Approximation Properties of Planning Benchmarks

Malte Helmert and **Robert Mattmüller** and **Gabi Röger**.[1]

Abstract. For many classical planning domains, the computational complexity of non-optimal and optimal planning is known. However, little is known about the area in between the two extremes of finding *some* plan and finding *optimal* plans. In this contribution, we provide a complete classification of the propositional domains from the first four International Planning Competitions with respect to the approximation classes **PO**, **PTAS**, **APX**, **poly-APX**, and **NPO**.

1 INTRODUCTION

Considering the important role that benchmark domains such as LOGISTICS and SATELLITE play in evaluating the performance of classical planning algorithms, comparatively little is known about their computational properties. With the notable exception of the BLOCKSWORLD domain [4, 12, 13], the published results on the computational complexity of planning benchmarks are rather coarse-grained. In most cases, they are limited to the analysis of two decision problems (relative to a certain planning domain):

- *Plan existence*: Is a given planning task solvable?
- *Bounded plan existence*: Is a given planning task solvable using no more than a certain number of actions?

For many planning domains, it turns out that the former problem is solvable in polynomial time, while the latter is **NP**-complete [5, 6]. In practice, this means that finding *some* plan for such a planning task is easy, while finding an *optimal* plan is difficult. In such a situation, it is natural to ask just how close to optimality we can get without sacrificing polynomial run-time.

This question has been raised and (partially) answered for many classical optimization problems such as the traveling salesperson problem, set covering problems, satisfiability-type problems, and countless others; it is the topic of the area of *approximation algorithms* [1]. In this contribution, we investigate the propositional planning domains introduced in the first four International Planning Competitions [2, 7, 9, 10] from this perspective, providing a complete classification with respect to the standard approximation classes.

We start our investigation by providing some background on the theory of approximation algorithms and how it applies to classical planning domains. We then present our main results in two parts, first discussing planning domains for which good, but not arbitrarily good, approximation algorithms exist, then planning domains for which good approximation algorithms do not exist. Finally, we summarize and conclude.

2 APPROXIMATION ALGORITHMS

Due to limited space, we keep our discussion of the theory of approximation algorithms short, pointing to the literature for formal detail [1]. An *optimization problem* is either a *maximization problem* or a *minimization problem*; we focus on the latter exclusively. A minimization problem is given by a set of *instances*, a set of *feasible solutions* for each instance (possibly empty), and a *measure function* associating a number with each feasible solution of each instance. A solution is called *optimal* for a given instance if it minimizes the measure function among all feasible solution for that instance. An *approximation algorithm* for a minimization problem is a polynomial-time algorithm that, given an instance of the problem, generates a feasible solution for that instance or detects that none exists. An approximation algorithm that only generates solutions with measure at most c times the optimal measure (for some $c \in \mathbb{R}$) is called *c-approximating*. If such an algorithm exists, the problem is called *c-approximable*.

For the purposes of this paper, *planning domains* are minimization problems; their instances are called *planning tasks*. Solutions to planning tasks are sequences of *actions* called *plans*. The only measure function we consider is *sequential plan length*, which associates each plan with the number of actions it contains. Other common measure functions include parallel plan length, or weighted sequential plan length where some actions are more costly than others.

Optimization problems are grouped into different *approximation classes* in the same way that decision problems are classified into decision complexity classes like **P** and **NP**. Translating the usual definitions to the terminology of planning domains and tasks, the most important approximation classes are the following ones:

- Domains where optimal plans can be generated in polynomial time (**PO**).
- Domains which are c-approximable for *all* $c > 1$ (**PTAS**).
- Domains which are c-approximable for *some* $c > 1$ (**APX**).
- Domains where plans of length polynomially bounded by the optimal plan length can be generated in polynomial time (**poly-APX**).
- Domains where optimal plans can be generated in polynomial time by a non-deterministic Turing Machine (**NPO**).

It is easy to see that **PO** $\subseteq$ **PTAS** $\subseteq$ **APX** $\subseteq$ **poly-APX** $\subseteq$ **NPO** and that all these classes are identical if **P** $=$ **NP**. More interestingly, all these inclusions are strict if **P** $\neq$ **NP** [1].

Membership of a minimization problem (planning domain) in one of these approximation classes can be shown constructively by providing a suitable approximation algorithm. Non-membership can be shown by *approximation-preserving reductions* (AP-reductions) from problems that are known to be hard, similar to the way many-one reductions are employed in classical complexity theory. Due to space restrictions, we do not present reductions in full formal details, so that we do not need to formally introduce AP-reductions.[2]

[1] Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Germany. Email: {helmert,mattmuel,roeger}@informatik.uni-freiburg.de

[2] Full proofs are available from the authors upon request.

3 APPROXIMATION AND PLANNING

There is little work in the literature on the approximation properties of planning domains, the main exception being Selman's paper which links approximability to *reactive planning* and shows that the BLOCKSWORLD domains is 2-approximable and thus in **APX**, but not in **PTAS** unless **P** = **NP** [12]. However, the literature does contain classifications of these planning domains from a classical complexity theory point of view [5, 6], addressing the *plan existence* and *bounded plan existence* decision problems. We can benefit from this body of work in the following ways:

- A domain belongs to **NPO** if shortest plan lengths are polynomially bounded by task size. Otherwise, it is not in **NPO**, as it is not always possible to *write down* a plan in polynomial time.
- Domains with polynomial-time planning algorithms belong to **poly-APX**, and to **PO** if the algorithms generate optimal plans.
- Domains with **NP**-hard *plan existence* problems do not belong to **poly-APX** unless **P** = **NP**.
- Domains with **NP**-hard *bounded plan existence* problems do not belong to **PO** unless **P** = **NP**.

Applying these rules to the complexity results for the benchmark domains of the four International Planning Competitions [5, 6], there are ten planning domains for which the classification into the approximation classes introduced in the previous section is not implied by the decision complexity results, namely BLOCKSWORLD, DEPOT, DRIVERLOG, GRID, LOGISTICS, MICONIC-SIMPLEADL, MICONIC-STRIPS, ROVERS, SATELLITE, and ZENOTRAVEL. In all these domains, polynomial non-optimal planning algorithms exist, but bounded plan existence is an **NP**-complete problem. Therefore, all these domains belong to **poly-APX**, but none belongs to **PO** unless **P** = **NP**. The open question, then, is whether these domains belong to **PTAS**, to **APX** $\setminus$ **PTAS**, or to **poly-APX** $\setminus$ **APX**.

As noted above, the approximation properties of BLOCKSWORLD have already been studied: it belongs to **APX**, but not to **PTAS** unless **P** = **NP**. In the following two sections, we present similar classification results for the remaining nine domains. We begin our investigation with domains which, like BLOCKSWORLD, are c-approximable but cannot be approximated arbitrarily well unless **P** = **NP**.

4 c-APPROXIMABLE DOMAINS

For most domains admitting polynomial planning algorithms, plans can be generated by simple greedy algorithms which satisfy the individual subgoals one after the other. If the number of steps required for each such individual goal can be bounded by a constant and an optimal plan requires at least one separate action per subgoal, then such an approach is c-approximating for some $c \in \mathbb{R}$. We first investigate a family of related domains sharing this property.[3]

Planning domain 1 SIMPLETRANSPORT
OBJECTS: *Locations V, packages P, and vehicles T.*
STATES: *Each package has an associated location or is inside a vehicle. Each vehicle has an associated location. Initially, no package is inside a vehicle.*
ACTIONS: *Moving a vehicle between any two locations; picking up a package with a vehicle at the same location; unloading a package from a vehicle.*
GOALS: *Each package has an associated goal location.*

[3] To keep discussion short, we introduce planning domains informally and refer to the literature for exact PDDL definitions [2, 7, 9, 10].

The SIMPLETRANSPORT domain is not itself a planning competition benchmark, but it is closely related to some of them:

- LOGISTICS is a generalization of SIMPLETRANSPORT where vehicles are partitioned into *trucks* and *airplanes* and locations are partitioned into a set of *cities*, each with a distinguished *airport* location. Trucks may only move between locations in the same city, and airplanes may only move between (and be initially located at) airports.
- MICONIC-STRIPS is the SIMPLETRANSPORT domain restricted to a single vehicle. In the MICONIC domains, the vehicle is usually called an *elevator*, the locations *floors*, and the packages *passengers*. Instead of saying that a passenger is picker up or unloaded, we say that he *boards* or *leaves* the elevator.
- MICONIC-SIMPLEADL is identical to MICONIC-STRIPS except that pickup and unload actions are replaced by a single *stop* action. When this action is applied at location l, all passengers at l which are not at their goal location board the elevator, and all passengers inside the elevator with goal location l leave.
- ZENOTRAVEL is identical to SIMPLETRANSPORT except that each vehicle has an associated *fuel level* (a natural number). Vehicles can only move if their fuel level is non-zero, and movement reduces their fuel level by one.[4] Refueling actions, increasing the fuel level of a vehicle by one, are applicable at any time.

It is quite easy to come up with c-approximation algorithms for these four domains. Indeed, for three of them, the greedy goal-at-a-time approach suffices. Therefore, all these domains belong to **APX**. We now show that it is hard to generate arbitrarily good plans.

Theorem 2 SIMPLETRANSPORT $\notin$ **PTAS**, *unless* **P** = **NP**. *This is true even in the special case with only one vehicle.*
Proof sketch: Proof by AP-reduction from the MINIMUM VERTEX COVER problem for graphs where all vertices have degree 2 or 3, which is not in **PTAS** unless **P** = **NP** [11]. A given graph $G = (V, E)$ is mapped to a SIMPLETRANSPORT task with one location for each vertex in V, plus one start location for the single vehicle. For each edge $\{u, v\} \in E$, one package must be moved from u to v and another one from v to u. The key observation is that the set of locations visited at least twice in a plan must form a vertex cover of G (i. e., includes at least one vertex from each edge in E). The degree restriction on the graphs guarantees that the size of a minimum vertex cover is $\Omega(|V|)$, which is important for proving that the reduction is approximation-preserving. ∎

While the theorem shows that there must be *some* value $c > 1$ for which there is no c-approximating algorithm for SIMPLETRANSPORT with a single vehicle if **P** $\neq$ **NP**, it does not provide us with an actual lower bound for approximability. In fact, the hardness result used for the reduction does not provide a lower bound either, but it is known that the MINIMUM VERTEX COVER problem for graphs with a vertex degree bound of 5 is not approximable within 1.0029 unless **P** = **NP** [1]. By adjusting our reduction accordingly, we can exploit this result to show that MICONIC-STRIPS is not approximable within 1.000112 unless **P** = **NP**.

Corollary 3 LOGISTICS $\in$ **APX** $\setminus$ **PTAS**, *unless* **P** = **NP**.
MICONIC-STRIPS $\in$ **APX** $\setminus$ **PTAS**, *unless* **P** = **NP**.
MICONIC-SIMPLEADL $\in$ **APX** $\setminus$ **PTAS**, *unless* **P** = **NP**.
ZENOTRAVEL $\in$ **APX** $\setminus$ **PTAS**, *unless* **P** = **NP**.

[4] There are also movement actions reducing fuel level by two, but there is no point in applying them.

Proof sketch: Greedy algorithms delivering one package (or passenger) at a time 2-approximate LOGISTICS and MICONIC-STRIPS and 3-approximate ZENOTRAVEL. (More sophisticated algorithms yield a $\frac{4}{3}$-approximation for LOGISTICS and a $\frac{7}{6}$-approximation for MICONIC-STRIPS.) These three domains are generalizations of SIMPLETRANSPORT with one vehicle, so they do not belong to **PTAS** unless $\mathbf{P} = \mathbf{NP}$.

MICONIC-SIMPLEADL is 2-approximable by an algorithm that moves to and stops at all initial floors of all passengers, then moves to and stops at all goal floors of all passengers. For proving non-membership in **PTAS**, essentially the same reduction as in the previous theorem can be used. ∎

There are two more c-approximable benchmark domains. One of them is DEPOT, a combination of SIMPLETRANSPORT (packages are transported between locations by trucks) and BLOCKSWORLD (packages at each location are stacked into towers). The BLOCKSWORLD subproblem uses named and limited table positions; however, trucks provide unlimited auxiliary storage, so limitation of table positions is not problematic.[5] Hardness for DEPOT follows from the BLOCKSWORLD and SIMPLETRANSPORT results immediately, and membership in **APX** is again fairly straightforward, so we do not define the domain formally but simply state our results briefly.

Theorem 4 DEPOT $\in$ **APX** $\setminus$ **PTAS**, *unless* $\mathbf{P} = \mathbf{NP}$.
Proof sketch: A straight-forward adaptation of the well-known 2-approximation for BLOCKSWORLD leads to a 3-approximation of DEPOT. The **PTAS** result follows because the domain generalizes both SIMPLETRANSPORT and BLOCKSWORLD. ∎

Finally, we turn our attention to the SATELLITE domain, which is defined as follows.

Planning domain 5 SATELLITE
OBJECTS: *Satellites S, instruments I, modes M, and directions D. Each instrument is located on one satellite, supports a set of modes, and has one direction as its calibration target.*
STATES: *Each satellite points at a certain direction. Each instrument may be powered on or off. Each instrument may or may not be calibrated. For each direction and mode, an image of this direction in this mode may or may not be available. Initially, no instrument is powered on or calibrated, and no images are available.*
ACTIONS: *Pointing a satellite at another direction; powering on an instrument (requires that no other instrument on this satellite is powered on; the instrument is not calibrated afterwards); powering off an instrument; calibrating an instrument (requires that its satellite points at its calibration target); taking an image of the pointing direction of a satellite in some mode (requires a powered-on, calibrated instrument supporting this mode on this satellite).*
GOALS: *Satellites may have associated target directions; images may need to be available for certain combinations of directions and modes.*

SATELLITE falls into the same approximation category as the other domains in this section.

Theorem 6 SATELLITE $\in$ **APX** $\setminus$ **PTAS**, *unless* $\mathbf{P} = \mathbf{NP}$.

[5] We require that there is at least one truck in each DEPOT task. DEPOT tasks without trucks would partition into sets of unrelated BLOCKSWORLD tasks with limited table positions. This BLOCKSWORLD variant is polynomially solvable [4], but we do not claim that it is in **APX**.

Proof sketch: A greedy algorithm solving one goal at a time (taking images first, then pointing satellites to their final directions, if any) is 6-approximating, showing membership in **APX**.

Limits of approximability can again be shown by an AP-reduction from MINIMUM VERTEX COVER for graphs with degree 2 or 3. A graph $G = (V, E)$ is mapped to a SATELLITE task with one satellite for each vertex $v \in V$ and one mode for each edge $e \in E$. The satellite corresponding to v has a single instrument supporting exactly the modes corresponding to edges incident to v. There is only one pointing direction d, which also serves as a calibration target for all satellites. An image of d must be taken in each mode. Given a plan for the task, the set of satellites used for taking images during the plan defines a vertex cover on G. The reduction is easily shown to be approximation-preserving. ∎

Instead of using many satellites with a single instrument each, we could also prove hardness for the case of a single satellite with many instruments. Similarly, instead of requiring many images of one direction, we could also require one image each of many directions.

5 NON-APPROXIMABLE DOMAINS

In this section, we discuss the remaining three planning domains: ROVERS, DRIVERLOG, and GRID. The results we present are mostly negative: While it is easy to see that all these domains are in **poly-APX**, none of them admit constant-factor approximations unless $\mathbf{P} = \mathbf{NP}$. Like some of the domains we have seen previously, all three domains have a transportation or path-planning flavor. Differently to the domains we have seen previously, however, they do not exhibit a *fixed roadmap*. In the ROVERS domain, different vehicles use different roads. In DRIVERLOG, some edges can only be traversed by car, others only on foot. In GRID, locations can be initially inaccessible and need to be opened by using keys. We will see that these aspects of the problems are responsible for hardness of approximation, starting with the ROVERS domain.

Planning domain 7 SIMPLEROVERS
OBJECTS: *Rovers R, waypoints W, and soil sample waypoints $S \subseteq W$. Each rover has an associated roadmap graph, which is a tree on a subset of waypoints. Each waypoint must be included in some roadmap graph. There is a waypoint common to all roadmap graphs called the lander location.*
STATES: *Each rover has an associated location, a waypoint on its roadmap graph. Rovers may carry a soil sample for a given waypoint or be empty. Soil data for a given waypoint may or may not have been transmitted. Initially, all rovers are empty and no soil data has been transmitted.*
ACTIONS: *Moving a rover between two waypoints connected on its roadmap; sampling soil on a soil sample waypoint with some rover (loads a soil sample into the rover, which must be empty; can only be done once for each waypoint); transmitting data for a soil sample with some rover (the soil sample must be inside the rover); emptying a rover.*
GOALS: *For each soil sample waypoint, a soil sample must be transmitted.*

The domain is called SIMPLEROVERS because it excludes features of the full ROVERS domain such as other science targets besides soil samples (namely, rock samples and images), restricted visibility for transmitting science data, and rovers equipped for only a subset of science tasks [9]. Including these would require many more details without providing much further insight. Our results equally apply to the unrestricted domain.

Theorem 8 ROVERS $\in$ **poly-APX** $\setminus$ **APX**, *unless* **P** $=$ **NP**. *This is true even in the restricted* SIMPLEROVERS *domain.*
Proof sketch: Any solvable ROVERS task can easily be solved one goal at a time, which shows membership in **poly-APX**. For the non-approximability result, we reduce from the MINIMUM SET COVER problem: Given a collection of subsets $\mathcal{C}$ of a set S, find a sub-collection $\mathcal{C}' \subseteq \mathcal{C}$ with $\bigcup_{C' \in \mathcal{C}'} C' = S$, using the cardinality of the chosen sub-collection as the measure of the solution. The problem is not c-approximable for any $c \in \mathbb{R}$ unless **P** $=$ **NP** [1].

For a given instance $(\mathcal{C}, S)$, the corresponding SIMPLEROVERS task has one rover for each subset in $\mathcal{C}$, and one soil sample waypoint for each element of S. The roadmaps for the rovers are designed in such a way that the rover for $C \in \mathcal{C}$ can reach exactly those soil sample waypoints corresponding to elements of C. All soil sample waypoints reachable by a rover are very near to each other and a very long distance from the original rover location, so that plan lengths are dominated by the number of rovers used for solving the goals. Any such set of rovers defines a solution to the set covering instance; the reduction is approximation-preserving. ■

We presented this proof first because the other two are similar in spirit, but somewhat more involved technically. In both cases, the key idea is again that selecting a certain subset for the set cover allows achieving a set of goals corresponding to the elements of that subset, and the total length of the plan is dominated by the number of chosen subsets. This is enforced by requiring that a very large distance needs to be crossed in the planning task at least once for each chosen subset. We continue our exposition with the DRIVERLOG domain.

Planning domain 9 DRIVERLOG
OBJECTS: *Trucks T, drivers D, major locations L, minor locations L_M (disjoint from L), and packages P. There is a connected road graph, whose vertices are the major locations, and a connected footpath graph, whose vertices are the major and minor locations. In the footpath graph, the degree of all minor locations is 2, and both adjacent vertices of a minor location are major locations.*
STATES: *A truck is located at some major location, a package at some major location or inside a truck, a driver at some major or minor location or inside a truck. Initially, all are located at major locations.*
ACTIONS: *Having a driver walk from one location to another (must be connected in the footpath graph); driving a truck from one location to another (must be connected in the road graph, must have a driver inside the truck); boarding a truck with a driver at the same location (must not have another driver inside the truck); debarking from a truck with a driver; loading a package into a truck at the same location; unloading a package from a truck.*
GOALS: *Packages, drivers, and trucks may have associated goal locations (which are always major locations).*

The somewhat peculiar restriction on footpath graphs merely ensures that walking from one major location to another requires two actions rather than just one, to reflect the fact that walking usually takes longer than driving. There is never a good reason to walk from a major location to a minor location except to continue to the other adjacent major location, so instead of modeling minor locations explicitly, our following proof assumes that there is only one kind of location and assigns a cost of 2 to walk actions.

Theorem 10 DRIVERLOG $\in$ **poly-APX** $\setminus$ **APX**, *unless* **P** $=$ **NP**. *This is true even when there is only a single driver, and goals are only defined for packages.*

Proof sketch: Again, a simple greedy algorithm solving one goal at a time suffices for showing membership in **poly-APX**. For hardness, we again provide an approximation-preserving reduction from MINIMUM SET COVER, mapping instance $(\mathcal{C}, S)$ to the following DRIVERLOG task:

- There is a central location $l_\star$, where the only driver starts. This is connected to *subset locations* l_C for each $C \in \mathcal{C}$ by very long disjoint footpaths. There is a truck at each subset location.
- For each element $s \in S$, there is an *element start location* l_s^- and an *element goal location* l_s^+, and a road connecting the two. There is a truck and a package at each element start location. The package must be moved to the corresponding element goal location.
- For each subset $C \in \mathcal{C}$ and element $s \in C$, there is a location $l_{C,s}$ connected to l_C by a road and to l_s^- by a footpath.

It is easy to see that the subsets corresponding to the subset locations visited in a plan correspond to a set cover. By making the distances between central location and subset locations long enough, we can ensure that plan length is dominated by the cardinality of this set cover, so that the reduction is approximation-preserving. (The DRIVERLOG task does not have connected road and footpath graphs, but this can be repaired by introducing connecting paths that are too long to be useful for generating short plans.) ■

The last missing domain is GRID, another variation of the transportation theme.

Planning domain 11 GRID
OBJECTS: *A single robot, locations L arranged into a rectangular grid, key shapes S, and keys K. Each key has a shape from S. A location may be initially locked by a lock with a certain shape from S.*
STATES: *The robot has an associated location. Keys have an associated location or are carried. Locations can be open or locked. Initially, no key is carried.*
ACTIONS: *Moving the robot to an adjacent open location; picking up a key at the current robot location if no key is currently being carried; dropping a key being carried; switching the currently carried key with a key at the current robot location; opening a locked location adjacent to the robot with a key of the same shape as the lock.*
GOALS: *Keys may have associated goal locations.*

From our earlier analysis of the GRID domain, we know that GRID $\in$ **poly-APX** [5]. This reference also contains two proofs of hardness for the bounded plan existence problem in this domain, one based on satisfiability and one based on a traveling-salesperson type problem. However, neither of these reductions can be used to prove hardness of approximation. Indeed, the GRID tasks generated by the satisfiability reduction can easily be 2-approximated, and the restricted version of the GRID domain considered in the traveling salesperson reduction even belongs to **PTAS**. We thus require a new reduction.

Theorem 12 GRID $\in$ **poly-APX** $\setminus$ **APX**, *unless* **P** $=$ **NP**.
Proof sketch: Membership in **poly-APX** is known [5]. Again, we map MINIMUM SET COVER instances $(\mathcal{C}, S)$ to planning tasks via an approximation-preserving reduction. The GRID task has three kinds of keys: for each subset $C \in \mathcal{C}$, there is a *subset key* k_C with key shape S_C; for each subset $C \in \mathcal{C}$ and element $s \in C$, there is an *element key* $k_{C,s}$ with key shape S_s; for each element $s \in S$, there is a *goal key* k_s with a key shape that cannot be used for opening any location. Goals are only defined for the goal keys.

The grid of the planning tasks consists of two areas which are very distant from each other. The first area contains the initial locations of the element and goal keys and the robot as well as the goal locations of the goal keys. The second area contains the initial locations of the subset keys. The initial location of element key $k_{C,s}$ is locked with a lock of shape S_C, and the initial location of goal key k_s is locked with a lock of shape S_s. These are the only locations which are initially locked.

To solve the task, the robot must unlock the initial locations of all goal keys, which requires obtaining keys of shape S_s for all $s \in S$. This in turn requires obtaining subset keys for a collection of subsets that covers S, so the set of subset keys picked up in a plan defines a set cover. By putting a large enough distance between the first and second area, we can ensure that the cardinality of this set cover dominates total plan length. (Note that the robot can only carry one key at a time.) Thus, the reduction is approximation-preserving. ∎

We point out that the hardness of approximately solving GRID tasks is largely due to the necessity of opening locations. If we restrict ourselves to tasks where opening is impossible (i. e., the shapes of locks are disjoint from the shapes of keys), we can provide a $(2 + \varepsilon)$-approximating planning algorithm for all $\varepsilon > 0$. However, we will not prove this result here, but instead turn to discussion.

6 SUMMARY AND CONCLUSION

The approximation properties of the benchmark domains are summarized in Fig. 1. There are some interesting observations to be made. First, there are no domains in the benchmark set that fall into the class **PTAS** $\setminus$ **PO**. There is no fundamental reason why this should be the case, except maybe for the fact that **PTAS** problems are rare in general.[6] However, as we pointed out when discussing the GRID domain, there are **NP**-hard special cases of the competition domains that do admit polynomial-time approximation schemes.

Looking into the hardness proofs, we observe that we could easily get away with only using two different optimization problems for our reductions, both of which are set covering problems. We believe that this is no coincidence: Set covering problems arise naturally in planning tasks through positive interactions of subgoals. We believe that finding a good way of integrating heuristics for set covering problems into admissible heuristics for domain-independent planning could lead to a significant advance for optimal planning algorithms. However, we consider this a very challenging avenue of research.

What to do with these results? After our earlier results [5, 6] on the decision complexity of planning in the benchmark domains, we see our work as a second step on the road towards a better understanding of the benchmark suite, which we believe to be critical to provide "the level of understanding required for its effective use as a benchmark" [13, Slaney and Thiébaux on BLOCKSWORLD]. In comparison to Slaney and Thiébaux's effort, at least two more steps are required until we can consider these standard benchmarks "understood": one is the identification of phase transition regions to know where the hard instances are, and the other is the provision of (domain-dependent) optimal solvers as a reference point for evaluation of solution quality.

In addition to such deeper domain-specific studies, we believe that some important insights into domain-*independent* planning can be obtained by generalizing the analyses conducted for the benchmark domains in a suitable way, for example by identifying syntactic or semantic fragments of PDDL for which the planning problem belongs to **APX**, in the spirit of the work on tractable subclasses of the SAS$^+$ planning formalism [3, 8].

[6] To readers familiar with other approximation classes, we point out that there *is* a good reason why there is no domain in the class **FPTAS** $\setminus$ **PO**; this follows quite easily from Theorem 3.15 in the book by Ausiello et al. [1]. Similarly, we cannot have planning domains in the class **exp-APX** $\setminus$ **poly-APX**.

$\in$ **PO**:
GRIPPER, MOVIE, PROMELA-OPTICALTELEGRAPH, PROMELA-PHILOSOPHERS, PSR, SCHEDULE

$\in$ **PTAS** $\setminus$ **PO** (unless **P** $=$ **NP**):
none

$\in$ **APX** $\setminus$ **PTAS** (unless **P** $=$ **NP**):
BLOCKSWORLD, LOGISTICS, MICONIC-SIMPLEADL, MICONIC-STRIPS, ROVERS, SATELLITE, ZENOTRAVEL

$\in$ **poly-APX** $\setminus$ **APX** (unless **P** $=$ **NP**):
DEPOT, DRIVERLOG, GRID

$\in$ **NPO** $\setminus$ **poly-APX** (unless **P** $=$ **NP**):
FREECELL, MICONIC-FULLADL, MPRIME, MYSTERY, PIPESWORLD*

$\notin$ **NPO**:
AIRPORT, ASSEMBLY

Figure 1. Classification results. For PIPESWORLD, only hardness is known; membership in **NPO** is open.

REFERENCES

[1] Giorgio Ausiello, Pierluigi Crescenzi, Giorgo Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi, *Complexity and Approximation*, Springer-Verlag, 1999.

[2] Fahiem Bacchus, 'The AIPS'00 planning competition', *AI Magazine*, **22**(3), 47–56, (2001).

[3] Christer Bäckström and Bernhard Nebel, 'Complexity results for SAS$^+$ planning', *Computational Intelligence*, **11**(4), 625–655, (1995).

[4] Naresh Gupta and Dana S. Nau, 'On the complexity of blocks-world planning', *Artificial Intelligence*, **56**(2–3), 223–254, (1992).

[5] Malte Helmert, 'Complexity results for standard benchmark domains in planning', *Artificial Intelligence*, **143**(2), 219–262, (2003).

[6] Malte Helmert, 'New complexity results for classical planning benchmarks', in *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*, eds., Derek Long, Stephen F. Smith, Daniel Borrajo, and Lee McCluskey, pp. 52–61. AAAI Press, (2006).

[7] Jörg Hoffmann and Stefan Edelkamp, 'The deterministic part of IPC-4: An overview', *Journal of Artificial Intelligence Research*, **24**, 519–579, (2005).

[8] Peter Jonsson and Christer Bäckström, 'State-variable planning under structural restrictions: Algorithms and complexity', *Artificial Intelligence*, **100**(1–2), 125–176, (1998).

[9] Derek Long and Maria Fox, 'The 3rd International Planning Competition: Results and analysis', *Journal of Artificial Intelligence Research*, **20**, 1–59, (2003).

[10] Drew McDermott, 'The 1998 AI Planning Systems competition', *AI Magazine*, **21**(2), 35–55, (2000).

[11] Christos H. Papadimitriou and Mihalis Yannakakis, 'Optimization, approximation, and complexity classes', *Journal of Computer and System Sciences*, **43**, 425–440, (1991).

[12] Bart Selman, 'Near-optimal plans, tractability, and reactivity', in *Proceedings of the Fourth International Conference on Principles of Knowledge Representation and Reasoning (KR'94)*, eds., Jon Doyle, Erik Sandewall, and Pietro Torasso, pp. 521–529. Morgan Kaufmann, (1994).

[13] John Slaney and Sylvie Thiébaux, ‘Blocks World revisited’, *Artificial Intelligence*, **125**, 119–153, (2001).