Viewing Landmarks as Temporally Extended Goals

Letao Wang **Jorge A. Baier** **Sheila A. McIlraith**
Department of Computer Science,
University of Toronto,
Canada

Abstract

Landmarks are facts that must hold true at some point in all solutions to a planning problem. The exploitation of landmarks and associated ordering constraints has been shown to be very effective in computing high-quality plans, quickly. Indeed the success of LAMA, winner of the 2008 International Planning Competition sequential satisficing track, can be attributed in some measure to its ability to generate reliable landmark and ordering information and to exploit them via a pseudo-heuristic. In this paper we view landmarks and their associated orderings as temporally extended goals (TEGs) and adapt an existing heuristic for TEGs to provide guidance to a planner in two ways: by estimating distance to achieving landmarks and goals while respecting orderings, and by identifying dead-ends in the search tree that can be pruned by the planner. We integrate our heuristic as an extension to LAMA. Experiments show that our planner outperforms LAMA in several domains, while exploring significantly less of the search space in over 50% of the experiments run.

Introduction

Domain-independent heuristics are central to the performance of state-of-the-art satisficing planners. Some of the most notable heuristics introduced in the last 10 years include the relaxed plan graph (RPG) heuristic (Hoffmann and Nebel 2001), the additive heuristic (Bonet and Geffner 2001), and the causal graph heuristic (Helmert 2006). These heuristics can provide useful guidance to a forward state-space planner by estimating, sometimes very accurately, the distance from a given state to a goal state.

Another technique that has been successful in enhancing plan generation is the computation of *landmarks* and *orderings*. Landmarks (Hoffmann, Porteous, and Sebastia 2004) are subgoals that need to be achieved in all plans, while orderings are temporal relationships that hold between landmarks. As such, they both provide valuable information that can be exploited by a planner.

Previous methods of incorporating landmarks and orderings into heuristic search planners do not exploit them fully. To find a plan, Hoffmann, Porteous, and Sebastia's approach (2004) calls the FF planner to find plan segments between landmarks, progressively building toward the goal. A major disadvantage of this approach is that individual plan segments are short-sighted. They do not consider what needs to be achieved in the future, which often leads to suboptimal plans. On the other hand, the LAMA planner uses a landmark counting pseudo-heuristic (Richter, Helmert, and Westphal 2008) in parallel with the standard FF heuristic that is computed for the problem's goals. This pseudo-heuristic does not estimate the distance from a state to the satisfaction of all landmarks, but rather counts the number of landmarks that remain to be achieved from that state. While this approach has proven to be effective, it implicitly assumes all landmarks are equally easy to achieve, and thus relies heavily on regularities in the problem's structure.

In this paper, we introduce a novel approach to exploiting landmarks and orderings by transforming them into temporally extended goals (TEGs), combining them with the original goal into a single property to be achieved, and exploiting FF's RPG heuristic. Our heuristic is not short-sighted since it always considers the task in its entirety, and makes no assumptions about the distribution of landmarks. It has the potential to provide a more accurate estimate of the distance from a state to successful satisfaction of the goal. Indeed, experiments show that our planner outperforms LAMA in several domains, while exploring significantly less of the search space in over 50% of the experiments run.

We begin with a review of the necessary background and related work. This is followed by a detailed description of our heuristic and its implementation. Finally, we present experimental results together with an analysis and discussion.

Background

In this section, we review a number of concepts that are employed in our work. We assume readers are familiar with the RPG heuristic in FF, which forms the basis for the heuristic employed in this paper. Details of the RPG heuristic can be found in Hoffmann and Nebel's paper (2001).

Landmarks and orderings

Landmarks by definition are facts that must hold true at some point in all solutions of a given planning task. Porteous and Sebastia (2000) first studied the properties and enumeration of landmarks, and many subsequent papers by these and other authors further elaborated and formalized the concept. In Porteous and Sebastia's work, landmarks and orderings are generated using a regression-based method on the relaxed planning graph. However, soundness cannot be

guaranteed for ordering constraints found with this method due to its relaxation.

Five types of orderings, which describe temporal relationships between pairs of facts, are introduced by Hoffmann, Porteous, and Sebastia (2004). Three of these types are *mandatory* in the sense that they dictate one fact must occur before another in any valid solution plan. Although the definitions allow for arbitrary facts to be ordered, only orderings between landmarks are relevant to this paper. The formal definitions for the orderings follow.

Definition 1 *A fact p is **added** at time i where $i > 0$ iff p is true at time i but false at time $i-1$. A fact p is **first added** at time i where $i > 0$ iff p is added at time i, but is not added at any time j such that $0 < j < i$.*

Definition 2 *There is a **natural ordering** between facts A and B, written $A \rightarrow B$, iff B is not initially true, and in every solution plan, B is true at time j implies A is true at some time $i < j$.*

Definition 3 *There is a **greedy-necessary ordering** between facts A and B, written $A \rightarrow_{gn} B$, iff B is not initially true, and in every solution plan, B is first added at time i implies A is true at time $i-1$.*

Definition 4 *There is a **necessary ordering** between facts A and B, written $A \rightarrow_{n} B$ iff B is not initially true, and in every solution plan and for every $i > 0$, B is added at time i implies A is true at time $i-1$.*

Richter, Helmert, and Westphal (2008) proposed an improved method to generate landmarks and orderings. The main contributions of this method include the incorporation of disjunctive landmarks and the extraction of landmarks by analysis of domain transition graphs. All mandatory orderings generated using this method are sound. Other attempts in the planning literature to generate landmarks and orderings include Zhu and Givan (2003), but will not be discussed further as they are not as relevant to our work.

Exploiting landmarks in heuristic search

As a way to exploit landmarks in planning, Hoffmann, Porteous, and Sebastia (2004) proposed the LM$^{\text{local}}$ procedure which searches iteratively toward the next nearest achievable landmark as specified by their orderings. By decomposing a planning problem into smaller tasks through the use of landmarks as intermediary goals, they were able to solve larger instances compared to a base planner that did not exploit landmarks. However, due to the greedy nature of the search and the presence of unsound orderings, this procedure can sometimes fail to find any solution plan, and often produces plans that are much longer than other competent planners.

Richter, Helmert, and Westphal (2008) proposed a counting algorithm that estimates the number of landmarks remaining to be achieved as a novel pseudo-heuristic. This pseudo-heursitic returns the sum of the number of landmarks that have not been accepted, plus the number of landmarks that are required again. A landmark L is *accepted* iff it was true at some previous state s', and all landmarks ordered before L were accepted in s'. A landmark L is *required again* iff it is accepted in the current state s but is not true in s, and some greedy-necessary successor of L is not accepted in s.

The landmark counting pseudo-heuristic is incorporated into LAMA, a heuristic search planner developed on the Fast Downward framework (Helmert 2006). It uses a combination of FF's RPG heuristic and helpful actions, in addition to the landmark counting pseudo-heuristic and preferred operators. Search is done on a best-first basis, with iterations of weighted A* to improve plan quality after the first plan is found. LAMA has demonstrated significantly better performance over comparable state-of-the-art planners, winning the sequential satisficing track of the sixth international planning competition (IPC 2008).

Unlike other planners that typically express planning tasks in the STRIPS or ADL formalisms, LAMA uses a SAS$^+$ representation. One particular difference between SAS$^+$ and the other formalisms is the fact that state variables in SAS$^+$ (analogous to grounded predicates in STRIPS or ADL) are not restricted to a binary domain. Propositions in the SAS$^+$ formalism have the form $x = v$, where x is a state variable, and v is a value in x's domain. Exactly one proposition is true for each variable at any time, i.e. variables always have a unique value assignment in any state.

Temporally extended goals and FSA

Temporally extended goals describe properties that must be achieved along a sequence of states visited during the execution of a plan, in contrast to classical goals which only specify properties of a final state and ignore the path taken to reach it. Statements such as "deliver all priority packages *before* delivering regular mail", "*always* have paper in the printer", and "*never* go swimming *right after* eating" are examples of TEGs in real life.

In the planning community, TEGs are typically expressed in linear temporal logic (LTL) (e.g. TLPlan, Bacchus and Kabanza 1998). LTL is a boolean logic with the addition of temporal operators. The formula $\Diamond p$ (*eventually* p) denotes the proposition that p is true at some state, while $\Box p$ (*always* p) denotes p is true in all states. Formula $\bigcirc p$ (*next* p) expresses p is true in the next state. Finally, $p \cup q$ (p *until* q) asserts p is true in all states prior to the achievement of q.

There is a known relationship between LTL and finite-state automata which has been exploited in planners for tasks involving temporally extended goals. For example, Baier and McIlraith (2006) proposed a method for transforming LTL formulae into nondeterministic parametrized FSA. Such FSA accept a sequence of states σ iff σ satisfies the corresponding LTL formula. Furthermore, they show that these FSA can be modeled within the planning problem, allowing, in particular, the accepting condition of the FSA to be represented as a condition on regular planning predicates. With this method, they reduce the satisfaction of a TEG to the achievement of a simple final-state goal. The translation allows advances in classical heuristic-search planning to be leveraged for planning with TEGs.

FSA Formulation of Orderings

The main idea behind our approach is to treat landmarks and their orderings as TEGs. Starting with the set of mandatory

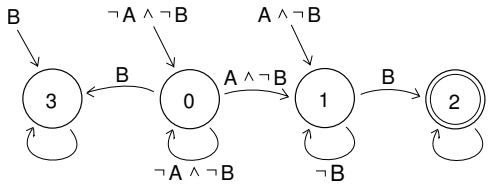

Figure 1: FSA template for natural ordering

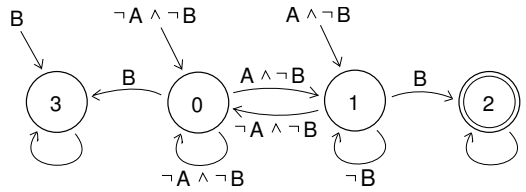

Figure 2: FSA template for greedy-necessary ordering

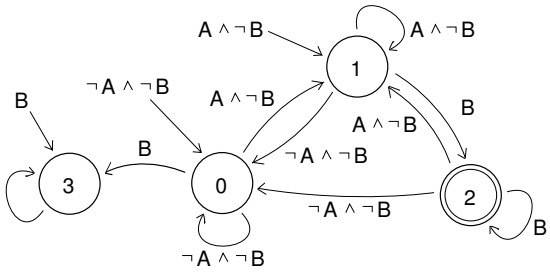

Figure 3: FSA template for necessary ordering

landmarks and orderings obtained from LAMA, we use a preprocessing procedure to eliminate ones that are redundant to our heuristic. One category of redundant orderings has the form $A \rightarrow B$, where A is of the form $x = v$, B is of the form $x = \overline{v}$, and x is a binary variable. Other orderings that are redundant for our heuristic, but are not yet handled by our implementation, include ones that are implied by transitivity ($A \rightarrow C$ when $A \rightarrow B$ and $B \rightarrow C$ are known), and those that are sound even in the relaxed planning task (for example, $A \rightarrow B$ where A is a precondition of all operators that achieve B). Non-mandatory types of orderings, namely reasonable and obedient-reasonable, are also disregarded since their soundness cannot be guaranteed.

Each ordering that we consider is treated as a TEG, from which we generate one corresponding finite-state automaton. Since the structure of each automaton depends solely on the type of its associated ordering, templates obtained using Baier and McIlraith's translation method (2006) are used for their generation. These templates were further simplified so that each automaton is in exactly one state at any time, like a deterministic FSA. The simplification is advantageous in the SAS$^+$ formalism of LAMA since a single state variable is now sufficient to represent the state of each automaton. Memory overhead is thus decreased significantly compared to the alternative of creating a new predicate for each state of each unsimplified automaton.

Proposition 1 *If there is a natural ordering between landmarks A and B, then the sequence of states visited by any valid plan satisfies the LTL formula:*

$$\Diamond B \rightarrow \neg B \,\mathsf{U}(A \wedge \neg B). \tag{1}$$

The FSA template for a natural ordering is given in Fig. 1. Arrows that are not attached to any source node determine the initial state of the automaton. Transitions without labels are unconditional by default. In this template, nodes 0, 1, and 2 respectively represent states where neither A nor B has been achieved, only A has been achieved, and both A and B have been achieved. State 3 is the *sink state* indicating the ordering has been violated.

In addition to satisfying the ordering constraint itself, the automaton also ensures both landmarks A and B are eventually achieved by marking state 2 as an *accepting state*.

Proposition 2 *The TEG encoded in Fig. 1 is the conjunction of Eq. 1 and $\Diamond A \wedge \Diamond B$.*

Proposition 3 *If there is a greedy-necessary ordering between landmarks A and B, then the sequence of states visited by any valid plan satisfies the LTL formula:*

$$\neg B \,\mathsf{U}\, B \rightarrow \neg B \,\mathsf{U}(A \wedge \neg B \wedge \bigcirc B). \tag{2}$$

The template for a greedy-necessary ordering is given in Fig. 2. It contains all the nodes and transitions of automata for natural orderings, since every greedy-necessary ordering is also a natural ordering, i.e. natural orderings are more general. In addition, an added transition arc from state 1 to state 0 handles the case where A is added and deleted before B is first achieved.

Proposition 4 *The TEG encoded in Fig. 2 is the conjunction of Eq. 2 and $\Diamond A \wedge \Diamond B$.*

Proposition 5 *If there is a necessary ordering between landmarks A and B, then the sequence of states visited by any valid plan satisfies the LTL formula:*

$$\Box(\neg B \wedge \bigcirc B \rightarrow A). \tag{3}$$

The template of a necessary ordering is given in Fig. 3. It contains all nodes and transitions of an automaton for a greedy-necessary ordering, as well as two additional transitions from state 2. These new transitions allow the automaton to reset to previous states in order to monitor the interaction between A and B for all time.

Proposition 6 *The TEG encoded in Fig. 3 is the conjunction of Eq. 3 and $\Diamond A \wedge \Diamond B$.*

Heuristic Search with Landmark FSA

There are two ways to use the FSA described above to help guide heuristic search. First, the relaxed plan graph in the FF heuristic can be augmented to include the structure of the automata. Second, the search tree can be pruned to eliminate actions that violate ordering constraints.

Augmented relaxed plan graph

Finite-state automata for landmark orderings are introduced into the expansion of the relaxed plan graph using a method similar to that of Baier, Bacchus, and McIlraith (2009) for

temporally extended constraints. Specifically, the following three modifications are made to the relaxed planning task.

First, for each automaton x, new predicates of the form `fsa_x_state` and `fsa_x_accepted` are added to the list of state variables. The former represents the current state of the automaton, having a domain of all possible automaton states, namely $\{0, 1, 2, 3\}$. The latter is a binary variable that indicates whether the automaton has visited the accepting state, namely state 2.

Second, the initial state of each automaton is added to the list of initial propositions of the RPG. For each automaton x, propositions of the form `fsa_x_accepted = w` are also added, where w is 1 if the automaton has visited its accepting state, and 0 otherwise. Propositions of the form `fsa_x_accepted = 1` for every automaton x are added to the list of goal propositions.

Finally, a new `fsa_update` operator with zero cost and no preconditions is created. This operator specifies transitions of all automata by a list of conditional effects. A transition of automaton x from state y to state z with condition p is encoded as an effect of the form

```
(when (and (p) (fsa_x_state = y))
      (fsa_x_state = z))
```

The operator also specifies the accepting condition of each automaton x by an effect of the form

```
(when (fsa_x_state = 2)
      (fsa_x_accepted = 1))
```

An augmented relaxed plan graph is built from this set of propositions and operators. Just as all FSA state variables should be updated once after the application of each operator, we enforce one application of the `fsa_update` operator for every layer of the RPG by evaluating domain operators and axioms in alternation with the update operator. Each layer of the augmented RPG is first expanded with domain operators and axioms as the FF heuristic normally does, then the `fsa_update` operator is evaluated on the result to further expand the graph.

Intuitively, relaxed plans that do not consider landmarks and orderings may be unrealistic ones that violate many orderings and cause heuristic estimates to become inaccurate. The augmentations above attempt to address this issue by encouraging the RPG to achieve landmarks and obey orderings during its computation.

Search tree expansion and pruning

Our planner preserves the architecture of LAMA, which for a single heuristic, expands nodes on a best-first basis. Similar to the relaxed planning task described in the previous section, each node in the search tree also contains a collection of state variables of the form `fsa_x_state` and `fsa_x_accepted`. The initial state begins with all FSA variables initialized according to the status of their associated landmarks. Whenever a new node is created, an application of the `fsa_update` operator assigns new values to all FSA state variables based on their previous value, transition arcs of the automata, and current values of state variables and derived predicates.

By definition, any sequence of actions that violates an ordering of landmarks cannot be part of a valid plan. In the FSA representation, a violation of an ordering constraint is marked by an automaton entering state 3. Thus during expansion of the search space, any node that is found to exhibit such a violation can be discarded without further consideration. This pruning technique is not currently applicable to the set of ordered landmarks found by LAMA due to the fact that all orderings extracted by LAMA are necessarily obeyed by *any* action sequence. Nevertheless, this pruning technique can be valuable for more novel orderings and for other forms of control knowledge.

Experiments and Results

We implemented a new planner based on LAMA that incorporates our heuristic. Our planner uses a combination of the augmented RPG heuristic described above and the landmark counting pseudo-heuristic found in LAMA. As with LAMA, preferred operators for the landmark counting pseudo-heuristic are also enabled in our planner. The same multi-heuristic search strategy as LAMA is used, selecting nodes to be expanded from two separate open lists, one for each heuristic. The ability to prune the search tree by detecting violations of orderings was omitted since it does not have any effect with the landmarks and orderings currently being exploited.

The performance of our planner was compared against LAMA on deterministic, sequential, nontemporal, and non-numeric domains from the last three International Planning Competitions. When multiple suitable formulations are given for the same domain, the one whose PDDL requirements are most advanced was chosen. All experiments were conducted on a single core of AMD Opteron 2220 processor running at 2.8 GHz, with time and memory limited to 20 minutes and 4 GB respectively for each task. Comparisons are made for two intervals of the iterative search process: the first search iteration alone where a solution plan is first produced, and the entire process including all search iterations until the planner terminates.

Table 1 shows a summary of results with domains divided into three sections. The top section lists domains where our planner performed significantly better than LAMA, while the bottom section lists the domains in which the opposite occurs. A planner is considered to perform significantly better if it solves at least as many problems, and finds plans of equal or shorter length while exploring no more nodes than its competitor. Domains in which the comparison is inconclusive due to a tradeoff between number of problems solved, plan quality, and nodes expanded, are placed in the middle section of the table.

Although there are a number of domains in which LAMA solved more instances than our planner, we were able to outperform LAMA in terms of number of problems solved in 4 domains as well, most notably in the Optical Telegraph domain. The results also give a strong indication that search with our planner tends to be more informed and leads more directly to a solution. For tasks in 7 out of 21 domains, our planner only needed to explore, on average, fewer than half

Domain (Problems)	PS	First iteration %IPL	First iteration %INE	First iteration %IST	Until termination %IPL	Until termination %INE
Optical telegraph (IPC4, 48)	48/3	0.0 (0/0)	99.8 (3/0)	99.8 (3/0)	0.0 (0/0)	99.8 (3/0)
Pipesworld-notankage (IPC4, 50)	20/20	0.2 (7/9)	32.2 (11/7)	24.4 (9/7)	1.6 (1/0)	-2.0 (7/12)
Rovers (IPC5, 40)	40/40	1.9 (20/8)	77.9 (38/1)	71.4 (30/0)	0.6 (15/13)	1.1 (20/19)
Scanalyzer (IPC6, 30)	29/28	12.2 (22/2)	78.4 (27/1)	69.4 (21/2)	6.6 (17/2)	86.0 (23/5)
Storage (IPC5, 30)	19/19	5.2 (5/1)	35.6 (9/3)	48.8 (9/4)	0.8 (3/0)	11.4 (10/4)
Transport (IPC6, 30)	30/30	3.3 (16/11)	57.3 (26/2)	30.7 (19/5)	4.0 (15/10)	39.8 (14/15)
Trucks (IPC5, 30)	12/12	-0.9 (4/5)	20.6 (5/6)	4.7 (4/6)	0.7 (2/0)	27.0 (6/5)
Airport (IPC4, 50)	31/36	4.5 (13/1)	-7.3 (16/14)	-83.3 (326)	1.2 (6/1)	34.7 (19/11)
Elevators (IPC6, 30)	30/25	-4.7 (12/13)	58.1 (21/4)	12.4 (14/7)	-4.1 (7/15)	-60.4 (9/16)
Openstacks (IPC5, 30)	28/29	0.0 (0/0)	0.0 (0/0)	-71.5 (0/25)	0.0 (0/0)	10.6 (7/0)
Parcprinter (IPC6, 30)	9/17	3.0 (2/0)	12.9 (4/3)	-86.7 (0/9)	0.0 (0/0)	27.5 (5/2)
Satellite (IPC4, 36)	33/32	-6.0 (2/19)	37.3 (24/7)	15.2 (15/9)	-3.0 (6/16)	-19.2 (18/12)
TPP (IPC5, 30)	30/30	-25.5 (0/26)	67.0 (24/2)	39.2 (21/1)	-24.7 (0/24)	63.3 (17/9)
Cyber security (IPC6, 30)	30/30	0.1 (2/0)	-10.5 (13/17)	-68.2 (0/29)	-7.5 (2/1)	-31.7 (14/16)
Openstacks (IPC6, 30)	30/30	0.0 (0/0)	-0.0 (0/1)	-58.1 (2/26)	-1.3 (0/6)	18.6 (22/4)
Pathways (IPC5, 30)	28/29	-3.3 (5/14)	70.2 (24/3)	49.8 (20/2)	-2.0 (4/15)	-68.8 (8/19)
Pegsol (IPC6, 30)	29/30	1.5 (11/8)	-59.8 (8/21)	-57.3 (7/18)	-0.2 (0/1)	-26.7 (8/21)
Pipesworld (IPC5, 50)	26/26	-12.6 (7/12)	-65.9 (8/16)	-83.3 (3/19)	-8.7 (4/8)	-34.2 (11/13)
PSR-large (IPC4, 50)	30/31	-10.3 (5/19)	-28.0 (9/20)	-34.9 (6/22)	-2.6 (1/6)	-61.3 (6/23)
Sokoban (IPC6, 30)	22/24	-6.2 (6/10)	17.1 (11/11)	-38.2 (4/18)	-0.9 (0/2)	4.1 (11/11)
Woodworking (IPC6, 30)	30/30	1.5 (17/9)	-40.8 (13/15)	-60.0 (4/22)	-1.5 (11/13)	-48.4 (10/19)
All domains	584/551	-2.2 (156/167)	30.2 (294/154)	-19.4 (194/257)	-2.5 (94/133)	2.1 (248/236)

Table 1: Comparative performance of our planner relative to LAMA. PS is the number of **p**roblems **s**olved, where an entry of the form "x/y" means our planner solved x instances whereas LAMA solved y instances. %IPL is the average percentage of **i**mprovement (decrease) in **p**lan **l**ength achieved by our planner over LAMA. %INE is the average **i**mprovement in number of **n**odes **e**xpanded during search. %IST is the average **i**mprovement in **s**earch **t**ime. In the last five columns, an entry of the form "x (y/z)" means our planner achieved better results in y instances and worse results in z instances (and identical results in the remaining ones), with x% being the average difference. Numbers reported in the last five columns include only instances solved by both planners. All average percentage differences are derived from the geometric mean of ratios.

the number of nodes LAMA required to find a solution. The converse is only true for 2 domains.

In general, the percentage decrease in search time remains lower than the percentage decrease in number of nodes expanded. This behaviour is explained by the overhead incurred by our heuristic in expanding the augmented RPG with an increased number of propositions and operator effects. As a consequence of this overhead, savings in search effort sometimes do not pay off in terms of number of problems solved. In some cases, the overhead seems to fully explain why fewer instances were solved by our planner, as a more informative heuristic did not translate into more solved instances (e.g. Parcprinter). Overhead may also explain why our planner has better results in terms of plan quality for the first iteration than for the full iterative search (e.g. Cyber security). As our planner spends more time evaluating its heuristic for each node, the LAMA planner can explore many more nodes in the given 20 minutes.

Improvements in plan length and nodes expanded appear positively correlated with only a few exceptions, which suggests the difference between the two planners truly lies in the informativeness of their heuristics, rather than a simple tradeoff between plan quality and search effort. A planner using a more informed heuristic should be able to find better quality plans while searching fewer nodes, and in this respect our planner shows increased informedness in 7 out of 21 domains on the first iteration, and in 9 domains after full iterative search despite its overhead. In comparison, LAMA only exhibited similar advantages in 3 domains on the first iteration and 8 domains after full iterative search.

The ability of each planner to solve tasks within limited node expansions can be seen in Fig. 4. Since planners use the vast majority of their memory to store nodes expanded in the search tree, the graph also reflects the ability of each planner to solve tasks given limited memory resources. A clear decrease can be seen in the number of nodes that our planner needs to expand in order to solve the set of test problems. For example, our planner solved the easiest 300 instances using only a maximum of 270 nodes per instance, whereas LAMA needed nearly 630 nodes to do the same. Alternatively, if the memory limit was sufficient to only store 1000 nodes in the search tree, then our planner would have been able to solve 385 problems whereas LAMA would only have been able to solve 333. Once again, this justifies the claim that our planner using the augmented heuristic is more informed and is capable of finding a solution without having to explore nearly as much of the search space.

To gain a better understanding of the reasons behind improvements in performance seen above, our heuristic was measured individually against the RPG heuristic and the

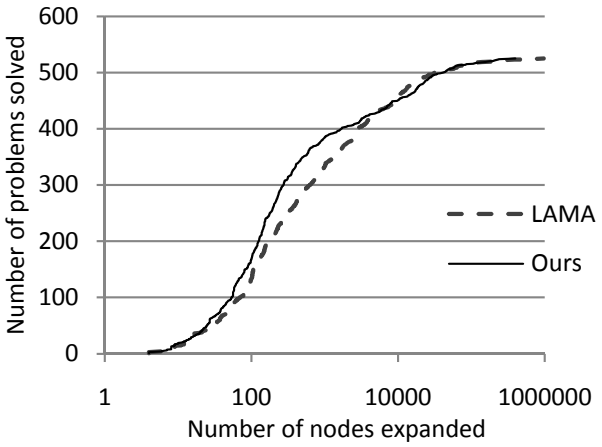

Figure 4: Number of problems each planner can solve as a function of limit on number of nodes expanded. Only instances solved by both planners are included.

landmark counting pseudo-heuristic using a single-heuristic, simple best-first search configuration as the base planner. In order to be consistent with experiments above, implementations of the latter two heuristics were taken directly from LAMA. Table 2 shows a summary of results.

Computation time appears to be the biggest drawback of our heuristic when it is used alone in this simplistic search scheme. Even though our heuristic had the benefit of searching fewer nodes in a number of domains, other heuristics often found their solutions more than 50% faster. This demonstrates the magnitude of time overhead in our heuristic. It is the primary reason that our heuristic could not solve as many problems as the RPG heuristic. On the other hand, results for multi-heuristic planners shown above suggest that the overhead can be mitigated by cooperating with the fast landmark counting heuristic. Further optimizations of our implementation can also lead to improvements in search time. It is also worth mentioning that state-of-the-art planners rarely use single heuristics in plain best-first search as done here. The perceived loss in number of problems solved here does not necessarily translate into an equal loss when our heuristic is integrated into a state-of-the-art planning algorithm such as that of LAMA.

Since our heuristic incorporates the guidance of both FF's RPG and ordered landmarks, it is reasonable to expect it to be better informed than either one of those alone. Indeed, against the FF heuristic, our heuristic was able to find solutions while exploring fewer nodes in 8 domains, while the opposite is true for only 2. The improvement is even more dramatic against the landmark counting pseudo-heuristic, where the number of nodes expanded is reduced by more than 50% in over half of the domains. The quality of solution plans was also affected, with mixed results, but differences are much smaller.

It may seem strange that our multi-heuristic planner achieved different results from LAMA even though on a best-search basis, our heuristic was no different from the RPG heuristic in many domains in terms of solution plan length and number of nodes searched. To explain this behaviour, note that the actual search tree and heuristic values do not need to be identical for the two heuristics when the number of nodes expanded is equal. These differences that are not visible in the aggregated results presented can lead to varying behaviours when a different search strategy is used.

In summary, the main observations that can be made here are as follows. Our heuristic alone can provide somewhat improved guidance to a simple best-first search planner, but its evaluation can be very time-consuming. On the other hand, the landmark counting pseudo-heuristic is very fast to evaluate, but when used alone it rarely provides enough guidance to produce good plans. However, when the two are employed together under a multi-heuristic planner framework such as the one for LAMA, the two heuristics can in fact complement each other, allowing the overall planner to exploit strengths of each while mitigating their weaknesses.

Discussion and Future Work

In this paper, we proposed a method for exploiting landmarks and orderings by transforming them into TEGs, combining them with the original problem goal, and exploiting FF's RPG heuristic on the resulting planning problem. To this end, we encode the TEGs as finite-state automata and incorporate the automata into the computation of the RPG heuristic. The automata not only characterize progress towards satisfaction of TEGs, but they can also identify dead ends for a planner to prune. The resulting heuristic was implemented in a planner based on LAMA and compared against LAMA on tasks from recent International Planning Competitions.

Experimental results show that the two planners display comparable general performance but with performance differing on individual domains. The two planners solved a comparable number of problems, with a few exceptions. In the Optical Telegraph domain from IPC4, our planner solved 45 more problems, whereas in the Parcprinter domain from IPC4, LAMA solved 8 more problems. Plan lengths were also comparable. Each planner produced first solutions with superior length in approximately 30% of solved instances. Plan lengths were identical in the remaining 40%. The main advantage of using our heuristic in the planner proved to be its informativeness. In the first iteration of planning, our planner expanded fewer nodes in almost 2/3 of the cases. The result, a 30% average reduction in nodes searched, suggests that our planner was initially far more informed. Measured to termination, the two planners' different search techniques seemed to excel on different domains, each expanding significantly fewer nodes in about 1/2 of the cases.

Despite some computational overhead and a tradeoff in plan length in certain domains, our approach appears very promising. Based on observations made from experimental results, our current planner can already provide significant advantages in applications where memory resources are limited. With some future work focusing on optimizing the computation of our heuristic, including the preprocessing step where all orderings that are redundant for our heuristic

Domain (Problems)	RPG+FSA vs. RPG				RPG+FSA vs. landmark counting			
	PS	%IPL	%INE	%IST	PS	%IPL	%INE	%IST
Airport (IPC4, 50)	23/28	0.0 (0/0)	0.0 (1/0)	-88.6 (0/23)	23/16	-0.6 (3/3)	66.2 (10/3)	-76.2 (2/10)
Optical telegraph (IPC4, 48)	3/3	0.0 (0/0)	0.0 (0/0)	-2.9 (1/1)	3/4	0.0 (0/0)	16.6 (1/2)	-45.0 (1/2)
Parcprinter (IPC6, 30)	8/10	0.0 (0/0)	8.6 (2/0)	-84.6 (0/8)	8/7	4.2 (2/0)	90.4 (4/1)	-99.7 (0/7)
Openstacks (IPC5, 30)	26/28	0.0 (0/0)	0.0 (0/0)	-82.1 (0/26)	26/30	-2.2 (0/20)	62.8 (26/0)	-87.7 (0/25)
Satellite (IPC4, 36)	19/22	-7.4 (1/8)	-16.3 (5/5)	-35.0 (6/9)	19/9	10.5 (4/3)	90.9 (8/0)	89.3 (6/1)
Elevators (IPC6, 30)	12/13	-3.7 (3/8)	28.2 (5/7)	-19.7 (5/7)	12/4	-40.3 (0/4)	34.0 (2/2)	-64.8 (1/3)
Pipesworld (IPC5, 50)	13/15	1.7 (2/1)	2.1 (2/1)	-30.2 (2/9)	13/19	-0.7 (5/4)	1.3 (2/10)	-95.4 (1/12)
Transport (IPC6, 30)	10/12	-7.0 (3/5)	55.9 (7/1)	-35.1 (4/6)	10/26	-9.7 (2/5)	21.5 (7/3)	-87.5 (0/9)
PSR-large (IPC4, 50)	13/13	0.0 (0/0)	0.0 (0/0)	-13.2 (5/6)	13/31	31.4 (12/1)	-69.8 (5/8)	-66.3 (4/7)
Openstacks (IPC6, 30)	30/30	0.0 (0/0)	0.0 (0/0)	-74.0 (0/25)	30/30	-9.5 (0/30)	-81.9 (0/30)	-96.4 (0/25)
Sokoban (IPC6, 30)	23/24	-1.2 (1/2)	2.3 (4/4)	-53.4 (1/20)	23/9	-11.1 (0/7)	89.1 (8/1)	48.4 (8/1)
Cyber security (IPC6, 30)	18/18	0.0 (0/0)	0.0 (0/0)	-18.4 (0/18)	18/3	0.0 (0/0)	89.3 (2/1)	49.0 (1/2)
Scanalyzer (IPC6, 30)	19/19	0.0 (0/0)	0.0 (0/0)	-27.1 (5/12)	19/29	3.1 (6/8)	-67.4 (2/17)	-97.4 (0/19)
Rovers (IPC5, 40)	22/23	0.0 (0/0)	0.0 (0/0)	-29.1 (1/15)	22/17	1.3 (8/5)	87.5 (16/0)	85.0 (11/0)
Woodworking (IPC6, 30)	14/14	0.0 (0/0)	0.0 (0/0)	-52.4 (0/12)	14/5	3.2 (1/2)	84.5 (1/3)	70.0 (1/1)
Storage (IPC5, 30)	18/18	0.9 (1/0)	2.9 (2/0)	-4.5 (3/10)	18/17	11.4 (8/1)	34.5 (9/5)	-17.8 (5/5)
TPP (IPC5, 30)	17/19	0.0 (0/0)	0.0 (0/0)	-19.3 (0/11)	17/16	17.8 (12/0)	60.0 (11/1)	29.2 (7/5)
Pipesworld-notank (IPC4, 50)	19/19	-1.7 (1/3)	2.6 (4/1)	-13.9 (3/12)	19/19	2.9 (8/4)	34.1 (12/4)	-59.4 (5/8)
Pegsol (IPC6, 30)	30/30	-0.3 (0/1)	-0.4 (0/1)	-57.0 (3/22)	30/29	5.4 (10/6)	9.8 (17/12)	-15.5 (9/12)
Trucks (IPC5, 30)	12/13	0.0 (0/0)	0.0 (0/0)	-39.7 (0/8)	12/6	5.1 (5/0)	96.7 (6/0)	96.0 (6/0)
Pathways (IPC5, 30)	10/10	0.0 (0/0)	0.0 (0/0)	23.0 (5/3)	10/4	0.0 (0/0)	96.9 (4/0)	98.3 (4/0)
All domains	359/381	-0.8(12/28)	3.1(32/20)	-50.6(44/263)	359/330	2.5(86/103)	38.7(153/103)	-64.5(72/154)

Table 2: Comparative performance of three individual heuristics in best-first search. "RPG+FSA" represents our new heuristic, while "RPG" and "landmark counting" represent the two heuristics taken from LAMA. PS is the number of **p**roblems **s**olved by each respective heuristic. %IPL is the average percentage of **i**mprovement (decrease) in **p**lan **l**ength achieved by a heuristic over its competitor. %INE is the average **i**mprovement in number of **n**odes **e**xpanded during search. %IST is the average **i**mprovement in **s**earch **t**ime. In all columns that report percentage improvement, an entry of the form "x (y/z)" means our heuristic achieved better results in y instances and worse results in z instances (and identical results in the remaining ones), with x% being the average difference. Other than the PS columns, all numbers reported include only instances solved by both heuristics. All average percentage differences are derived from the geometric mean of ratios.

are removed, the overall performance of our heuristic can be improved even further.

We have also looked for possible relationships between the number of landmarks and orderings found for a given task and the degree of improvement obtained by our heuristic. However, the results are not consistent across domains and tasks, and little correlation seems to exist at a global level. Some tasks boasting thousands of landmarks can lead our planner to produce much worse results, whereas just a few landmarks for other tasks may lead to drastic improvements. This suggests that not all landmarks and orderings contribute equally to our heuristic. As part of future work, the ability to determine the helpfulness of landmarks and orderings may play an integral role in improving our heuristic's performance and in reducing overhead.

The conversion of orderings into FSA is another area worthy of further investigation. TEGs generated by our method may be represented more compactly, for example, by merging orderings with transitive relations into a single automaton. This should be feasible since the same method of encoding TEGs into FSA applies equally well to more complex, possibly nested, LTL formulae. Merging orderings into a single automaton also opens the door for alternative heuristics that compute the distance to goal based on the structure of the automaton itself, without the need to relax the original planning task.

Finally, our approach may be extended beyond landmarks. While landmarks and associated orderings reflect properties of *all* plans, there exist other properties that are perhaps only found in a subset of high-quality plans. As part of future work, we would like to devise a method to detect and exploit these *soft* landmarks and orderings, possibly by treating them as hard landmarks or by treating them a temporally extended preferences following, for example, Baier, Bacchus, and McIlraith (2009).

Acknowledgements

The authors gratefully acknowledge funding from the Natural Sciences and Engineering Research Council of Canada (NSERC) and from the Ontario Ministry of Innovation Early Researcher Award (ERA). We would also like to thank the anonymous referees for useful feedback on an earlier draft of this paper.

References

Bacchus, F., and Kabanza, F. 1998. Planning for temporally extended goals. *Annals of Mathematics and Artificial Intelligence* 22(1-2):5–27.

Baier, J. A., and McIlraith, S. A. 2006. Planning with first-order temporally extended goals using heuristic search. In *Proc. 21st National Conference on Artificial Intelligence (AAAI)*, 788–795.

Baier, J. A.; Bacchus, F.; and McIlraith, S. A. 2009. A heuristic search approach to planning with temporally extended preferences. *Artificial Intelligence* 173(5-6):593–618.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1-2):5–33.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 14:253–302.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered landmarks in planning. *Journal of Artificial Intelligence Research* 22:215–278.

Porteous, J., and Sebastia, L. 2000. Extracting and ordering landmarks for planning. In *Proc. of the UK Planning and Scheduling SIG Workshop*.

Richter, S.; Helmert, M.; and Westphal, M. 2008. Landmarks revisited. In *Proc. 23rd AAAI Conference on Artificial Intelligence (AAAI)*, 975–982.

Zhu, L., and Givan, R. 2003. Landmark extraction via planning graph propagation. In *ICAPS Doctoral Consortium*.