Approximation Properties of Planning Benchmarks

Malte Helmert, Robert Mattmüller and Gabi Röger
Institut für Informatik, Albert-Ludwigs-Universität Freiburg
Georges-Köhler-Allee, Gebäude 052
79110 Freiburg, Germany
{helmert,mattmuel,roeger}@informatik.uni-freiburg.de

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 well-known approximation classes **PO**, **PTAS**, **APX**, **poly-APX**, and **NPO**.

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 studied by Slaney and Thiébaux (2001) and other researchers (Gupta & Nau 1992; Selman 1994), the published results on the computational complexity of planning benchmarks are rather coarse-grained. In most cases, they are limited to classifying the computational complexity 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 (Helmert 2003; 2006). 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 (Ausiello *et al.* 1999); it is the topic of the area of *approximation algorithms*. In this contribution, we investigate the propositional planning domains introduced in the first four International Planning Competitions (McDermott 2000; Bacchus 2001; Long & Fox 2003; Hoffmann & Edelkamp 2005) from this perspective, providing a complete classification with respect to the following approximation classes:

- Domains where optimal plans can be generated in polynomial time (**PO**).
- Domains where plans at most c times as long as the optimal solution can be generated in polynomial time for *all* rational numbers $c > 1$ (**PTAS**).
- Domains where plans at most c times as long as the optimal solution can be generated in polynomial time for *some* rational number $c > 1$ (**APX**).
- Domains where plans of length polynomially bounded by the optimal solution 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**).

We start our investigation by providing some background on the theory of approximation algorithms. Subsequently, we summarize the existing complexity results for planning benchmarks and explain how they relate to our investigation. This overview is followed by two sections containing our new results, 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.

Approximation algorithms

Because we cannot expect our readers to be intimately familiar with approximation algorithms, we first review some basic definitions in so far as they are required for our analysis. For a thorough introduction, we refer to the textbook by Ausiello et al. (1999).

The central concept in the theory of approximation algorithms is that of *optimization problems*, which play the same role that decision problems do in classical complexity theory. Optimization problems are either *minimization problems* or *maximization problems*.

Definition 1 ***minimization problem***
*A **minimization problem** is given by a set of **instances** $\mathcal{I}$, which are words over a finite alphabet. Each instance $I \in \mathcal{I}$ has a set of **solutions** $S(I)$ (possibly empty), also words over a finite alphabet. Each solution $s \in S(I)$ has an associated **measure** $m(s) \in \mathbb{N}_0$. Its **performance ratio** is defined as $\frac{m(s)}{m(s^*)}$ for some **optimal** solution $s^* \in S(I)$, i. e., a solution minimizing the measure among all elements of $S(I)$.*

Maximization problems are similarly defined; however, they are not relevant to this work. The algorithmic task associated with a minimization problem is the following: Given an instance of the problem, compute a solution with measure as small as possible, or prove that no solution exists.

Like in complexity theory, in practice the instances of an optimization problem are described by graphs, sets, or functions rather than words over an alphabet, and it is assumed that a "reasonable encoding" is used for representing such mathematical structures in contexts that require "proper" words, for example as inputs to Turing Machines. Such leniency is justified by the fact that problems that differ only in encoding share the same approximability (and complexity) properties if the encoding lengths are polynomially related.

Examples of minimization problems are the MINIMUM VERTEX COVER and MINIMUM SET COVER problems.

Problem 2 MINIMUM VERTEX COVER
INSTANCE: *A graph* (V, E).
SOLUTION: *A subset of vertices* $V' \subseteq V$ *such that for all edges* $\{u, v\} \in E$, $u \in V'$ *or* $v \in V'$.
MEASURE: $|V'|$.

Problem 3 MINIMUM SET COVER
INSTANCE: *A finite set* S *and collection* C *of subsets of* S.
SOLUTION: *A set cover of* S*, i. e., a subset* $C' \subseteq C$ *such that each element of* S *is contained in some set in* C'.
MEASURE: $|C'|$.

We say that an optimization problem is *polynomially solvable* iff some polynomially time-bounded Turing Machine decides for each instance whether or not a solution exists, and writes a solution to the tape if one exists. We say that it is f*-approximable* for some function $f : \mathbb{N}_0 \to \mathbb{R}$ iff it is polynomially solvable by a machine such that the performance ratio of the solution generated for any instance I is bounded by $f(\|I\|)$. We say that it is *c-approximable* for some $c \in \mathbb{R}$ iff it is f-approximable for the constant function which always equals c. We say that it is *optimally solvable* iff it is 1-approximable. Optimal solvability by non-deterministic Turing Machines is similarly defined.

We can now define the following *approximation classes*:

Definition 4 PO, PTAS, APX, poly-APX, NPO
An optimization problem belongs to approximation class...

- **PO** *(***P***-optimization) iff it is optimally solvable.*
- **PTAS** *(polynomial-time approximation scheme) iff it is* c*-approximable for all real numbers* $c > 1$.
  The c*-approximating Turing Machine may depend on* c.
- **APX** *(approximable) iff it is* c*-approximable for some real number* $c > 1$.
- **poly-APX** *(poly-approximable) iff it is* p*-approximable for some polynomial* $p \in \mathbb{R}[x]$.
- **NPO** *(***NP***-optimization) iff it is optimally solvable by a non-deterministic Turing Machine.*

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**. For example, if **P** $\neq$ **NP**, then MINIMUM VERTEX COVER belongs to **APX** $\setminus$ **PTAS** and MINIMUM SET COVER belongs to **poly-APX** $\setminus$ **APX** (Ausiello *et al.* 1999).

Planning domains

For the purposes of this work, planning domains are minimization problems. Instances of planning domains are called *tasks*. Solutions to planning tasks are sequences of *actions* called *plans*. In this work, the measure of a plan is its length (action count). We define planning domains informally in terms of *objects*, *states*, *actions*, and *goals*, notions that we hope to be familiar enough not to require formal definitions. Here is an example of a transportation domain.

Planning domain 5 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.*

As discussed in the introduction, the scope of this paper is limited to the propositional domains of the first four International Planning Competitions. There is little work in the literature on the approximation properties of planning domains, the main exception being Selman's (1994) paper which links approximability to *reactive planning* and shows that, while being 2-approximable, the BLOCKSWORLD domain is not in **PTAS** unless **P** $=$ **NP**.

However, the literature does contain classifications of these planning domains from a classical complexity theory point of view (Helmert 2003; 2006), addressing the *plan existence* and *bounded plan existence* decision problems. We can benefit from this body of work in the following ways:

- Domains where shortest plan lengths are polynomially bounded in the instance size belong to **NPO**. Other domains do not belong to **NPO**, because it is not possible to *write down* a plan in polynomial time in these domains.
- 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**.

Fig. 1 shows the classification obtained from these rules and the complexity results in the literature. The planning domains listed in the second category are those for which plan existence is an easy problem, while bounded plan existence is hard. [1] For these domains, the existing complexity results do not give a satisfactory account of their approximation properties. Therefore, these are the planning domains we will chiefly be concerned with in this paper.

[1]For some domains in the second category, the literature does not contain proofs of **NP**-completeness for *bounded plan existence*. We remark that such proofs can be easily derived from the proofs in the following sections showing hardness of approximation.

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

between **PO** and **poly-APX** (mostly open):
BLOCKSWORLD, DEPOT, DRIVERLOG, GRID, LOGISTICS, MICONIC-SIMPLEADL, MICONIC-STRIPS, ROVERS, SATELLITE, ZENOTRAVEL

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

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

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

We begin our investigation with domains which, like BLOCKSWORLD, admit an approximation algorithm but cannot be approximated arbitrarily well unless **P** = **NP**.

c-approximable domains

For many of the "open" domains from Fig. 1, 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 will be c-approximating for some constant $c \in \mathbb{R}$. We first investigate a family of related domains sharing this property.[2]

Planning domain 6 LOGISTICS
The LOGISTICS *domain 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.*

Planning domain 7 MICONIC-STRIPS
The MICONIC-STRIPS *domain 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*. For uniformity, we do not use this terminology.

Planning domain 8 MICONIC-SIMPLEADL
The MICONIC-SIMPLEADL *domain is identical to the* MICONIC-STRIPS *domain except that pickup and unload actions are replaced by a single* stop *action. When this action is applied at location l, all packages at l which are not at their goal location are picked up, and all packages inside the vehicle with goal location l are unloaded.*

[2]For space reasons, we define most planning domains rather informally and refer to the literature for the exact PDDL definitions (McDermott 2000; Bacchus 2001; Long & Fox 2003; Hoffmann & Edelkamp 2005).

Planning domain 9 ZENOTRAVEL
The ZENOTRAVEL *domain 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 moving them reduces their fuel level by one. Refueling actions, increasing the fuel level of a vehicle by one, are applicable at any time.*

Compared to the original PDDL domain, our definition of ZENOTRAVEL lacks the *zoom* action, which uses up two units of fuel but is otherwise like normal movement. Moreover, in some of the PDDL domains it is possible to have packages without an associated goal location. Clearly, neither property influences the outcome of our analysis.

It is quite easy to come up with c-approximation algorithms for these four domains. Indeed, for three of the four domains, the greedy goal-at-a-time approach suffices.

Theorem 10 LOGISTICS $\in$ APX
LOGISTICS *is* 2*-approximable.*

Proof: By *transporting* a package to location l with vehicle v, we mean the following sequence of actions: Move v to the current location of the package unless it is there already, pick up the package with v, move v to l, then drop the package.

For each package, generate the following actions:

- If the current location of the package is not an airport and in a different city than the goal location, transport the package to the airport of its current city with some truck.
- If the goal location of the package is in a different city than the current location (which must then be an airport), transport the package to the airport of the goal location city with some airplane.
- If the package is not at its goal location, transport it there with some truck.

The algorithm clearly runs in polynomial time. It will fail only if no airplane is available or if no truck is available in some city where a package needs to be moved. In these cases, the task has no solution.

The number of pickup or drop actions in the plan generated by our algorithm is minimal among all plans, and hence bounded by the optimal plan length L^*. Moreover, the plan contains at most as many movement actions as non-movement (pickup or drop) actions, bounding total plan length by $2L^*$. The algorithm is thus 2-approximating. ∎

As an immediate corollary, we obtain the following result.

Theorem 11 MICONIC-STRIPS $\in$ APX
MICONIC-STRIPS *is* 2*-approximable.*

Proof: The MICONIC-STRIPS domain is a special case of LOGISTICS (one truck, one city, no airplanes). ∎

Note that the algorithm would not be c-approximating for any $c \in \mathbb{R}$ in either domain if we only counted the number of *movements* rather than the number of total actions. However, better (but more complicated) approximation algorithms exist, which are 2-approximating under this measure and $\frac{4}{3}$-approximating when counting all actions. For MICONIC-STRIPS, we can even obtain a $\frac{7}{6}$-approximating algorithm when counting all actions. For space reasons, we only present the simpler results above here.

The same greedy approach also works for ZENOTRAVEL.

Theorem 12 ZENOTRAVEL $\in$ APX
ZENOTRAVEL *is* 3*-approximable.*
Proof: We use the same algorithm as for LOGISTICS except that every movement is preceded by a *refuel* action when necessary. For the optimal plan, the same bound as for LOGISTICS applies. The approximate plan may now contain as many as L^* additional refuel actions, so that we can bound the total number of actions by $3L^*$ to prove 3-approximability. ∎

Using techniques from the improved MICONIC-STRIPS algorithm, we can 2-approximate the ZENOTRAVEL domain. Again, we choose not to present this result here.

Solving the individual goals one at a time does not lead to a c-approximation algorithm for MICONIC-SIMPLEADL because the absence of explicit *pickup* and *unload* actions means that the optimal plan length is no longer dominated by those. However, another simple algorithm gives us 2-approximability.

Theorem 13 MICONIC-SIMPLEADL $\in$ APX
MICONIC-SIMPLEADL *is* 2*-approximable.*
Proof: Consider the following plan: In the first phase, stop at the initial vehicle location if there are any packages there which need to be moved. Then, move to and stop at all other locations from which packages need to be moved, in any order. In the second phase, move to and stop at goal locations of packages picked up in the first phase (except for the location reached at the end of the first phase). Clearly, this is a polynomial-time solution algorithm.

In any plan, the vehicle must move to and stop at all locations moved to and stopped at during the first phase of the algorithm. Therefore, the number of actions for the first phase is bounded by the optimal plan length L^*. Similarly, the number of actions for the second phase is bounded by L^*. Putting this together, the generated plan contains no more than $2L^*$ actions, proving the 2-approximability result. ∎

While it is good to have c-approximation algorithms available for those transportation domains, the question arises how much better we can do. We have already mentioned that the 2-approximation result for MICONIC-STRIPS can be strengthened to a $\frac{7}{6}$-approximation. Is there a limit to how much better we can get? The following theorem shows that there is.

Theorem 14 MICONIC-STRIPS $\notin$ PTAS
MICONIC-STRIPS $\notin$ **PTAS**, *unless* **P** $=$ **NP**.
Proof: We prove that MICONIC-STRIPS $\in$ **PTAS** implies that the MINIMUM VERTEX COVER problem, restricted to a certain set of graphs, is in **PTAS**, which implies **P** $=$ **NP**.

Papadimitriou and Yannakakis (1991) have shown that MINIMUM VERTEX COVER for graphs where all vertices have a degree of at most 3 is not in **PTAS** unless **P** $=$ **NP**. In addition to the upper bound on the degree, we make the further restriction that the degree of each vertex must be at least 2. This does not affect the hardness result since other vertices can be trivially dealt with: As long as there are any vertices of degree 1, pick any such vertex, add its neighbor to the vertex cover, and remove the vertex and its neighbor along with any incident edges from the graph. If no vertices of degree 1 are left, remove all vertices of degree 0. All decisions made in this preprocessing step are optimal, so if there were arbitrarily good approximation results for our restricted graph class, then we could obtain an arbitrarily good approximation for arbitrary graphs with degree bound of 3 by combining the algorithm for the restricted graph class with the preprocessing technique.

So let us assume that MICONIC-STRIPS $\in$ **PTAS**. We show that the restricted MINIMUM VERTEX COVER problem is in **PTAS** by constructing a $(1 + \varepsilon)$-approximating vertex covering algorithm for arbitrary $\varepsilon > 0$. Given a graph (V, E), the algorithm works as follows:

- Construct a MICONIC-STRIPS task with locations $V \cup \{v_0\}$, where $v_0 \notin V$ is the initial location of the unique vehicle. For each edge $\{u, v\} \in E$, we generate two packages: one with initial location u and goal location v, and one with initial location v and goal location u.
- Compute a $(1 + \frac{1}{16}\varepsilon)$-approximate plan for the MICONIC-STRIPS task.
- Return the set of vertices in V that correspond to locations visited at least twice in the plan.

Clearly, this is a polynomial-time algorithm. We must show that it returns a vertex cover and that it is $(1 + \varepsilon)$-approximating.

To see the former, assume that the algorithm does not return a vertex cover, i. e., that there is some edge $\{u, v\} \in E$ so that neither u nor v are included in the result set. This implies that in the plan, both locations u and v are visited at most once, so either u is never visited after v or vice versa. In the first case, the goal condition for the package that must be moved from v to u cannot be satisfied by the plan; in the second case, the goal condition for the package that must be moved from u to v cannot be satisfied by the plan, a contradiction.

Let us now estimate the quality of the returned vertex cover. Let $V^* \subseteq V$ be an optimal vertex cover. Then the corresponding planning task can be solved by first moving the vehicle to all locations in V^* (first phase), then to all locations in $V \setminus V^*$ (second phase), then again to all locations in V^* (third phase), picking up packages whenever their initial location is first reached and unloading packages at their goal location when possible. Within a given phase, locations are visited in arbitrary order.

To see that this is indeed a solution, we must make sure that for each package, the initial location is visited for the first time before the goal location is visited for the last time. If the initial location is in V^*, then it is visited in the first phase and the goal location is visited in the second or third phase, so the condition is satisfied. If the goal location is in V^*, then the initial location is visited in the first or second phase and the goal location is visited in the third phase, so again the condition is satisfied. There cannot be a package where neither the initial location u nor goal location v is in V^*, since then the edge $\{u, v\}$ would not be covered by V^*.

The number of actions in the plan is $L = 4|E| + |V| + |V^*|$, where the first term comes from the pickup and unload actions required for the $2|E|$ packages, the second term comes from the cost for moving to each location in V at least once, and the last term comes from the cost for moving to the locations in V^* again. Given that there exists a solution of length L, an *optimal* solution must have length at most L (in fact, the solution *is* optimal, but we do not need this for the proof), and hence the approximate plan generated within the set covering algorithm has length at most $(1 + \frac{1}{16}\varepsilon)L$.

The approximate plan must contain at least $2|E|$ pickup actions and $2|E|$ unload actions and visit each location in V at least once, so the number of locations visited at least twice, and hence the size of the returned vertex cover, is at most $K = (1 + \frac{1}{16}\varepsilon)L - 4|E| - |V| = |V^*| + \frac{1}{16}\varepsilon(4|E| + |V| + |V^*|)$. Because each vertex has degree at most 3, we have $|E| \leq 3|V^*|$. Moreover, because each vertex has degree at least 2, we have $|V| \leq |E|$ and thus $|V| \leq 3|V^*|$. Plugging these into the formula, we get $K \leq |V^*| + \frac{1}{16}\varepsilon(4 \cdot 3|V^*| + 3|V^*| + |V^*|) = (1 + \varepsilon)|V^*|$, as required. ∎

While the theorem shows that there must be *some* value $c > 1$ for which there is no c-approximating algorithm for MICONIC-STRIPS if **P** $\neq$ **NP**, it does not provide us with an actual lower bound for approximability. Indeed, the result by Papadimitriou and Yannakakis 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** (Ausiello *et al.* 1999). By adjusting our reduction accordingly, we can exploit this result to show that MICONIC-STRIPS is not approximable within 1.000112 unless **P** = **NP** Obviously, there is still a significant gap between this value and the $\frac{7}{6}$-approximability result mentioned earlier.

As a corollary, we immediately get similar hardness results for two of the other transportation domains.

Theorem 15 LOGISTICS $\notin$ PTAS
LOGISTICS $\notin$ **PTAS**, *unless* **P** = **NP**. *This is true even if we restrict the domain to tasks with only one truck, one city and no airplanes, or only one airplane, no trucks, and all cities consisting of only one location.*
Proof: These restricted LOGISTICS tasks are exactly the MICONIC-STRIPS tasks. ∎

Theorem 16 ZENOTRAVEL $\notin$ PTAS
ZENOTRAVEL $\notin$ **PTAS**, *unless* **P** = **NP**. *This is true even if we restrict the domain to tasks with only one vehicle.*
Proof: ZENOTRAVEL tasks with one vehicle are a generalization of MICONIC-STRIPS tasks (consider ZENOTRAVEL tasks where the vehicle has enough fuel never to require refueling in a reasonable plan). ∎

For the last remaining domain, the proof needs to be only slightly adjusted.

Theorem 17 MICONIC-SIMPLEADL $\notin$ PTAS
MICONIC-SIMPLEADL $\notin$ **PTAS**, *unless* **P** = **NP**.
Proof: We use the same reduction from the restricted case of MINIMUM VERTEX COVER as above, only that this time we use a $(1 + \frac{1}{4}\varepsilon)$-approximating algorithm for MICONIC-SIMPLEADL instead of a $(1 + \frac{1}{16}\varepsilon)$-approximating algorithm for MICONIC-STRIPS.

As in the MICONIC-STRIPS proof, if V^* is an optimal vertex cover, then there exists a plan with $|V| + |V^*|$ movement actions. Instead of pickup and drop actions, the approximate plan now contains $|V| + |V^*|$ stop actions, one after each movement, for a total plan length of $2(|V|+|V^*|)$.

The planning algorithm then produces a plan of length at most $(1 + \frac{1}{4}\varepsilon) \cdot 2(|V| + |V^*|)$. We can assume that at most half of those are movement actions: If the plan contains two or more movements in a sequence, these can be replaced by single movements, and if the last action is a movement, it can be removed. Therefore, the total number of movements in the plan can be assumed to be at most $(1+\frac{1}{4}\varepsilon)(|V|+|V^*|)$. All locations in V must be visited at least once, so that at most $|V^*| + \frac{1}{4}\varepsilon(|V| + |V^*|) \leq (1 + \varepsilon)|V^*|$ locations are visited at least twice, where the inequality is again due to the fact that $|V| \leq 3|V^*|$. This shows that the MINIMUM VERTEX COVER algorithm is $(1 + \varepsilon)$-approximating. ∎

Two more benchmark domains belong to **APX** but not to **PTAS** unless **P** = **NP**. The first of these is the DEPOT domain, a combination of BLOCKSWORLD and SIMPLETRANSPORT featuring packages that can be moved between locations by trucks and stacked into towers at locations. The BLOCKSWORLD subtask uses named and limited table positions; however, trucks provide unlimited auxiliary storage.

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 18 DEPOT $\in$ APX
DEPOT *is* 3*-approximable.*
Proof: We view the DEPOT task as a BLOCKSWORLD task, building the union of the stacks at all locations, and solve this task with the standard two-phase 2-approximation algorithm. To translate the generated plan back, we replace actions that drop a block onto an auxiliary table position by actions that move a truck to the location, then load the block into the truck. Similarly, we replace actions that pick up blocks from an auxiliary table position with actions that move the truck holding the block to the location, then unload the block. At most half of the actions in the BLOCKSWORLD task involve auxiliary table positions (the other ones are *stack* or *unstack* actions), which leads to a performance ratio of at most 3 for the DEPOT plan. ∎

Note that we assume 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 according to a result by Gupta and Nau (1992), but we do not claim that it is in **APX**.

Theorem 19 DEPOT $\notin$ PTAS
DEPOT $\notin$ **PTAS**, *unless* **P** = **NP**. *This is true even if we restrict the domain to tasks with only one location, or to tasks with only one truck and no package on top of another package in the initial state or goal specification.*
Proof: BLOCKSWORLD tasks can be encoded as DEPOT tasks of the first kind, MICONIC-STRIPS tasks as DEPOT tasks of the second kind. ∎

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

Planning domain 20 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.*

Again, it is fairly obvious that the domain is in **APX**.

Theorem 21 SATELLITE $\in$ APX
SATELLITE *is* 6*-approximable.*

Proof: A greedy planning algorithm first solves all image goals by a sequence of 6 steps each: After selecting an instrument supporting the required image mode (if none exists, the task has no solution), power on the instrument, point its satellite at the calibration target, calibrate the instrument, point its satellite at the image direction, take the image, and power off the instrument. After taking all images, each satellite with a pointing goal is turned to the required direction if it differs from the current direction. As a post-processing step, for each satellite that has been used for taking an image, the final powering-off action is removed.

The length of the plan is at most $6I + P$, where I is the number of image goals and P is the number of pointing goals where the goal direction is different from the initial direction. (Note that it may be the case that a satellite originally pointed at the goal direction and has been pointed away to take an image. However, in this case, the action required for pointing it back at the original direction has been "paid for" in the step where extraneous power-off actions are removed.)

An optimal plan clearly contains at least $I + P$ actions, since it must contain a separate image-taking action for each image goal and a pointing action for each pointing goal where the goal direction differs from the initial direction. Because $6I + P \leq 6(I + P)$, this concludes the proof. ■

We wrap up this section by showing that the SATELLITE domain cannot be approximated arbitrarily well.

Theorem 22 SATELLITE $\notin$ PTAS
SATELLITE $\notin$ **PTAS**, *unless* **P** $=$ **NP**.

Proof: Again we employ a proof by contradiction, showing that the existence of arbitrarily good approximation algorithms for SATELLITE leads to arbitrarily good approximation algorithms for MINIMUM VERTEX COVER on graphs where all vertices have degree 2 or 3.

Given a graph (V, E), the $(1 + \varepsilon)$-approximation algorithm for MINIMUM VERTEX COVER works as follows:

- Construct a SATELLITE task with one satellite for each vertex $v \in V$, each with a single instrument, one mode for each edge $e \in E$, and two directions d_1 and d_2, where d_1 is the initial pointing direction and calibration target for all satellites. The instrument on the satellite for vertex v supports precisely those modes corresponding to edges in E to which this vertex is incident in the graph. There is one image goal for direction d_2 and each mode, and there are no pointing goals.
- Compute a $(1 + \frac{1}{2}\varepsilon)$-approximate plan for the SATELLITE task.
- Return the set of vertices corresponding to satellites whose instruments are used for making images in the computed plan.

Clearly, this is a polynomial-time algorithm. If it did not return a vertex cover, there would be an edge $\{u, v\}$ such that neither the satellite for u nor the satellite for v is used for taking an image. This implies that no image in the mode corresponding to $\{u, v\}$ was taken, a contradiction. Thus, the algorithm returns a vertex cover.

If V^* is an optimal vertex cover, then there exists a plan for the corresponding SATELLITE task of length $|E| + 3|V^*|$, namely the plan that powers on the instruments whose satellites correspond to V^*, calibrates those instruments, turns the satellites to d_2, then takes all images. Therefore, the length of the approximate plan is at most $(1 + \frac{1}{2}\varepsilon)(|E| + 3|V^*|)$. Of those actions, at least $|E|$ must be image-taking actions. Moreover, the number of satellites used for taking images in the plan cannot be higher than one third of the total number of non-image-taking actions, since using a satellite for taking images requires three set-up actions (powering on the instrument, calibrating it, and turning the satellite). Therefore, the number of instruments used is at most $\frac{1}{3}((1+\frac{1}{2}\varepsilon)(|E|+3|V^*|)-|E|) = |V^*|+\frac{1}{2}\varepsilon(\frac{1}{3}|E|+|V^*|) \leq |V^*| + \frac{1}{2}\varepsilon(|V^*| + |V^*|) = (1 + \varepsilon)|V^*|$ as required. The inequality is due to the fact that we must have $|E| \leq 3|V^*|$ as shown in Theorem 14. ■

Like for the other domains, this hardness result already holds for a fairly restricted class of SATELLITE tasks, namely those with only two pointing directions and one instrument per satellite. In fact, we could easily have used only a single pointing direction, but the existing SATELLITE benchmarks obey the restriction that no images are ever required of directions that serve as calibration targets (presumably because calibration targets are "known" objects, of which we do not require images). Our proof follows this restriction in case it is considered integral to the SATELLITE domain rather than incidental to the benchmark set.

Instead of 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.

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 can be approximated within a constant performance ratio 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 keep us from coming up with c-approximating algorithms for any constant c, starting with the ROVERS domain.

Planning domain 23 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. Including these would require many more details without providing much further insight. Obviously, our hardness results carry over to the unrestricted domain. Using the original PDDL definition of ROVERS (Long & Fox 2003), it is quite easy to see that the following (fairly simple) result carries over to the unrestricted domain as well.

Theorem 24 ROVERS $\in$ poly-APX
ROVERS $\in$ **poly-APX**.
Proof: It is sufficient to show that there exists a polynomial-time planning algorithm for the domain. Clearly, any solvable ROVERS task can be solved by addressing the individual goals one at a time, which is easy (details omitted). ∎

The following hardness result is more interesting.

Theorem 25 ROVERS $\notin$ APX
ROVERS $\notin$ **APX**, *unless* $\mathbf{P} = \mathbf{NP}$. *This is already true for the restricted* SIMPLEROVERS *domain.*
Proof: We show that from a c-approximating algorithm for SIMPLEROVERS, we can construct a c'-approximating algorithm for MINIMUM SET COVER for some value of c', which implies $\mathbf{P} = \mathbf{NP}$. Given a set S and collection of subsets C, the set covering algorithm works as follows:

- Construct a ROVERS task with one soil sample waypoint w_i for each item $s_i \in S$ and one rover r_j for each subset $C_j \in C$. Moreover, there are additional waypoints l_k for all $k \in \{0, \dots, M\}$ where $M := 5|S|$. In all rover roadmaps, waypoint l_k is connected to waypoint l_{k+1} for all $k \in \{0, \dots, M-1\}$. Moreover, for rover r_j, waypoint l_M is connected to w_i iff $s_i \in C_j$. All rovers are additionally located at l_0, which is the lander location.
- Compute a c-approximate plan for the SIMPLEROVERS task.
- Return the set cover consisting of those subsets in C for which the corresponding rover transmits any soil data in the plan.

This algorithm indeed returns a set cover: For each element $s \in S$, only rovers which correspond to those sets containing s could possibly gather (and thus communicate) the soil data. Therefore, by including sets for all rovers that communicate any soil data, each element of S is included in some set in the returned set cover.

We now need to prove that the performance ratio of the returned set cover is bounded by a constant. To do so, we first estimate the length of an optimal plan given the cardinality of an optimal set cover. Let C^* be an optimal set cover. Then the following plan solves the task:

- Move all rovers corresponding to sets in C^* to l_M.
- For each $s \in S$, pick some set $C_j \in C^*$ which includes s. With rover r_j, move to w_s, sample the soil, transmit the data, empty the store, and move back to l_M.

The first step requires a total of $|C^*| \cdot M$ actions, the second a total of $5|S| = M$ actions. Therefore, an optimal plan cannot be longer than $(|C^*| + 1)M \leq 2|C^*|M$ actions, ignoring the trivial instance where $C^* = \emptyset$. The c-approximate plan thus contains at most $2c|C^*|M$ steps. In such a plan, no more than $2c|C^*|$ rovers can gather soil data, as each rover requires more than M actions to move to a soil sample waypoint. Thus, the returned set cover has cardinality at most $2c|C^*|$, proving that the algorithm is $2c$-approximating, which concludes the proof. ∎

We presented this proof first because the other two are very similar in spirit, but somewhat more involved technically. In both cases, like in the preceding proof, the key idea is that choosing a certain subset 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 subsets chosen. 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 26 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. Since 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, we will in the following eliminate minor locations by assuming that the footpath graph is a connected graph over the major locations, counting walking actions twice when computing the cost of a plan. We will also refer to major locations simply as "locations". We first show that finding a plan in the DRIVERLOG domain without strict quality bounds is easy.

Theorem 27 DRIVERLOG $\in$ poly-APX
DRIVERLOG $\in$ **poly-APX**.
Proof: Consider the following algorithm:

- For each unsatisfied package goal, walk some driver to some truck, board the truck, drive to the initial package location, load the package, drive to the goal package location, unload the package, debark.
- For each unsatisfied truck goal, walk some driver to the truck, board the truck, drive to the goal truck location, debark.
- For each unsatisfied driver goal, walk the driver to the driver goal location.

The algorithm runs in polynomial time, generating a solution if the task is solvable and failing otherwise. The latter can only happen for tasks that lack trucks or drivers. ■

We now show that it is hard to find good plans in the DRIVERLOG domain.

Theorem 28 DRIVERLOG $\notin$ APX
DRIVERLOG $\notin$ **APX**, *unless* **P** $=$ **NP**. *This is true even when there is only a single driver, and goals are only defined for packages.*

Proof: Similar to the SIMPLEROVERS proof, we show that from a c-approximating algorithm for DRIVERLOG, we can construct a c'-approximating algorithm for MINIMUM SET COVER. Given a set S and collection of subsets C, the set covering algorithm works as follows:

- Check whether there exists any set cover at all, and stop if this is not the case.
- Let $M := 8|S|$. Construct a DRIVERLOG task with one location v^0 (called *central location*), locations $v^k_{C_j}$ for all subsets $C_j \in C$ and all $k \in \{1, \dots, M\}$ (called *subset path locations*, or *subset locations* if $k = M$), one location $v_{C_j,s}$ for all subsets $C_j \in C$ and elements $s \in C_j$ (called *subset element locations*), and locations v^-_s and v^+_s for each element $s \in S$ (called *element start locations* and *element goal locations*).
  For all $C_j \in C$, $s \in C_j$ and $k \in \{1, \dots, M-1\}$, the footpath graph contains edges from v^0 to $v^1_{C_j}$, from $v^k_{C_j}$ to $v^{k+1}_{C_j}$, and from $v_{C_j,s}$ to v^-_s.
  For all $C_j \in C$ and $s \in C_j$, the road graph contains an edge from $v^M_{C_j}$ to $v_{C_j,s}$. For all $s \in S$, it contains an edge from v^-_s to v^+_s.
  There is a single driver, originally located at the central location, with no defined goal.
  Each subset location and each element start location is the initial location of some truck, also with no defined goal.
  For all elements $s \in S$, there is a package initially located at the element start location v^-_s with goal location v^+_s.
  Finally, there is an additional *remote location*, connected to all other locations through very long roads and footpaths such that the distance between the remote location and the locations introduced before is $\lceil c \cdot 4M|C|+1 \rceil$. (We only include this location to make the graphs connected, as required by the domain definition.)
- Compute a c-approximating plan for the DRIVERLOG task.
- Return the set cover consisting of those subsets corresponding to subset locations visited by the driver within the plan.

If $C^* \subseteq C$ is a set cover, then the following plan is a solution to the DRIVERLOG task:

- For each set $C_j \in C^*$:
  - Walk from central location v^0 to subset location $v^M_{C_j}$.
  - For each $s \in C_j$ such that the package initially located at v^-_s is not yet at its goal location, board the truck at the subset location $v^M_{C_j}$, drive to the subset element location $v_{C_j,s}$, debark, walk to the element start location v^-_s, board the truck there, load the package, drive to the element goal location v^+_s, unload the package, drive back to v^-_s, debark, walk back to $v_{C_j,s}$, board the truck, drive back to $v^M_{C_j}$ and debark.
  - If this is not the last set in C^* to consider, walk back from $v^M_{C_j}$ to v^0.

To estimate the length of the plan, note that the inner for loop is executed exactly once for each element in S. Taking into account that walking costs two actions, this means that a total of $16|S| = 2M$ actions are required in the inner loop, while $2M \cdot (2|C^*|-1)$ actions are required for walking from the center location to some subset location $|C^*|$ times and back to the center $|C^*| - 1$ times. Added together, the total plan length is thus $4M|C^*|$.

The approximate plan thus contains at most $c \cdot 4M|C^*|$ actions. In particular, as $|C^*| \leq |C|$, this implies that the remote location is not visited by the plan. For the rest of the discussion, we can therefore pretend that the remote location and the paths leading up to it do not exist.

In order to satisfy any goal, the driver must walk to some truck, which requires walking to some subset location for a subset $C_j \in C$. Moreover, once this location is reached, the only goals that can be accomplished without walking back to the central location are those related to the elements of C_j. The driver must thus eventually return to the central location, and due to the placement of the trucks and road layout, all trucks will be at the same locations as initially at that point. Therefore, the driver will need to walk to the subset location for *each* subset in some set cover, each time requiring at least $2M$ actions (not even considering the cost of walking back). In a plan of length at most $c \cdot 4M|C^*|$, the number of such locations visited is therefore at most $2c|C^*|$, which implies that we have designed a $2c$-approximating algorithm for MINIMUM SET COVER, completing the proof. ■

The final domain we consider, GRID, is another variation of the transportation theme.

Planning domain 29 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 the analysis of the GRID domain by Helmert (2003), we know that GRID $\in$ **poly-APX**. 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 provide a new hardness proof here.

Theorem 30 GRID $\notin$ APX
GRID $\notin$ **APX**, *unless* **P** $=$ **NP**.
Proof: We show that from a c-approximating algorithm for GRID, we can construct a c'-approximating algorithm for MINIMUM SET COVER. Given a set $S = \{s_1, \dots, s_n\}$ and collection of subsets $C = \{C_1, \dots, C_k\}$, the set covering algorithm works as follows:

- Let $w := \max\{n, k\}$ and $M := (2w + 10)|S|$. Construct a GRID task with a grid of width $w + 1$ and height $M + 5$; we refer to the locations of the grid as pairs $(x, y) \in \{0, \dots, w\} \times \{0, \dots, M + 4\}$. The set of shapes is $S \cup C \cup \{\bot\}$, where $\bot$ is a shape used for keys that cannot open any location. The robot is initially located at $(0, M)$. For each subset $C_j \in C$, there is one key k_j with key shape C_j initially located at $(j, 0)$. These keys are called *subset keys.* For each element $s_i \in C_j$, there is one key $k_{j,i}$ with shape s_i initially located at $(j, M + 1)$. These keys are called *element keys*, and their initial locations are called *subset doors*. Each subset door $(j, M + 1)$ is initially locked, with lock shape C_j.
  For each element $s_i \in S$, there is one key k_i^+ of key shape $\bot$ initially located at $(i, M+3)$ with goal location $(i, M+4)$. These keys are called *goal keys*. Their initial locations $(i, M + 3)$ are locked with locks of key shape s_i and are called *element doors*.
- Compute a c-approximating plan for the GRID task.
- Return the set cover consisting of all subsets that correspond to subset keys used to open a door within the plan.

To deliver the goal key corresponding to a set element, the initial goal key location (element door) must be opened with a matching element key. Element keys are locked behind subset doors for those subsets containing this element. Opening these requires fetching a subset key from the bottom of the grid. Therefore, the subsets returned by the algorithm indeed form a set cover.

Due to the fact that the distance between subset key locations and subset locks is very large and only one key can be carried at a time, fetching the subset keys is the bottleneck of any plan, and good plans will make use of as few subset keys as possible.

If $C^* \subseteq C$ is a set cover, then the following plan is a solution to the GRID task:

- For each subset $C_j \in C^*$, do the following:
  1. Move from $(0, M)$ to $(j, 0)$, pick up the subset key for C_j, move to (j, M), open location $(j, M + 1)$, move to $(j, M + 1)$, drop the key.
  2. For each element $s_i \in C_j$ for which the goal key has not yet been moved to its goal location, pick up the element key $k_{j,i}$, move from $(j, M + 1)$ to $(i, M + 2)$, open location $(i, M + 3)$, move to $(i, M + 3)$, switch key $k_{j,i}$ with the goal key k_i^+, move to $(i, M + 4)$, drop the key, move back to $(j, M + 1)$.
  3. Move back from $(j, M + 1)$ to $(0, M)$.

For each iteration of step 1., no more than $2M + w + 4$ actions are required (note that $j \leq w$), so the sum over all iterations is bounded by $(2M + w + 4)|C^*|$. For each goal key in step 2., no more than $2w + 10$ steps are required (note that $i, j \leq w$), summing to $(2w + 10)|S|$ over all iterations. Finally, for each iteration of step 3., at most $w + 1$ steps are required, for a sum of $(w + 1)|C^*|$ over all iterations. We thus obtain a total bound of $(2M + 2w + 5)|C^*| + (2w + 10)|S| \leq 3M|C^*| + M \leq 4M|C^*|$, unless $S = C^* = \emptyset$, a trivial case that we shall ignore.

Given that a plan no longer than $4M|C^*|$ steps exists, the generated plan has at most $c \cdot 4M|C^*|$ actions. Using a subset key requires more than $2M$ actions, so no more than $2c|C^*|$ keys are used. Thus, the MINIMUM SET COVER algorithm is $2c$-approximating. ∎

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.

Summary and conclusion

The approximation properties of the benchmark domains are summarized in Fig. 2. 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.[3] However, as we pointed out when discussing the GRID domain, there are special cases of the competition domains that do admit polynomial-time approximation schemes.

$\in$ **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 2: Improved classification results. For PIPESWORLD, only hardness is known; membership in **NPO** is open.

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 Helmert's (2003) results 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" (Slaney & Thiébaux 2001, on BLOCKSWORLD). In comparison to Slaney and Thiébaux's effort, at least two more steps are required for most of the benchmarks: identifying phase transition regions to understand where the hard tasks are, and having (domain-dependent) optimal solvers available to be able to evaluate solution quality of non-optimal planners.

[3] 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. (1999). Similarly, we cannot have planning domains in the class **exp-APX** $\setminus$ **poly-APX**.

References

Ausiello, G.; Crescenzi, P.; Gambosi, G.; Kann, V.; Marchetti-Spaccamela, A.; and Protasi, M. 1999. *Complexity and Approximation*. Springer-Verlag.

Bacchus, F. 2001. The AIPS'00 planning competition. *AI Magazine* 22(3):47–56.

Gupta, N., and Nau, D. S. 1992. On the complexity of blocks-world planning. *Artificial Intelligence* 56(2–3):223–254.

Helmert, M. 2003. Complexity results for standard benchmark domains in planning. *Artificial Intelligence* 143(2):219–262.

Helmert, M. 2006. New complexity results for classical planning benchmarks. In Long, D., and Smith, S., eds., *Proceedings of the Fifteenth International Conference on Automated Planning and Scheduling (ICAPS 2006)*. AAAI Press. To appear.

Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: An overview. *Journal of Artificial Intelligence Research* 24:519–579.

Long, D., and Fox, M. 2003. The 3rd International Planning Competition: Results and analysis. *Journal of Artificial Intelligence Research* 20:1–59.

McDermott, D. 2000. The 1998 AI Planning Systems competition. *AI Magazine* 21(2):35–55.

Papadimitriou, C. H., and Yannakakis, M. 1991. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences* 43:425–440.

Selman, B. 1994. Near-optimal plans, tractability, and reactivity. In Doyle, J.; Sandewall, E.; and Torasso, P., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR'94)*. Morgan Kaufmann. 521–529.

Slaney, J., and Thiébaux, S. 2001. Blocks World revisited. *Artificial Intelligence* 125:119–153.