_2}$:



Since the cost of a_2 and $a_5^{\alpha_1}$ is 0, p_4 is added to $V^*_{2,1}$ and a request for a_5 is sent to α_2. Agent α_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 $V_2^{a_5}$ is empty, but p_5 is added to $V^*_{2,2}$ 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 $J_2^{\mathsf{D}\alpha_i}$. When all such facts are stored in respective $V^*_{k,i}$ or $V^*_{k,j}$, the next step of the $h^{\mathsf{D}\alpha_i}_{lmc}$ 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_{lmc}. Similarly to the previous case, a fact p may be reachable from i via some agent $\alpha_{j\neq i}$. To find all such facts, it is enough, to find all edges which contain a projected actions a^{α_j} in the label reachable from i and for each such action send a request to α_j. Agent α_j then finds all facts reachable from all $q \in \mathsf{add}(a)$ without visiting any fact in $V^*_{k,j}$. 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 α_i finds all facts p' reachable from all $i' \in \mathsf{add}(a')$ for all $a' \in A^0_k$ 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 $k = 2$, $V^*_{2,1} = \{p_4\}$ and $V^*_{2,2} = \{p_5\}$. The facts reachable in $J_2^{\alpha_1}$ without visiting p_4 are $\{\mathsf{i}, p_1, p_2\}$. Requests are sent for $a_3^{\alpha_1}$ and for i. In $J_2^{\mathsf{L}\alpha_2}$, the facts reachable without visiting p_5 are $\{\mathsf{i}, p_3\}$.

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

Step 3.3 Find landmarks. In h_{lmc}, 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 α_i on $J_k^{\alpha_i}$, action a is added to $L_k^{\alpha_i}$ if a is in a label of edge reachable from i ending in some $p \in V^*_{k,i}$, as in h_{lmc}. When the reply in Step 3.2 is computed by agent α_j for some requested projected action, landmark actions are added to $L_k^{\alpha_j}$ private to α_j, again as in h_{lmc}. 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 $c_k(\bar{a}) = c^{lm}_k(L_k^{\alpha_j})$. The public part of the landmark $L_k^{\mathsf{pub}_j} \leftarrow (L_k^{\alpha_j} \cap A_j^{\mathsf{pub}}) \cup \{\bar{a}\}$ is sent in reply alongside the set of reached public actions A^0_k. When received, it is merged with $L_k^{\alpha_i}$ 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 $L_0^{\alpha_1} = \{a_2, a_5^{\alpha_1}, \bar{a}\}$ and $L_0^{\alpha_j} = \{a_5\}$ where $c_0(\bar{a}) = c_0(a_5) = 1$. In this case $\bar{a}$ has no influence in the cost of $L_0^{\alpha_1}$ which is 1. In the $k = 2$ iteration, $L_2^{\alpha_1} = \{a_3^{\alpha_1}, \bar{a}\}$ and $L_2^{\alpha_j} = \{a_4\}$ where $c_2(\bar{a}) = c_2(a_4) = 1$, whereas $c(a_3^{\alpha_1}) = 2$ 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 α_j-local landmarks, the same update formula as in h_{lmc} can be used also in the distributed version. The only difference is that when the $c^{lm}_k(L_k^{\alpha_i})$ value is computed it is sent to all participating agents α_j so that the cost of actions in $L_k^{\alpha_j}$ can be locally updated as well.

Notice, that in the example iteration $k = 2$ the α_j-projected landmark $L_2^{\alpha_j} = \{a_4\}$ is missing the action a_3. It is not a problem for the computation of the cost of $L_2^{\alpha_1}$, because it contains $a_3^{\alpha_1}$, but the cost of a_3 will not be updated. This issue can be handled in various ways, in h^{D}_{lmc} it is handled in the computation of $h^{\mathsf{D}\alpha_1,3}_{max}$ where the updated cost of projected actions is sent from the initiator to the participants.

Equality of h_{lmc} and $h^{D\alpha_i}_{lmc}$

To show the equality of the centralized h_{lmc} and distributed h^{D}_{lmc} heuristic, it is crucial to have the distributed h^{D}_{max} equal to the centralized h_{max}. 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 $J^{D\alpha_i}$ and the presented algorithms directly follows:

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

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 $L_k^{\alpha_0}, ..., L_k^{\alpha_n}$ constructed by the distributed algorithm on $J_k^{D\alpha_i}$ holds $L_k = \bigcup_{j=1}^{n} L_k^{\alpha_j} \setminus \{\bar{a}\}$.*

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

The constructed α_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 α_i-projected landmark $L_k^{\alpha_i}$ constructed by the distributed algorithm initiated by agent α_i on $J^{D\alpha_i}$ holds $c_{lm}(L_k^{\alpha_i}) = c_{lm}(L_k)$.*

Proof. From proof of Lemma 10, for each $\alpha_{j\neq i}$, $L_k^{\alpha_j} = L_k \cap A_j$. When $L_k^{\alpha_j}$ is finished, the public part $L_k^{\mathsf{pub}} = (L_k^{\alpha_j} \cap A_j^{\mathsf{pub}}) \cup \{\bar{a}\}$ of $L_k^{\alpha_j}$ is sent from α_j to α_i. For the place-holder action $\bar{a}$ holds $c_k(\bar{a}) = c_k^{lm}(L_k^{\alpha_j})$. This ensures, that $c_k^{lm}(L_k^{\mathsf{pub}}) = c_k^{lm}(L_k^{\alpha_j})$ even if the least-cost action is not public. When L_k^{pub} is received, $L_k^{\alpha_i} \leftarrow L_k^{\alpha_i} \cup L_k^{\mathsf{pub}}$, retaining the least-cost $\bar{a}$. From the definition of c_k^{lm} follows $c_k^{lm}(L_k^{\alpha_i}) = min(c_k^{lm}(L_k^{\alpha_i}), c_k^{lm}(L_k^{\mathsf{pub}}))$. Therefore, when L_k^{pub} is received from all agents $\alpha_{j\neq i}$ and $L_k^{\alpha_i}$ is completed, $c_k^{lm}(L_k^{\alpha_i}) = min_{0<j\leq n}(L_k^{\alpha_j}) = c_k^{lm}(L_k)$. □

Finally we conclude that:

Theorem 12. *For any state s and any agent α_i, $h_{lmc}(G,s) = h_{lmc}^{D\alpha_i}(G,s)$.*

Proof. From Theorems 5 and 6 the result of distributed computation of $h_{max}^{D\alpha_i}(G,s)$ is equal to the centralized $h_{max}(G,s)$ for any state s, any agent α_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 $J^{D\alpha_i} = (J^{\alpha_i}, \{J^{L\alpha_j}\}_{j\neq i})$ 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 α_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 $L_k^{\alpha_i}$ 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[1] (sokoban with multiple robots as agents), woodworking (the instruments are agents) and elvators, 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 $h_{lmc}^{D\alpha_i}$ 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,

[1]We have created problems specifically for multiple robots.

domain	$h_{lmc}^{\alpha_i}$	$h_{lmc}^{D\alpha_i}$	$\hat{h}_{lmc}$	$\hat{e}_{lmc}$	$\hat{t}_{lmc}$	$\hat{t}_{lmc}^s$
elevators08 (20)	**2**	**2**	0.18	39.9	0.15	0.01
logistics00 (20)	6	**12**	0.26	2521.7	4.78	0.01
zenotravel (18)	6	**10**	0.41	142.4	0.74	0.03
rovers (18)	**6**	**6**	0.52	33.7	0.45	0.15
blocksworld (30)	17	**20**	0.54	45.5	0.56	0.09
satellites (18)	5	**10**	0.55	63.9	0.52	0.03
driverlog (20)	**13**	12	0.8	4.4	0.22	0.12
depot (20)	**7**	4	0.88	1.4	0.13	0.11
woodwork.08 (20)	6	**8**	0.88	12	1.11	0.35
ma-sokoban (10)	**8**	5	1	1	0.15	0.14
total (194)	76	**89**	-	-	-	-

Table 1: Coverage and average of $h_{lmc}^{\alpha_i}/h_{lmc}^{D\alpha_i}$ ratios for initial state heuristic ($\hat{h}_{lmc}$), expanded states ($\hat{e}_{lmc}$), total planning time ($\hat{t}_{lmc}$) and time per expanded state ($\hat{t}_{lmc}^s$).

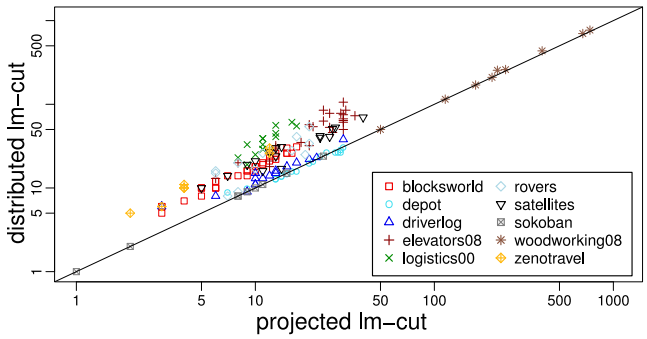


Figure 2: Per-problem $\hat{h}_{lmc} = h_{lmc}^{\alpha_i}/h_{lmc}^{D\alpha_i}$ ratios for initial state.

which means that the projected $h_{lmc}^{\alpha_i}$ has the same information as $h_{lmc}^{D\alpha_i}$ moreover, $h_{lmc}^{D\alpha_i}$ has to handle a lot of projected actions.

As expected, both variants of the h_{max} 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 $h_{max}^{\alpha_i}$ and 0 for $h_{max}^{D\alpha_i}$) and elevators domain (coverage 0 for $h_{max}^{\alpha_i}$ and 2 for $h_{max}^{D\alpha_i}$), the difference between $h_{max}^{\alpha_i}$ and $h_{max}^{D\alpha_i}$ is not significant.

To understand the cause of the behavior of $h_{lmc}^{\alpha_i}$ and $h_{lmc}^{D\alpha_i}$ better, we have extracted the heuristic values computed for the initial state by both heuristics (for $h_{lmc}^{\alpha_i}$ taking average for all agents), computed a ratio $h_{lmc}^{\alpha_i}/h_{lmc}^{D\alpha_i}$ for each problem and averaged the ratios per domain. The results are in Table 1 in the column labeled $\hat{h}_{lmc}$ (computed from all problems for which the init. state heuristic values were obtained). The results 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 $\hat{h}_{lmc}$ 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_{max} in Lemma 2. Against intuition, the answer is no. The reason lies in the inherent variance of the h_{lmc} heuristic depending on the tie-breaking behavior of the precondition choice function (pcf). Although in the proofs of equality of h_{lmc} and $h^{\mathsf{D}\alpha_i}_{lmc}$ it was possible to fix the tie-breaking behavior (thanks to use of distributed h_{max}), it is not the case with $h^{\alpha_i}_{lmc}$. The fact p which maximizes h_{max} in $h^{\alpha_i}_{lmc}$ may not maximize it in $h^{\mathsf{D}\alpha_i}_{lmc}$ (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 $h^{\alpha_i}_{lmc} > h^{\mathsf{D}\alpha_i}_{lmc}$, 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 $\hat{e}_{lmc}$ shows the average ratio of expanded states, restricted to problems solved by both heuristics. The most significant improvement is in the logistics domain, where $h^{\alpha_i}_{lmc}$ expands over $2500\times$ more states than $h^{\mathsf{D}\alpha_i}_{lmc}$, 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 ($\hat{t}_{lmc}$ in Table 1) and time per expanded state ($\hat{t}^s_{lmc}$ 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. $2\times$–$10\times$ faster solution, which was not unexpected. The average time spent on an expanded state shows that the projected heuristic is $10\times$–$100\times$ 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 $h^{\mathsf{D}\alpha_i}_{lmc}$ 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 loading 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 is the same (or even better) than is the global estimate (such as depot or ma-sokoban). Further improvements may include the use of incremental LM-Cut computation (so the distributed version can be used only sparingly) or some other way of combining the projected and distributed estimates, such as multi-heuristic search.

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

Bonet, B., and Geffner, H. 1999. Planning as heuristic search: New results. In *ECP*, 360–372.

Brafman, R. I., and Domshlak, C. 2008. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of ICAPS'08*, 28–35.

Fabre, E., and Jezequel, L. 2009. Distributed optimal planning: an approach by weighted automata calculus. In *Proceedings of the 48th IEEE Conference on Decision and Control, CDC 2009, combined withe the 28th Chinese Control Conference, December 16-18, 2009, Shanghai, China*, 211–216.

Fabre, E.; Jezequel, L.; Haslum, P.; and Thiébaux, S. 2010. Cost-optimal factored planning: Promises and pitfalls. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, 65–72.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In *Proceedings of ICAPS'09*.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of ICAPS'07*, 176–183.

Jezequel, L., and Fabre, E. 2012. A#: A distributed version of a* for factored planning. In *Proceedings of the 51th IEEE Conference on Decision and Control, CDC 2012, December 10-13, 2012, Maui, HI, USA*, 7377–7382.

Maliah, S.; Shani, G.; and Stern, R. 2014. Privacy preserving landmark detection. In *Proceedings of ECAI'14*.

Nissim, R., and Brafman, R. I. 2012. Multi-agent A* for parallel and distributed systems. In *Proceedings of AAMAS'12*, 1265–1266.

Štolba, M., and Komenda, A. 2013. Fast-forward heuristic for multiagent planning. In *Proc. of DMAP Workshop of ICAPS'13*, 75–83.

Štolba, M., and Komenda, A. 2014. Relaxation heuristics for multiagent planning. In *Twenty-Fourth International Conference on Automated Planning and Scheduling*.

Torreño, A.; Onaindia, E.; and Sapena, O. 2014. FMAP: Distributed cooperative multi-agent planning. *Applied Intelligence*.