To appear in the *Journal of Experimental & Theoretical Artificial Intelligence*
Vol. 00, No. 00, Month 20XX, 1–28

Outer Entanglements: A General Heuristic Technique for Improving the Efficiency of Planning Algorithms

Lukáš Chrpa[a,b,*] and Mauro Vallati[c] and Thomas Leo McCluskey[c]

[a]*Department of Computer Science, Czech Technical University in Prague, Prague Czech Republic*
[b]*Department of Theoretical Computer Science and Mathematical Logic, Charles University in Prague, Prague Czech Republic*
[c]*Department of Informatics, University of Huddersfield, Huddersfield, UK*

(*Received 00 Month 20XX; final version received 00 Month 20XX*)

In the field of Automated Planning, a central research focus is on domain-independent planning engines which accept planning tasks (domain models and problem descriptions) in a standard language such as PDDL, and return solution plans. Performance of planning engines can be improved by gathering additional heuristic knowledge about specific planning domain models/tasks (e.g. control rules) which can guide the search for a solution plan. However, there is no convention on the syntax or semantics of such additional knowledge, and thus domain-independent planning engines cannot benefit from it. Using techniques to transform the given planning task to incorporate additional heuristic knowledge, however, while keeping to the same input language, can overcome this issue to some degree.

In this paper we present *outer entanglements*, that are relations between planning operators and predicates whose instances are present in the initial state or the goal. These relations are used to restrict "entangled" operator instances such that only "entangled" predicate instances present in the initial state or the goal are considered (e.g., picking up packages only at their initial locations).

The contribution of this paper is to provide an in depth analysis and evaluation of outer entanglements, including theoretical aspects such as complexity results. Also, it includes an extensive empirical study using competition benchmarks and state-of-the-art planning engines illustrating the effectiveness of using Outer Entanglements as heuristics for improving the efficiency of planning algorithms.

Keywords: Classical Planning; Outer Entanglements; Domain Reformulation; State Space Pruning

1. Introduction

Automated planning is an important research area of Artificial Intelligence (AI) where an autonomous entity (e.g. a robot) reasons about the way it can act in order to achieve its goals. AI planning has therefore a great potential for applications where a certain level of autonomy is required such as in the Deep Space 1 mission (Bernard et al., 2000).

Automated Planning involves combinatorial search that, roughly speaking, examines numerous combinations of action sequences in order to find a solution plan (an action sequence transforming the environment from the initial state to some goal state). In the last few decades, there has been a great deal of activity in the research commu-

*Corresponding author. Email: chrpaluk@fel.cvut.cz

nity designing planning techniques and planning engines. The International Planning Competition (IPC)[1] has been staged regularly since 1998 and is increasingly attracting the attention of the AI planning community (Vallati et al., 2015). Thanks to the IPC we have many advanced planning engines, and PDDL (Ghallab et al., 1998) that is a standardized language family for describing planning tasks and standard benchmarks for measuring planners' performance. Along with those planning engines, many novel planning techniques have been proposed, such as heuristic search (Bonet and Geffner, 1999), translating planning tasks into SAT (Kautz and Selman, 1992) just to mention a few.

The performance of planning engines can be improved by extracting and exploiting Domain Control Knowledge (DCK), i.e., additional knowledge about planning tasks, for instance, in the form of Control Rules (Minton and Carbonell, 1987) or Decision Trees (de la Rosa et al., 2011). DCK, roughly speaking, provides a guidance for planning engines, so they can find solution plans more quickly. The usefulness of learning and exploiting DCK in planning has been demonstrated by Yoon et al. (Yoon et al., 2008) whose approach that learns DCK from relaxed plans (obtained by solving planning tasks while omitting negative effects of actions) won the best learner award at IPC 2008. However, these types of knowledge are often tied to specific planning engines using a planner-specific language (e.g. as with TALPlanner (Kvarnström and Doherty, 2000)) or a convention on extending the input language to take advantage of them, so that planning engines can maintain some level of domain-independence. On the contrary, knowledge which can be directly encoded into the standard definition language (such as PDDL) is planner-independent, so a standard planning engine can straightforwardly exploit it. For example, action-centric DCK can be compiled into PDDL (Baier et al., 2007). The best known specific type of DCK, macro-operators ("macros"), which encapsulate sequences of operators, can be encoded as normal planning operators, so they can be exploited in a planner-independent way (Chrpa, 2010; Korf, 1985; McCluskey and Porteous, 1997; Newton et al., 2007). Abstracting planning tasks by their reformulation in order to reveal their hierarchical structures can mitigate "accidental complexity" of their domain models[2] (Haslum, 2007).

Beside macros, another type of domain-independent DCK are *Entanglements* (Chrpa and Barták, 2009; Chrpa and McCluskey, 2012), which represent relations between planning operators and predicates, aimed at eliminating unpromising alternatives in a planning engine's search space. *Inner Entanglements* (Chrpa and McCluskey, 2012) are relations between pairs of operators and predicates which capture exclusivity of predicate achievement or requirement between the given operators.

The vast majority of planning engines such as FF (Hoffmann and Nebel, 2001) or those built on top of the Fast Downward planner (Helmert, 2006) performs a pre-processing step –called grounding– in which they compute all atoms and actions that can be reachable from a given initial state, mutual exclusivity of pair of atoms (those that cannot be both present in any reachable state), and structures such as Causal Graph (Knoblock, 1994). Large grounded representation, i.e., a large number of atoms describing the environment and actions one can perform, poses high CPU time and memory requirements for traditional domain-independent planning engines. Noteworthy, use of some types of DCK such as macros or inner entanglements often exacerbate the problem of large representation, since, for example, macros have more arguments than ordinary operators and thus have much more instances.

Outer Entanglements (Chrpa and Barták, 2009; Chrpa and McCluskey, 2012), we focus

[1] http://ipc.icaps-conference.org

[2] "accidental complexity of domain models" means their inefficient encodings decreasing performance of planning engines

on in this paper, aim at reducing the size of problem representation by eliminating possibly unpromising grounded actions. Outer entanglements are relations between planning operators and predicates whose instances are present in the initial state or the goal. These relations capture a useful knowledge that some planning operators are needed only to modify initial situation (e.g., picking up a package at its initial location) or achieve goal situation (e.g., delivering a package to its goal location). Hence, only limited numbers of instances of "entangled" operators has to be considered in the planning tasks which reduces the size and complexity of the state space planning engines have to search through. In particular, eliminating some actions (operators' instances) often makes some atoms unreachable (e.g. a package cannot be in other than initial or goal location). Smaller problem representation increases efficiency of pre-processing planning engines perform. Consequently, state space is smaller too, as some unpromising alternatives in it are eliminated. Hence, planning engines have to make less effort to search through such a reduced state space in order to find solution plans.

Outer entanglements can be encoded in planning tasks, effectively re-formulating them, and thus they are planner-independent. Deciding whether a given outer entanglement holds in a planning task is PSPACE-complete, the same complexity class as the problem of solving the planning task, hence finding outer entanglements for a planning task is generally as hard as solving the task. On the other hand, outer entanglements are often domain-specific rather than task specific. Therefore, we have developed an approximate, heuristic method that is used to learn outer entanglements from training plans, solution plans of simpler tasks in a given domain. One of the advantages of the learning approach is that we do not have to know *why* a set of outer entanglements holds in a given domain. Arguably, the nature of outer entanglements differs per different domain models as discussed in Section 4.4. On the other hand, the heuristic method follows the premise that outer entanglements generalize well, i.e., if a set of outer entanglements holds for a set of (simpler) planning tasks, then it also holds for the whole class of the planning tasks sharing the same domain model. Being a *heuristic* method, it is possible that the reformulation results in incorrect choices, and there may even be instances where is preferable to use the original search space: we investigate this issue in our experimental evaluation.

Initial work on outer entanglements has been reported in a series of shorter papers detailing the discovery, use and effectiveness of outer entanglements. In this paper, we integrate and extend previous work, with:

- encodings of outer entanglements (Chrpa and Barták, 2009) including formal proofs of their correctness;
- a collected summary of the known complexity results (Chrpa et al., 2012), and trivial cases where entanglements hold (Chrpa and Barták, 2009);
- case studies in which we investigate what outer entanglements hold, and under what conditions;
- an analysis of the potential impact of outer entanglements on the planning process;
- an approximation method for extracting outer entanglements (Chrpa and Barták, 2009);
- an extensive empirical study of the impact of outer entanglements in the planning process using all the domains from the 6th and 7th IPC's learning track[3], and 7 state-of-the-art planning engines based on very different principles.

The main empirical findings from this paper are that the use of outer entanglements

[3]Learning track benchmarks are more natural, since entanglements extraction phase can be understood as a learning process

improve the planning process, often remarkably, through the planner and domain model combinations we experimented with. The results demonstrate that the learning method, despite being heuristically-based, often learns a useful set of outer entanglements that generalize well for non-training planning tasks. Also, the thorough experimental analysis provides invaluable lessons from which domain engineers can learn how to extract effective and efficient sets of outer entanglements for their domain models.

The paper is organised as follows. After discussing related work, classical planning is introduced, the required terminology is defined and a BlocksWorld domain running example is introduced. Then, outer entanglements are defined, complexity of their identification and case studies referring to easy and possibly hard instances of outer entanglement identification are discussed. After that a reformulation approach enforcing outer entanglements in a planner-independent way (i.e., any standard planner can make use of outer entanglements) is presented, and an approximation algorithm for learning outer entanglements from training plans is introduced. Empirical analysis of impact of outer entanglements in the planning process is provided after that and then, finally, we conclude and present some future avenues of research.

2. Related Work

Generating DCK which can be exploited by planning engines dates back into 1970s when systems such as REFLECT (Dawson and Siklóssy, 1977) were developed. Macros are one of the best known type of DCK in classical planning, because they can be encoded as normal planning operators and thus easily added into planning domain models. Macro-FF CA-ED version (Botea et al., 2005), which learns macros through analysis of relations between static predicates, and Wizard (Coles et al., 2007), which learns macros by genetic algorithms, are good examples of planner-independent macro learning systems.

While the main disadvantage of using macros is the risk of a significant increase of the branching factor during searching, there are several techniques which are used for reducing the branching factor. One way is to combine macros with another learning technique, specifically aimed at pruning unpromising instances of macros, such as in McCluskey's early work (McCluskey, 1987). The complementary technique here created 'chunks' - learnt relations between initial states and operator preconditions, similar to entanglements by init (a subtype of outer entanglements). The method required a specially extended planner, however, and was aimed at plan space search rather than state space search. Recently, it has been shown that there is a synergy between macros and outer entanglements, in other words, outer entanglements can prune unpromising instances of macros as well as provide heuristics for their generation, which has been demonstrated by MUM (Chrpa et al., 2014) and OMA (Chrpa et al., 2015b), where the former learns macros from training plans while the latter generates macros online (without training plans).

Determining action relevance is an important branch of research, which reduces the number of instances of planning operators planning engines have to deal with. The FF planner (Hoffmann and Nebel, 2001) instantiates only actions appearing at some level of relaxed Planning Graph. FastDownward (Helmert, 2006) uses the "reach-one-goal" idea, i.e., achieves the goals of the planning task consecutively, where the solver focuses only on such actions that may be relevant for a particular goal. Other work focusing on cost-optimal SAS+ planning (Coles and Coles, 2010) prunes irrelevant actions (e.g. actions changing a value of a variable having no dependants from a goal value) or exploits "tunnel macro-actions" (i.e. if a certain action is executed then there is no other choice than

to execute specific actions forming the "tunnel"). Haslum and Jonsson (2000) proposed a technique for pruning actions whose effects can be acquired by executing a sequence of different actions. Scholz (2004) proposed a method for determining action relevance on problems with acyclic causal graphs. Scholz's method has recently been extended to cover problems with non-acyclic causal graphs (Haslum et al., 2013). "Expansion Core" is a method which in a node expansion phase (in A* search) restricts on relevant Domain Transition Graphs rather than all of them (Chen and Yao, 2009). The idea of "expansion cores" is extended into *strong stubborn sets* that guarantees stronger pruning than expansion cores (Wehrle et al., 2013). Recently, *factored planning* (Brafman and Domshlak, 2013) has been exploited for extending the strong stubborn sets approach and introducing *decoupled strong stubborn sets* that in some cases provide exponentially stronger reductions of the problem (Gnad et al., 2016). In contrast to aforementioned techniques, outer entanglements focus on pruning operator instances that either do not require initial atoms, or do not achieve goal atoms and thus are complementary to aforementioned techniques.

3. Preliminaries

This section is devoted to introducing the terminology that will be used throughout the paper.

3.1. *Classical Planning*

Classical planning is concerned with finding a (partially or totally ordered) sequence of actions transforming the static, deterministic and fully observable environment from the given initial state to a desired goal state (Fox and Long, 2003; Ghallab et al., 2004).

In the classical representation, a *planning task* consists of a *planning domain model* and a *planning problem*, where the planning domain model describes the environment and defines planning operators while the planning problem defines concrete objects, an initial state and a set of goals. The environment is described by *predicates* that are specified via a unique identifier and terms (variable symbols or constants). For example, a predicate `at(?t ?p)`, where `at` is a unique identifier, and `?t` and `?p` are variable symbols, denotes that a truck `?t` is at location `?p`. Predicates thus capture general relations between objects.

Definition 1: A **planning task** is a pair $\Pi = (Dom_\Pi, Prob_\Pi)$ where a **planning domain model** $Dom_\Pi = (Preds_\Pi, Ops_\Pi)$ is a pair consisting of a finite set of predicates $Preds_\Pi$ and planning operators Ops_Π, and a **planning problem** $Prob_\Pi = (Objs_\Pi, I_\Pi, G_\Pi)$ is a triple consisting of a finite set of objects $Objs_\Pi$, initial state I_Π and goal G_Π.

Let ats_Π be the set of all **atoms** that are formed from the predicates $Preds_\Pi$ by substituting the objects $Objs_\Pi$ for the predicates' arguments. In other words, an atom is an **instance** of a predicate (in the rest of the paper when we use the term instance, we mean an instance that is fully *grounded*). A **State** is a subset of ats_Π, and the **initial state** I_Π is a distinguished state. The **goal** G_Π is a non-empty subset of ats_Π, and a **goal state** is any state that contains the goal G_Π.

Notice that the semantics of state reflects the full observability of the environment. That is, that for a state s, atoms present in s are assumed to be true in s, while atoms not present in s are assumed to be false in s.

Planning operators are "modifiers" of the environment. They consist of *preconditions*, i.e., what must hold prior an operators' application, and *effects*, i.e., what is changed after operators' application. *Actions* are instances of planning operators, i.e., operators' arguments as well as corresponding variable symbols in operators' preconditions and effects are substituted by objects (constants). Planning operators capture general types of activities that can be performed. Similarly to predicates that can be instantiated to atoms to capture given relations between concrete objects, planning operators can be instantiated to actions to capture given activities between concrete objects.

Definition 2: A **planning operator** is a tuple $o = (name(o), pre(o), \mathit{eff}^-(o), \mathit{eff}^+(o))$ is specified such that $name(o) = op\_name(x_1, \ldots, x_k)$, where op_name is a unique identifier and $x_1, \ldots x_k$ are all the variable symbols (arguments) appearing in the operator, $pre(o)$ is a set of predicates representing o's precondition, $\mathit{eff}^-(o)$ and $\mathit{eff}^+(o)$ are sets of predicates representing o's negative and positive effects. Notice that all the predicates in o's definition must be defined in $Preds_\Pi$ of the corresponding domain model. **Actions** are instances of planning operators that are formed by substituting objects, which are defined in a planning problem, for operators' arguments as well as for corresponding variable symbols in operators' preconditions and effects. An action $a = (pre(a), \mathit{eff}^-(a), \mathit{eff}^+(a))$ is **applicable** in a state s if and only if $pre(a) \subseteq s$. If possible, application of a in s, denoted as $\gamma(s, a)$, results in a state $(s \setminus \mathit{eff}^-(a)) \cup \mathit{eff}^+(a)$.

A solution of a planning task is a sequence of actions transforming the environment from the given initial state to a goal state.

Definition 3: A **solution plan**, or shortly **plan**, of a planning task $\Pi = (Dom_\Pi, Prob_\Pi)$, where $Prob_\Pi = (Objs_\Pi, I_\Pi, G_\Pi)$, is a sequence of actions $a_1, \ldots, a_n$ (all actions are instances of planning operators defined in Dom_Π) such that $G_\Pi \subseteq \gamma(\ldots\gamma(I_\Pi, a_1), \ldots, a_n)$.

3.2. *BlocksWorld Domain*

We briefly introduce a model representing the *BlocksWorld* domain (Slaney and Thiébaux, 2001), which is one of the best known planning domains, that will be used as a running example in the paper.

The BlocksWorld domain describes an environment where we have a finite number of blocks, one table with unlimited space, and one robotic hand. A block can be either stacked on another block, placed on the table or held by the robotic hand. No block can be stacked on more than one block at the same time as well as no more than one block can be stacked on a block at the same time. The robotic hand can hold at most one block. The BlocksWorld domain model consists of four operators: pickup(?x) refers to a situation when the robotic hand picks-up a block ?x from the table, putdown(?x) refers to a situation when the robotic hand puts down the block ?x it is holding to the table, unstack(?x ?y) refers to a situation when the robotic hand unstacks a "clear" block ?x from a block ?y, and stack(?x ?y) refers to a situation when the robotic hand stacks the block ?x it is holding to a "clear" block ?y. As mentioned before, planning operators are instantiated by substituting constants (objects) for variable symbols that appear in operators' definition. For example, putdown(?x) can be instantiated by substituting a, which refers to a concrete block "a", for ?x. We then obtain an action putdown(a) that requires the robotic hand to hold the block a, and the effect is that the block a is placed on the table, the block a is clear (no other block is stacked on it), and the hand no longer

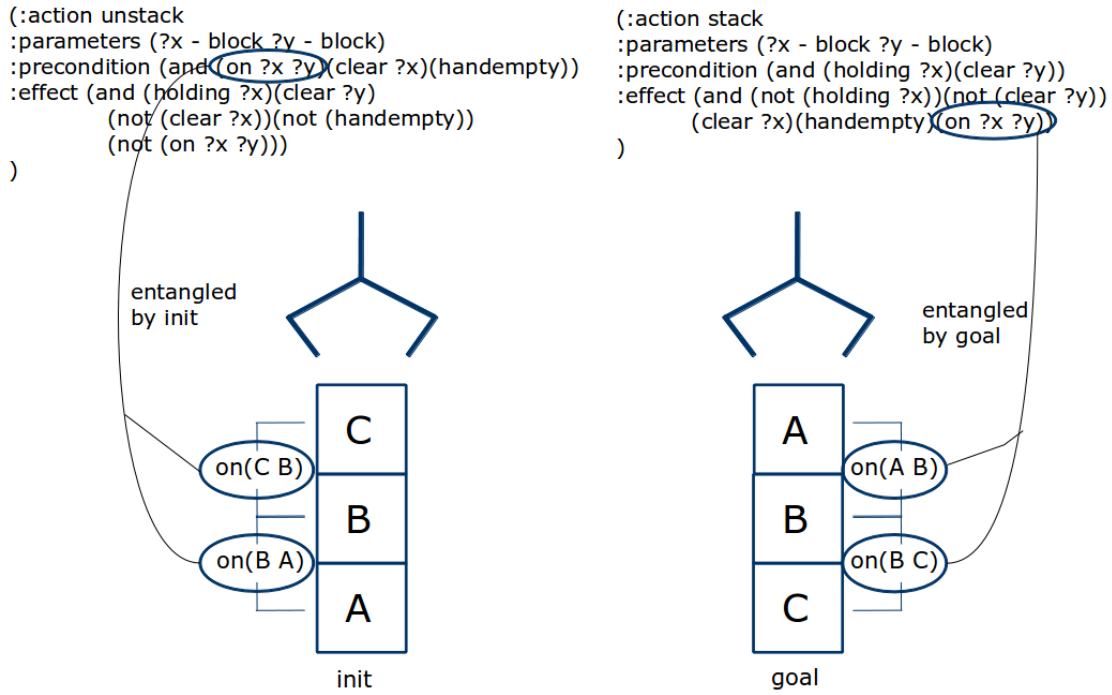

Figure 1. An illustrative example of outer entanglements. On the left hand side, unstack is entangled by init with on, and, on the right hand side, stack is entangled by goal with on.

holds it.

4. Outer Entanglements

This section formally introduces outer entanglements and provides theoretical analysis of them.

4.1. *Introduction*

Outer Entanglements are relations between planning operators and predicates whose instances are present in the initial state or the goal of some solution plan. In a BlocksWorld planning task there exists a solution plan where unstack actions may only unstack blocks from their initial positions (e.g., if on(a b) holds in the initial state, then unstack(a,b) ca be present in the plan) and stack actions may only stack blocks to their goal position. Notice that blocks can be temporarily put on the table (there are no space limits on the table). Formally speaking, an *entanglement by init* will capture that if an atom on(a b) is to be achieved for a corresponding instance of operator unstack(?x ?y) (unstack(a b)), then the atom is present in the initial state. Similarly, an *entanglement by goal* will capture that an atom on(b a) achieved by a corresponding instance of operator stack(?x ?y) (stack(b a)) is present in the goal. For illustration, see Figure 1. The outer entanglements relation, i.e., the entanglements by init and goal, are defined as follows.

Definition 4: Let Π be a planning task, where I_Π is the initial state and G_Π is the goal. Let o be a planning operator and p be a predicate defined in the domain model of Π. We say that operator o is **entangled by init (resp. goal)** with a predicate p in Π if and

only if $p \in pre(o)$ (resp. $p \in eff^+(o)$) and there exists π, a solution plan of Π, such that for every action $a \in \pi$ being an instance of o and for every atom p_{gnd} being an instance of p, it holds: $p_{gnd} \in pre(a) \Rightarrow p_{gnd} \in I_\Pi$ (resp. $p_{gnd} \in eff^+(a) \Rightarrow p_{gnd} \in G_\Pi$). We also say that π **satisfies** the entanglement (by init or goal) conditions.
Henceforth, entanglements by init and goal are denoted as **outer entanglements**.

Notice that the definition allows for initial atoms to be deleted / re-achieved during the plan, without falsifying the entanglement relationship with an operator instance requiring such an atom later in the plan.

Also, a single outer entanglement requires only the existence of one solution plan of the given planning task where the entanglement conditions are met. However, different entanglements might hold in different solution plans. It is practical to consider a set of outer entanglements for a planning task rather than a single one, which means that there must exist a solution plan in which all entanglements from the set hold. Also, in practice, outer entanglements are domain- or class of tasks-specific rather than task-specific. The above definition can be extended to reflect these aspects.

Definition 5: Let Π be a planning task. We say that a set of outer entanglements ENT_Π holds for Π if and only if there exists a solution plan of Π in which all the entanglements from ENT_Π hold.
Similarly, $ENT_\mathcal{P}$ holds for a set of planning tasks $\mathcal{P}$ if and only if $ENT_\mathcal{P} = \bigcap_{\Pi \in \mathcal{P}} ENT_\Pi$.

Both the aforementioned BlocksWorld-related outer entanglements hold for every BlocksWorld planning task with unlimited table space.

4.2. *Intractability of Deciding on Entanglements*

Landmark theory (Hoffmann et al., 2004) is a useful framework for studying structures of planning tasks. We will use a fragment of the landmark theory to prove intractability (PSPACE-completeness) of deciding whether a given outer entanglement holds in a given task. *Landmarks* are atoms which must be achieved at some point in every solution plan of a given planning task. *Action landmarks* are actions which must be applied at some point in every solution plan of a given planning task. Deciding whether atoms are landmarks as well as whether actions are action landmarks is PSPACE-complete (Hoffmann et al., 2004)

The intractability (PSPACE-completeness) of deciding whether a given outer entanglement holds is proved by the following theorem.

Theorem 1: *Let* Π *be a planning task,* o *be a planning operator and* p *a predicate defined in the domain model of* Π*. The problem of deciding whether* o *is entangled by init (resp. goal) with* p *in* Π *is PSPACE-complete.*

Proof. First, we show that the problem of deciding whether o is entangled by init (resp. goal) with p in Π belongs to the PSPACE class. To do this, we reformulate Π by encoding the given entanglement as described in Section 5.2. Hence, the decision problem of whether the given outer entanglement holds can be encoded as a planning task, i.e., the entanglement holds if and only if the reformulated planning task is solvable. We know that we can solve planning tasks in polynomial space, hence this decision problem belongs to PSPACE.

Next, we reduce (in polynomial time) the problem of deciding whether an action a is an action landmark in a planning task Π', which is PSPACE-complete, to the problem

of deciding whether o is entangled by init (resp. goal) with p in Π. Let o' be a planning operator defined in the domain model of Π' such that a is its instance. Let p be a predicate which has the same variable symbols (arguments) as operator o' and without loss of generality we assume that p is not defined in the domain model of Π'.

To decide that an action a is an action landmark in a planning task Π' by exploiting entanglements by init, we modify Π' in a following way. We create a planning operator o as a modification of o', that is, $o = (name(o'), pre(o') \cup \{p\}, \mathit{eff}^-(o'), \mathit{eff}^+(o'))$. Then, we create a planning operator o'' such that o'' has the same variable symbols (arguments) as o', $pre(o'') = \mathit{eff}^-(o'') = \emptyset$ and $\mathit{eff}^+(o'') = \{p\}$. Finally, we create a planning task Π by modifying Π' in such a way that o' is removed, and o and o'' are added into the set of operators, p is added into the set of predicates, and all the instances of p but one that has the same arguments as the action a are added into the initial state. We can see that o is entangled by init with p in Π if and only if a is not an action landmark in Π'. This is, because the entanglement tells us that there exists a solution plan of Π' where a is not present. The missing instance of p can be achieved by a corresponding instance of o'', so Π remains solvable even if a is an action landmark, however, the entanglement does not hold.

To decide that an action a is an action landmark in a planning task Π' by exploiting entanglements by goal, we modify Π' in a following way. We create a planning operator o as a modification of o', that is, $o = (name(o'), pre(o'), \mathit{eff}^-(o'), \mathit{eff}^+(o') \cup \{p\})$. Then, we create a planning task Π by modifying Π' in such a way that o' is removed, and o is added into the set of operators, p is added into the set of predicates, and all the instances of p but one that has the same arguments as the action a are added into the initial state and into the goal. We can see that o is entangled by goal with p in Π if and only if a is not an action landmark in Π'. This is as in the previous case, because the entanglement tells us that there exists a solution plan of Π' where a is not present. If a must be applied in all solution plans of Π', then the instance of p which is missing in the goal of Π is always achieved and thus the entanglement does not hold.

Clearly, modification of Π' in both cases is done in polynomial time. Hence, since the problem of deciding whether a is a landmark action in Π' is PSPACE-complete, the problem deciding whether modified o is entangled by init (resp. goal) with p in Π, which belongs to PSPACE, is PSPACE-complete as well.

□

Intractability of deciding whether a single outer entanglement holds for a given planning task implies intractability of deciding whether a set of outer entanglements holds for that task.

Corollary 1: *Let e_1 and e_2 be outer entanglements that hold in a planning task Π. The problem of deciding whether a set $\{e_1, e_2\}$ holds in Π is PSPACE-complete.*

Proof. Without loss of generality, let Π_{e_1} be a planning task obtained by reformulating Π considering e_1 (see Section 5.2). Then, the problem of deciding whether $\{e_1, e_2\}$ holds in Π is equivalent to the problem of deciding whether e_2 holds in Π_{e_1} which is PSPACE-complete. □

4.3. *Special Cases*

Despite the intractability, there are some cases where we can trivially identify outer entanglements (hereinafter referred as trivial outer entanglements). The following situations

refer to special cases where there is no way to violate outer entanglements in the planning process. However, trivial outer entanglements do not provide any new domain-specific information and hence we do not have to consider them in the reformulation.

For instance, if a predicate p is not achieved by any operator defined in the domain model (e.g. p is a static predicate), then any operator having p in its precondition is entangled by init with p.

Lemma 1: *Let* Π *be a planning task,* p *be a predicate and Ops be the set of planning operators defined in the domain model of* Π*. If* $p \notin eff^{+}(o)$ *for every* $o \in Ops$*, then for every* $o \in Ops$ *it is the case that* o *is entangled by init with* p *in* Π *if and only if* $p \in pre(o)$*.*

Another trivial example where outer entanglements can be easily determined is when all instances of a predicate are present in the initial state or the goal.

Lemma 2: *Let* Π *be a planning task,* I *its initial state and* G *its goal. Let Ops be the set of planning operators and* p *be a predicate defined in the domain model of* Π *such that all possible instances of* p *are present in* I *(resp.* G*). Then, an operator* $o \in Ops$ *is entangled by init (resp. goal) with* p *in* Π *if and only if* $p \in pre(o)$ *(resp.* $p \in eff^{+}(o)$*).*

4.4. *Case Studies*

In the BlocksWorld domain, which we also used as a running example, we can identify two non-trivial outer entanglements. The operator unstack is entangled by init with the predicate on and the operator stack is entangled by goal with on. A typical task (re-stacking the blocks from initial stacks to goal stacks) can be solved as follows. Blocks are unstacked from their initial positions and put down on the table until all the blocks are on the table. Then, we pick the blocks up and stack them on their goal positions (in the right order). We can see that both the entanglements hold for such solution plans. However, limiting the space on the table (in some modification of the domain) might invalidate these entanglements since due to lack of table space we might be forced to temporarily stack blocks on other blocks. Clearly, if the table space is greater or equal the number of blocks, or if goal stacks of blocks are reverted initial stacks of blocks, the entanglements still hold. However, in a general case deciding whether one or both entanglements hold for a given planning task having the modified domain model can be as hard as solving the task.

In the well-known ZenoTravel domain, which addresses the problem of transporting passengers by planes between cities, we can observe that the operator board is entangled by init with the predicate at and the operator debark is entangled by goal with at. A typical problem can be solved as follows. Each passenger can board an aircraft at the location of origin (if no aircraft is there, then it will arrive from a different location), then the aircraft flies to passenger's destination location where the passenger debarks. The entanglements hold in such solution plans. However, modifying the domain by constraining the locations where a particular aircraft can fly might invalidate the entanglements which will be the case when some passenger would have to change the aircraft at some (non-initial) location. Deciding whether the entanglements (or one of them) hold is easy in the modified domain, since for each passenger we can check whether there is a direct flight or not. Similarly, we can identify outer entanglements in the similar logistic-based domains.

Although we identified some domain-specific cases where identifying (non-trivial) outer

entanglements is easy, we are still missing a more general domain-independent approach for identifying a subclass of non-trivial outer entanglements in polynomial time. We believe that analysing structure of planning tasks (e.g. relations between planning operators, mutexes) can be useful for identifying some non-trivial outer entanglements.

5. The Use of Outer Entanglements to Speed-up the Planning Process

This section is devoted to practical use of outer entanglements.

5.1. *Motivation*

The reason for introducing the outer entanglement relation was to form the basis of a tool for eliminating potentially unnecessary instances of planning operators and thus reduce "overheads" for planning engines (Chrpa and Barták, 2009). This follows the observation that in many domains some operators are needed only to modify the initial state of the object, or achieve the goal state of the object. Given the example of the BlocksWorld domain (see Figure 1 in Section 4), we can see that since the unstack operator is entangled by init with the on predicate only the instances unstack(b a) and unstack(c b) are necessary, so we can prune the rest of unstack's instances because they are not necessary to find a solution plan. Similarly, we can see that since the stack operator is entangled by goal with the on predicate only the instances stack(a b) and stack(b c) are necessary, so we can prune the rest of stack's instances. Usefulness of such pruning can be demonstrated in the following way. Given n blocks, we can have at most $n \cdot (n-1)$ instances of stack or unstack (we do not consider instances when a block is unstacked from or stacked on itself – e.g stack(a a)). Considering both the entanglements, we can have at most $n-1$ instances of the stack or unstack operators. In summary, while in the original setting, the number of operators' instances grows quadratically with the number of blocks, considering outer entanglements reduces the growth of the number of operators' instances to linear. Consequently, the state space (i.e., the number of reachable states) can be also reduced. In the BlocksWorld case, a block cannot be stacked on another block unless it is its initial or goal configuration. Hence, when the given outer entanglements are applied there are only at most two (other) blocks a block can be stacked on at any point of the planning process. Otherwise (in the original encoding), a block can be stacked on $n-1$ other blocks (excluding itself) at any point of the planning process.

Outer entanglements are encoded by supplementary static predicates that are added into preconditions of operators involved in the outer entanglement relation. Most of existing planning engines generate operators' instances in preprocessing, i.e., they perform grounding. Static predicates are only useful at this stage; they are useful in filtering unreachable operators' instances[4], however, static predicates do not provide any valuable information in search, so planners are compiling them away after grounding. Hence, introducing supplementary static predicates does not increase the number of atoms planners have to deal with during the search. Reducing the number of actions planners have to consider during the search reduces the branching factor and thus "narrows" the search space. Moreover, memory requirements for planners can be often considerably lowered.

However, outer entanglements might cause some actions to become irreversible. For example, if we allow unstacking blocks only from their initial positions (captured by the entanglement by init mentioned before), then we cannot recover from a situation

[4]by unreachable operator instances we mean those not applicable at any point of the planning process

```
(:action unstack
:parameters (?x - block ?y - block)
:precondition (and (on ?x ?y)(clear ?x)(handempty)(stai_on ?x ?y))
:effect (and (holding ?x)(clear ?y)
             (not (clear ?x))(not (handempty))(not (on ?x ?y)))
)
```

Figure 2. An example of the encoding of an entanglement by init between the unstack operator and the on predicate.

where a block becomes eventually stacked on an "incorrect" block. Also by having a "symmetrical" entanglement by goal between the operator stack and the predicate on which allows stacking blocks only on their goal position, we might not recover from a situation where the goal tower of blocks is being built from "the middle". Hence, outer entanglements may introduce dead-ends which might be detrimental for some planning techniques, especially those based on local search.

5.2. *Reformulating Planning Tasks*

To exploit outer entanglements during the planning process we have to develop a specific planner, modify an existing one, or we have to reformulate planning tasks in such a way that outer entanglements are enforced during the search. The last option is planner-independent because, as we will show later, reformulation does not require any features which are not provided within classical (STRIPS) planning (see Section 3).

Encoding outer entanglements is done by introducing static predicates that eliminate instances of operators that do not "comply" with these entanglements (for more background details, see (Chrpa and Barták, 2009)). Let Π be a planning task, I be its initial state and G its goal. Let an operator o be entangled by init (resp. goal) with a predicate p (o and p are defined in the domain model of Π) in Π. Then the task Π is reformulated as follows:

(1) Create a predicate p' (not defined in the domain model of Π) having the same arguments as p and add p' into the domain model of Π.
(2) Modify the operator o by adding p' into its precondition. p' has the same arguments as p which is in precondition (resp. positive effects) of o.
(3) Create all possible instances of p' which correspond to instances of p listed in I (resp. G) and add the instances of p' to I.

Adding p', which is in fact a static predicate, into precondition of o causes that instances of o that are "prohibited" by the entanglement become unreachable. Figure 2 depicts the encoding of an entanglement by init between the unstack operator and the predicate on. In our terminology, unstack(?x ?y) refers to o, on(?x ?y) to p and stai_on(?x ?y) to p'. Correctness of the reformulation is formally proved as follows.

Proposition 1: *Let Π be a planning task, o be a planning operator and p be a predicate (o and p are defined in the domain model of Π) such that o is entangled by init (resp. goal) with p in Π. Let Π' be a planning task obtained by reformulating Π using the previous approach. π' is a solution plan of Π' if and only if π' is a solution plan of Π that satisfies the entanglement conditions (see Definition 4).*

Proof. Hereinafter, we will refer to modified o as o'. Adding predicates only into precon-

dition of an operator does not affect the result of application of its instances. For each ground substitution ξ (mapping variable symbols to constants) it holds that applying $\xi(o')$ in some state s (if possible) results in the same state as applying $\xi(o)$ in s (o and o' have the same variable symbols). From this, we can observe that if π' is a solution plan of Π', then π' is a solution plan of Π. For each ground substitution ξ (mapping variable symbols to constants) it holds that $\xi(p') \in I' \leftrightarrow \xi(p) \in I$ or $\xi(p') \in I' \leftrightarrow \xi(p) \in G$ respectively (I' is the initial state of Π'). No instance of p' can be achieved or deleted during the planning process, since no operator defined in the domain model of Π' has p' in its positive or negative effects. Hence, for every o's instance $a \in \pi'$ and p's corresponding instance $p_{gnd} \in pre(a)$ it is the case that $p_{gnd} \in pre(a) \rightarrow p_{gnd} \in I$ (resp. $p_{gnd} \in pre(a) \rightarrow p_{gnd} \in G$). From this, π' also satisfies the entanglement conditions in Π (see Definition 4). Also, given that each instance of p present in I (resp. G) has its "twin", i.e., a corresponding instance of p' present in I', only instances of o that violate the entanglement are pruned. Therefore, if π' is a solution plan of Π that satisfies the entanglement conditions, then π' is a solution plan of Π'. □

5.3. *Extracting Entanglements from Training Plans*

Deciding whether a given outer entanglement holds is generally PSPACE-complete as well as deciding whether the set of outer entanglements hold (as discussed in Section 4.2). Trivial entanglements, which can be identified easily (see Section 4.3), are not informative and thus not considered for task reformulation. Therefore, we have to devise an effective approximation technique for extracting sets of outer entanglements. We assume that tasks having the same domain model have a similar structure, so the same set of outer entanglements holds in all of them. Hence, we can select a representative set of simple tasks for each domain model as training tasks, so those can be solved easily by standard planning engines. Generated training plans, that is the solutions of these training tasks, are then explored in order to find what entanglements hold in them.

The above approach can be formalised as follows. Let $\mathcal{P}$ be a class of planning tasks that has the same domain model. Let $\mathcal{P}_T \subset \mathcal{P}$ be a set of training tasks. In our approximation method, we assume that $ENT_{\mathcal{P}_T} = ENT_{\mathcal{P}}$, in other words, a set of outer entanglements valid on training planning tasks is also valid on the whole class of planning tasks. This assumption is a potential source of incompleteness, since using a set of outer entanglements that does not hold for some planning tasks may make a task unsolvable within that re-formulation. On the other hand, planning tasks having the same domain model are of similar structure (e.g. they differ only by number of objects), which is the case of the most of IPC benchmarks. This provides evidence that selecting a small set of these tasks such that selected tasks are easy but not trivial, can mitigate the heuristic nature of the method, and thus support the assumption. A thorough empirical study that also explores these issues is provided in Section 6.

Determining whether a set of outer entanglements holds in all the training plans is often not a very efficient way to determine a useful set of outer entanglements, as previously discussed in the literature (Chrpa and Barták, 2009). There are two main reasons. Firstly, training plans might contain redundant actions or very sub-optimal sub-plans which can prevent detecting some useful entanglements. Secondly, there might be several strategies how a task can be solved, where only some of these lead into discovery of some useful entanglements. For example, in BlocksWorld, we might "put aside" blocks in two different ways: put them on the table, or stack them on other blocks. Only the former way leads to the discovery of two useful outer entanglements (i.e., unstack is entangled by init with on

Algorithm 1 Checking how many times the outer entanglement conditions are met for each relevant pair (operator,predicate).

Require: a set of training tasks with corresponding solution plans (training plans), flaw ratio η
Ensure: a set of outer entanglements ENT

initialize_ent_arrays(); {create empty arrays $entI, entG$ of size [Ops, Preds]}
initialize_op_counter(); {create an empty array *counter* of size [Ops]}
for each training plan $\pi = \langle a_1, \dots a_n \rangle$ **do**
  for $i := 1$ to n **do**
    for each $p \in pre(a_i)$ **do**
      if $p \in I$ **then**
        $entI[is\_inst(a_i), is\_inst(p)] + +$;
      end if
    end for
    for each $p \in eff^+(a_i)$ **do**
      if $p \in G$ **then**
        $entG[is\_inst(a_i), is\_inst(p)] + +$;
      end if
    end for
    $counter[is\_inst(a_i)] + +$;
  end for
end for
$ENT = \emptyset$
for each $(o, p) \in$[Ops,Preds] such that $counter(o) > 0$ **do**
  if not trivial $e_I(o, p)$ **and** entI$[o, p]/counter(o) \geq 1 - \eta$ **then**
    $ENT = ENT \cup \{e_I(o, p)\}$
  end if
  if not trivial $e_G(o, p)$ **and** entG$[o, p]/counter(o) \geq 1 - \eta$ **then**
    $ENT = ENT \cup \{e_G(o, p)\}$
  end if
end for

and stack is entangled by goal with on). Using optimal planners might alleviate the issue related to sub-optimal plans but it might be computationally very expensive even for training tasks. Moreover, using optimal planners might not handle the "strategy issue".

Introducing a *flaw ratio* $\eta \in [0; 1]$ which is a parameter referring to an allowed percentage of "flaws" in training plans can identify outer entanglements that can be discovered in plans that are somehow "close" to the training plans. Let η be a flaw ratio, then the outer entanglements are extracted as follows:

$$e_I(o, p) \quad \Leftrightarrow \quad \frac{entI[o, p]}{counter[o]} \geq 1 - \eta \tag{1}$$

$$e_G(o, p) \quad \Leftrightarrow \quad \frac{entG[o, p]}{counter[o]} \geq 1 - \eta \tag{2}$$

The method for extracting sets of outer entanglements in training plans is presented in our previous work (Chrpa and Barták, 2009). For every action we check how many times instances of predicates in its precondition and positive effects, respectively correspond with atoms in the initial state and the goal, respectively of the given training task.

Algorithm 2 Extraction of outer entanglements with the flaw ratio.

Require: $init\text{-}fr$ (the initial value of flaw ratio), $step$ (a decrement of flaw ratio)
Ensure: reformulated planning tasks
1: generate training plans
2: $\eta = init\text{-}fr + step$
3: **repeat**
4: $\eta = \max(0, \eta - step)$
5: extract entanglements by Alg. 1 considering η
6: generate reformulated training tasks
7: **until** $\eta = 0$ **or** all the reformulated training tasks are solvable
8: generate reformulated (testing) tasks

This information is then used for extracting a set of outer entanglements. This idea is elaborated in Algorithm 1. We define an array *counter*, which stores information about how many instances of given operators occur in the training plans, arrays $entI$, $entG$, which count how many times the conditions of entanglement by init or goal are satisfied for pairs of operators and predicates (Lines 3-17). Function $is\_inst(arg)$ returns either an operator if arg (action) is an instance of it or a predicate if arg (atom) is an instance of it. Then, a set of outer entanglements is extracted according to a given flaw ratio η (equations (1) and (2)) while ignoring trivial outer entanglements (Lines 18-26).

Algorithm 1 requires linear time with respect to the lengths of given training plans if the number of atoms in actions' preconditions and effects is much lower than lengths of training plans, so it can be bounded by a constant.

Introducing the flaw ratio (η) might invalidate the assumption that the extracted set of outer entanglements (by Algorithm 1) holds for the training tasks. Hence, the assumption must be verified after the set of outer entanglements is extracted. This idea is elaborated in Algorithm 2. A value of flaw ratio η is initially set to $init\text{-}fr{+}step$ (Line 2). The main loop (Lines 3-7) iteratively decreases η by the decrement $step$ (Line 4), extracts a set of outer entanglements by Algorithm 1 (Line 5), reformulates the training tasks according to the approach described in Section 5.2 (Line 6) and tries to solve these reformulated training tasks. Failing to solve any of the reformulated training tasks indicates that the extracted set of entanglements does not hold for all the training tasks (so, we have to continue by going back to Line 3). Clearly, if $\eta = 0$ then, the training plans are also solution plans of the reformulated training tasks.

6. Experimental Evaluation

This section is devoted to the empirical evaluation of the impact of outer entanglements in the plan generation process. The aims of the experiments are: (i) to analyse the impact of outer entanglements on state-of-the-art planning engines; (ii) to assess how different training plans influence extraction of outer entanglements; and (iii) to measure the influence of outer entanglements on grounding, i.e., how reduced is the size of the search space.

6.1. *Experimental Setup*

In order to perform our analysis, we selected a number of planners according to i) their performance in the IPCs, and ii) the variety of techniques they exploit. Selected planners

are: *Metric-FF* (Hoffmann, 2003), *LPG-td* (Gerevini et al., 2003), *LAMA* (Richter and Westphal, 2010; Richter et al., 2011), *Probe* (Lipovetzky and Geffner, 2011; Lipovetzky et al., 2014), *MpC* (Rintanen, 2012, 2014), *Yahsp3* (Vidal, 2014), and *Mercury* (Domshlak et al., 2015).

For the empirical evaluation purposes we selected all the domains used in the learning tracks of IPC-6 and IPC-7; since outer entanglements are automatically extracted domain-specific knowledge, the learning track benchmarks seem to be the most appropriate. This test set is thus independent, open, and gives a relatively wide coverage.

In each domain, the planning tasks have the same domain model and thus differ only by planning problem specifications. Henceforth, *training problems* denote tasks that are used for learning entanglements, and *testing problems* denote tasks that are used as benchmarks. In the learning track of IPC-7 (Coles et al., 2012), a set of training problems is not explicitly provided and thus the training problems have to be generated by provided problem generators.

In Machine Learning, it is important to have a good quality training set in order to maximise the outcome of the learning process. From the planning perspective, training plans should capture the important structural aspects that are generalizable to the whole class of planning tasks. According to the observation made by Chrpa et al. (2013) sets of extracted entanglements often do not change with increasing number of training problems. Similar observations have been made when configuring portfolios of planners (Núñez et al., 2012). On the other hand, using very few training problems increases the risk of extracting outer entanglements that do not generally hold (we might be "lucky" to have a very atypical problem as a training one). Following these observations, 5 training problems per domain were used. Regarding complexity of training problems, there are some aspects that should be taken into account. If training plans are too short, it indicates that their structure might be over-constrained and thus not typical for tasks in a given class. Consequently, we might extract some outer entanglements that do not hold for such "typical" tasks. On the other hand, obtaining long training plans might be too time consuming or even impossible, since planning is computationally very expensive. Hence, we have experimentally observed –by conducting preliminary investigations on a disjoint set of benchmarks, and by considering results from literature (Chrpa and Barták, 2009; Chrpa and McCluskey, 2012)– that a reasonable size for training problems is when the length of their solution plans is at least 20 in average. With larger number of defined operators in the domain model the length of training plans should be higher (more operators yields longer solution plans).

The benchmark planners were used to generate training plans. The flaw ratio (η) was initially set to 0.2, and, in case of any of the training problems became unsolvable after incorporating outer entanglements[5], the flaw ratio was iteratively reduced by the decrement of 0.05 until the set of extracted outer entanglement held for all training problems, or the flaw ratio dropped to 0.0 (for details, see Algorithm 2). Although in the literature (Chrpa and Barták, 2009; Chrpa and McCluskey, 2012) the flaw ratio is set to 0.1, we observed on some preliminary experiments, performed on a small set of benchmarks (not included in the rest of this experimental analysis) that such a value is too conservative. On the other hand, setting the value above 0.2 led to extraction of outer entanglements that often did not hold in the training problems.

A CPU-time cutoff of 900 seconds (15 minutes, as in learning tracks of IPC) was used for both learning and testing runs. All the experiments were run on 3.0 Ghz CPU

[5]by "unsolvable" we mean those problems where the planner did not find a solution within the given time limit of 900 CPU-time seconds.

Domain	Outer Entanglements
Barman	$\forall$
Bw	{Probe} $\supset$ {Lama,Mercury,Yahsp}
Depots	{FF,Lama,LPG,MpC,Probe}
Gold-m	$\forall\backslash$ {Yahsp} $\supset$ {Yahsp}
Gripper	$\forall\backslash$ {Yahsp} $\supset$ {Yahsp}
Matching-Bw	{Lama} $\supset$ {FF,Mercury} $\supset$ {LPG} / {MpC,Yahsp} $\supset$ {Probe}
Parking	{FF}, {Lama}
Rovers	$\forall$
Satellite	$\forall$
Sokoban	$\forall$
Spanner	$\forall$
Thoughtful	{Probe} $\supset$ {FF}, {MpC} $\supset$ {LPG}, {LAMA}, {Mercury}, {Yahsp}
TPP	{FF,LPG,Mercury} $\supset$ {Lama,MpC,Probe,Yahsp}

Table 1. Sets of extracted outer entanglements according to planners whose training plans were used. $\forall$ denotes the set outer entanglements consisting of all the planners. The $\supset$ relation denotes the superset relation between corresponding sets of outer entanglements.

machine with 4GB of RAM. In this experimental analysis, IPC scores as defined in IPC-7 are used. For a planner $\mathcal{C}$ and a problem p, $Time(\mathcal{C}, p)$ is 0 if p is unsolved, and $1/(1 + \log_{10}(T_p(\mathcal{C})/T_p^*))$, where $T_p(\mathcal{C})$ is the CPU time needed by planner $\mathcal{C}$ to solve problem p (if the actual CPU time is less than 1 second, then $T_p(\mathcal{C}) = 1$, i.e., 1 second is considered as a minimum CPU time needed to solve any problem) and T_p^* is the CPU time needed by the best considered planner, otherwise. Similarly, $Qual(\mathcal{C}, p)$ is 0 if p is unsolved, and $N_p^*/N_p(\mathcal{C})$, where $N_p(\mathcal{C})$ is the cost of the plan, solution of p, obtained by $\mathcal{C}$ and N_p^* is the minimal cost of the solution plan of p among all the considered planners, otherwise. The IPC score on a set of problems is given by the sum of the scores achieved on each considered problem.

6.2. *Experimental Results: The Learning Phase*

Table 1 shows the different sets of outer entanglements that were extracted by using considered planners for generating training plans. The planners used for generating training plans are arranged into sets, where each set contains all the planners that generate the same set of outer entanglements. Where applicable, the sets are ordered according to the $\supset$ relation, which denotes the superset relation between corresponding sets of outer entanglements. For example, in Bw, we have extracted two different sets of outer entanglements. The first set was extracted from training plans generated by Probe, the second was extracted from training plans generated by either LAMA or Mercury or Yahsp. The first set is a superset of the second one. Notice that using training plans generated by either FF or LPG or MpC has led to an empty set of outer entanglements (i.e., no non-trivial outer entanglements have been extracted). Hereinafter, a *planner*-plan set of entanglements will denote a set of outer entanglements obtained by using training plans extracted by a given *planner*, i.e., an FF-plan set of entanglements is obtained by using plans extracted by FF. We have made the following observations:

- no outer entanglements were detected at all in the nPuzzle domain, hence we have omitted the results from this domain in the rest of the analysis;

- in five domains, namely Barman, Rovers, Satellite, Sokoban and Spanner, the sets of outer entanglements were the same for all the planners;
- in Matching-bw and Thoughtful, there were considerable differences among the planners;
- in Parking and Thoughtful, there is no set that contains all the outer entanglements (i.e., there is not a superset to all the other sets)

Pruning power of outer entanglements along with numbers of extracted outer entanglements is shown in Table 2. Ratios of instantiated atoms and actions by the Probe planner in reformulated vs. original problems are presented (e.g. a value of 0.7 means that 70% of atoms/actions are considered in the reformulated tasks). Notice that Mercury- and Probe-sets in Thoughtful do not hold for 6 and 4 testing problems respectively. In these problems, the reachability check did not reach the goal, so these (reformulated) problems are unsolvable. Table 2 therefore refers to reduction of the size of problem representation when outer entanglements are applied.

Remarkably, different outer entanglements have different pruning power. For example, in Rovers five entanglements prune only about 11% of actions and 5% of atoms, while one entanglement in Bw (set no. II) prunes about 97% of action and 94% of atoms.

6.3. *Experimental Results: The Testing Phase*

Table 3 gives us an overview of performance of planners on the original testing problems and problems reformulated by different sets of outer entanglements we extracted during the learning phase (see Tables 1 and 2). In particular, the results simulates a "competition" between different encodings for each domain and planner. Coverage denotes how many tasks (out of 30) were solved in the given time-limit (15 minutes) for each configuration. Remarkably, in Bw, Gripper, and Matching-Bw, the use of inner entanglements allowed some planners to solve all 30 problems in the given time-limit in spite of the fact that they did not solve any task using the original encoding. The IPC score gives a relative evaluation that ranges per task from 0, i.e., the task has not been solved, to 1, i.e., the task has been solved in the smallest CPU-time (or in 1 second) or the solution plan is of the best quality. Hence, if the IPC score is equal (or very close to) the coverage, then the tasks were solved in the (nearly) smallest time or (nearly) the best quality among the encodings. For example, In Depots, the I set provides the best results against the original encoding among almost all the planners.

In summary, in the vast majority of cases, using outer entanglements has positive impact on the planners' performance. For example, in Bw, Depots, Matching-bw, Gripper and TPP the impact is remarkable. Table 4 summarizes results across the planners for each set of outer entanglements as well as original encodings and ranks them according to achieved score in three categories - coverage, speed and quality. As can be seen from Table 2, the sets of outer entanglements are ordered according to their pruning power, i.e., the set I is the most pruning set of outer entanglements. Also, as Table 1 shows in all the cases, except Parking and Thoughtful, I contains all the outer entanglements extracted in a particular domain, i.e., I is a superset of all other sets of outer entanglements.

In three domains out of 13, namely Barman, Parking and Sokoban, the original encoding achieved the best overall results, although in Sokoban, the original encoding yielded to worse coverage. In Thoughtful, the set III yielded to the best performance, while the set I unperformed even the original encoding. In TPP, the set II was the best in the coverage and quality metrics, while the set I was the best in the speed metric. In the rest of domains, the set I shows the best performance according to all the criteria.

No.	Set	EI	EG	Atoms	Actions
Barman					
I	∀	1	1	0.54	0.53
Bw					
I	{Probe}	1	1	0.06	0.02
II	{Lama,Mercury,Yahsp}	0	1	0.06	0.03
Depots					
I	{FF,Lama,LPG,MpC,Probe}	2	1	0.28	0.07
Gold-miner					
I	∀\{Yahsp}	3	0	1.00	0.90
II	{Yahsp}	2	0	1.00	1.00
Gripper					
I	∀\{Yahsp}	2	1	0.71	0.07
II	{Yahsp}	2	0	1.00	0.54
Matching-bw					
I	{Lama}	1	4	0.25	0.08
II	{FF,Mercury}	0	4	0.25	0.10
III	{LPG}	0	3	0.63	0.44
IV	{MpC,Yahsp}	0	3	0.63	0.44
V	{Probe}	0	2	0.63	0.56
Parking					
I	{Lama}	1	0	1.00	0.79
II	{FF}	1	0	1.00	0.79
Rovers					
I	∀	2	3	0.95	0.89
Satellite					
I	∀	0	1	0.52	0.98
Sokoban					
I	∀	2	0	0.97	0.90
Spanner					
I	∀	1	0	1.00	1.00
Thoughtful					
I	{Yahsp}	6	0	1.00	0.70
II	{FF}	8	0	1.00	0.79
III	{Lama}	5	1	1.00	0.81
IV	{MpC}	10	0	1.00	0.81
V	{LPG}	6	0	1.00	0.92
VI	{Mercury}	7	1	N/A	N/A
VII	{Probe}	15	0	N/A	N/A
TPP					
I	{FF,LPG,Mercury}	3	0	0.40	0.03
II	{Lama,MpC,Probe,Yahsp}	2	0	0.40	0.09

Table 2. Numbers of entanglements by init (EI) and by goal (EG) per set. Ratios of instantiated atoms and actions by the Probe planner in reformulated vs. original testing problems (in ascending order per domain). ∀ denotes the set consisting of all the planners.

Figure 3 shows the coverage performance of the considered planners when exploiting the original encodings, and the encodings enhanced with the I sets of outer entanglements. Results are cumulative across all the testing benchmarks and demonstrate that coverage increased overall for each planner when the I sets of outer entanglements are used. The largest impact can be observed on the performance of the MpC planner, where the coverage is increased by 125 instances (32.1%). We believe that such an improvement was

	FF			LPG			Lama			Probe			MpC			Mercury			Yahsp		
Set	C	S	Q	C	S	Q	C	S	Q	C	S	Q	C	S	Q	C	S	Q	C	S	Q
Barman																					
O	0	0.0	0.0	0	0.0	0.0	0	0.0	0.0	4	3.3	4.0	0	0.0	0.0	**24**	**24.0**	**23.9**	0	0.0	0.0
I	0	0.0	0.0	0	0.0	0.0	0	0.0	0.0	**24**	**24.0**	**23.1**	0	0.0	0.0	3	2.4	3.0	0	0.0	0.0
Bw																					
O	0	0.0	0.0	24	8.0	16.1	25	10.3	16.0	25	8.0	21.4	0	0.0	0.0	19	9.6	7.7	28	17.4	5.1
I	**30**	**30.0**	**30.0**	**30**	29.3	**30.0**	28	**28.0**	**28.0**	**30**	**29.8**	**29.9**	**30**	**29.7**	**30.0**	**30**	**26.6**	**30.0**	**30**	**28.9**	**30.0**
II	21	12.6	11.0	**30**	**29.6**	16.1	**29**	19.9	15.2	**30**	28.4	29.2	**30**	24.3	22.6	29	25.5	11.1	**30**	27.5	5.2
Depots																					
O	1	0.3	0.7	11	4.0	9.5	0	0.0	0.0	**30**	12.8	27.4	18	6.1	15.1	0	0.0	0.0	21	8.9	4.4
I	**30**	**30.0**	**30.0**	**30**	**30.0**	**29.8**	**27**	**27.0**	**27.0**	**30**	**29.5**	**29.7**	**30**	**30.0**	**30.0**	**27**	**27.0**	**27.0**	**30**	**30.0**	**30.0**
Gold-miner																					
O	**30**	24.8	**30.0**	**30**	**30.0**	**29.5**	**30**	**30.0**	14.2	**30**	**30.0**	29.4	**30**	**30.0**	24.0	**30**	28.1	19.5	25	24.0	24.1
I	**30**	**30.0**	27.6	**30**	**30.0**	29.4	**30**	**30.0**	**29.6**	**30**	**30.0**	**29.5**	**30**	**30.0**	**27.3**	**30**	**30.0**	**30.0**	**30**	**30.0**	**27.3**
II	**30**	24.8	**30.0**	**30**	**30.0**	**29.5**	**30**	**30.0**	28.2	**30**	**30.0**	29.4	**30**	**30.0**	27.3	**30**	27.9	19.5	28	25.1	27.0
Gripper																					
O	0	0.0	0.0	**30**	16.1	27.1	0	0.0	0.0	0	0.0	0.0	0	0.0	0.0	0	0.0	0.0	0	0.0	0.0
I	**3**	**3.0**	**3.0**	**30**	**30.0**	29.9	0	0.0	0.0	**30**	**30.0**	**30.0**	**30**	**30.0**	**30.0**	0	0.0	0.0	0	0.0	0.0
II	0	0.0	0.0	**30**	21.0	**29.9**	0	0.0	0.0	0	0.0	0.0	15	7.6	14.9	0	0.0	0.0	0	0.0	0.0
Matching-bw																					
O	12	4.7	10.2	21	10.4	15.0	22	17.4	15.7	15	6.6	9.2	0	0.0	0.0	9	4.7	6.4	14	9.4	6.0
I	**30**	**30.0**	**29.8**	**30**	**29.6**	**29.4**	**30**	**29.8**	**29.6**	**30**	**29.7**	**30.0**	**30**	**30.0**	**29.8**	**30**	**29.9**	**30.0**	**30**	**30.0**	**30.0**
II	**30**	25.9	24.0	**30**	26.9	25.3	**30**	26.0	21.5	**30**	25.2	19.5	**30**	**30.0**	22.7	**30**	27.5	19.5	**30**	28.1	14.4
III	25	13.6	19.5	27	22.3	19.0	28	21.1	20.8	21	14.0	13.5	22	9.4	9.6	25	15.0	14.1	26	19.9	11.9
IV	27	17.8	21.4	**30**	12.3	23.6	29	20.5	20.7	25	14.3	16.0	25	14.8	11.7	28	17.5	17.3	28	24.1	12.8
V	21	10.1	16.7	27	16.7	20.2	28	19.4	20.3	16	9.9	10.3	12	6.0	5.1	24	14.0	12.9	25	18.6	11.3
Parking																					
O	**7**	5.2	**6.6**	0	0.0	0.0	**9**	**8.5**	**9.0**	3	2.8	2.8	**5**	**5.0**	**5.0**	6	5.5	6.0	0	0.0	0.0
I	5	4.6	4.8	0	0.0	0.0	2	1.5	1.4	1	1.0	1.0	1	0.5	0.8	**8**	**7.7**	**6.3**	0	0.0	0.0
II	**7**	**6.5**	6.4	0	0.0	0.0	2	2.0	1.5	**5**	**4.8**	**4.6**	4	3.4	3.5	1	0.6	0.7	**5**	**5.0**	**5.0**
Rovers																					
O	0	0.0	0.0	**28**	**26.7**	**27.9**	28	25.4	27.7	28	26.5	27.6	6	4.2	5.9	**24**	23.6	**24.0**	**30**	21.4	**30.0**
I	0	0.0	0.0	26	25.8	25.9	**29**	**29.0**	**28.9**	**29**	**29.0**	**28.7**	**23**	**23.0**	**22.9**	**24**	**23.9**	**24.0**	**30**	**30.0**	**30.0**
Satellite																					
O	0	0.0	0.0	**30**	27.4	**30.0**	3	3.0	2.9	0	0.0	0.0	**1**	**1.0**	**1.0**	**20**	19.9	**20.0**	**16**	15.4	**16.0**
I	0	0.0	0.0	**30**	**30.0**	**30.0**	**4**	**4.0**	**4.0**	0	0.0	0.0	**1**	0.9	1.0	**20**	**20.0**	**20.0**	**16**	**16.0**	**16.0**
Sokoban																					
O	18	16.0	17.5	**28**	**25.9**	23.9	**19**	**18.1**	**18.0**	**24**	20.6	**22.2**	**30**	**28.6**	**28.3**	**19**	**18.1**	**18.0**	25	20.2	23.8
I	**22**	**19.9**	**20.5**	26	21.4	**24.4**	**19**	14.6	15.3	**24**	**22.4**	21.7	**30**	27.7	27.9	**19**	14.6	15.3	**28**	**26.2**	**24.5**
Spanner																					
O	0	0.0	0.0	**30**	29.9	**30.0**	0	0.0	0.0	0	0.0	0.0	**30**	29.9	**30.0**	0	0.0	0.0	0	0.0	0.0
I	0	0.0	0.0	**30**	**30.0**	**30.0**	0	0.0	0.0	0	0.0	0.0	**30**	**30.0**	**30.0**	0	0.0	0.0	0	0.0	0.0
Thoughtful																					
O	17	12.9	16.0	0	0.0	0.0	25	19.4	22.3	20	16.7	18.1	0	0.0	0.0	21	17.3	19.7	8	6.3	6.0
I	17	11.6	15.3	1	0.6	0.9	23	20.0	18.8	18	15.1	13.7	0	0.0	0.0	21	18.7	19.0	2	2.0	1.7
II	18	16.7	17.2	1	0.6	0.9	25	19.4	22.3	21	15.6	18.5	**2**	**2.0**	**2.0**	22	20.5	21.3	17	14.6	16.2
III	**27**	**25.3**	**26.2**	0	0.0	0.0	**28**	**24.5**	**27.4**	**25**	**19.5**	**23.7**	0	0.0	0.0	**27**	**24.2**	**26.1**	**23**	**18.9**	**21.8**
IV	17	14.0	15.9	1	1.0	1.0	20	17.7	17.8	21	19.2	18.5	0	0.0	0.0	19	16.9	17.7	6	4.5	3.6
V	15	11.2	14.1	0	0.0	0.0	25	18.9	22.5	17	14.1	15.8	0	0.0	0.0	20	16.3	18.8	12	9.8	9.8
VI	16	13.5	13.8	**17**	**15.3**	**17.0**	15	14.0	13.2	15	12.1	12.9	1	1.0	1.0	15	13.6	12.9	13	9.4	11.8
VII	11	7.4	8.7	11	11.0	9.5	10	7.2	8.1	8	5.9	6.2	1	1.0	1.0	10	7.3	7.6	10	9.2	9.0
TPP																					
O	0	0.0	0.0	1	0.3	0.6	16	6.1	15.3	14	5.1	13.8	11	4.7	10.6	19	8.3	19.0	20	12.7	19.9
I	0	0.0	0.0	27	**26.8**	26.3	**30**	**30.0**	28.9	**30**	**30.0**	27.0	**21**	**19.7**	**20.8**	**30**	**30.0**	26.0	**30**	**30.0**	29.4
II	**3**	**3.0**	**3.0**	**29**	20.7	**27.7**	**30**	20.0	**29.2**	**30**	18.7	**29.7**	18	14.5	17.4	**30**	22.2	**28.9**	**30**	26.2	**29.8**

Table 3. Comparing planners' performance in terms of (C)overage, (S)peed IPC score and (Q)uality IPC score on the (O)riginal and reformulated tasks by different sets of outer entanglements. The best results per planner and domain are highlighted.

achieved because outer entanglements reduced, often considerably, memory requirements. On the contrary, Yahsp coverage performance are less affected by the exploitation of set I entanglements: 39 more instances are solved (10.0%).

Coverage		Speed		Quality		Coverage		Speed		Quality	
Set	Count	Set	Score	Set	Score	Set	Count	Set	Score	Set	Score
Barman						**Rovers**					
O	28	O	27.3	O	27.9	I	161	I	160.7	I	160.4
I	27	I	26.4	I	26.1	O	144	O	127.7	O	143.1
Bw						**Satellite**					
I	208	I	202.3	I	207.9	I	71	I	70.9	I	70.9
II	199	II	167.8	II	119.2	O	70	O	66.6	O	69.9
O	121	O	53.4	O	66.2	**Sokoban**					
Depots						I	168	O	147.3	O	151.7
I	204	I	203.5	I	203.4	O	163	I	146.9	I	149.6
O	81	O	32.0	O	57.1	**Spanner**					
Gold-miner						I/O	60	I	60.0	I/O	60.0
I	210	I	210.0	I	200.7	I/O	60	O	59.9	I/O	60.0
II	208	II	197.9	II	190.9	**Thoughtful**					
O	205	O	196.9	O	170.8	III	130	III	112.4	III	125.1
Gripper						II	102	II	88.5	II	95.8
I	93	I	93.0	I	92.8	VI	92	VI	78.9	VI	82.4
II	45	II	28.6	II	44.8	O	91	IV	73.3	O	82.1
O	30	O	16.1	O	27.1	V	89	O	72.6	V	81.0
Matching-bw						IV	84	V	70.2	IV	74.5
I/II	210	I	209.0	I	208.7	I	82	I	60.8	I	69.3
I/II	210	II	189.6	II	146.9	VII	61	VII	49.0	VII	50.2
IV	192	IV	123.6	IV	123.4	**TPP**					
III	174	III	115.4	III	108.4	II	169	I	166.5	II	165.7
V	153	V	94.6	V	96.7	I	167	II	125.4	I	158.5
O	93	O	53.2	O	62.4	O	80	O	37.3	O	79.2
Parking											
O	30	O	27.0	O	29.3						
II	24	II	22.3	II	21.7						
I	17	I	15.3	I	17.2						

Table 4. Ranking the cumulative results for (O)riginal tasks and reformulated tasks by different sets of outer entanglements in coverage, and speed and quality IPC score.

6.4. *Analysis of the Results*

The aim of outer entanglements is to (i) eliminate unpromising instances of planning operators, which, consequently, reduces the branching factor, and (ii) reduce the size of task representation, which, consequently, can also reduce the size of the state space. Given the planner-independent nature of outer entanglements, i.e., they can be encoded directly in the planning task, any standard planning engine can benefit from them. The most significant impact on planners' performance is given by outer entanglements in Bw, Depots, Gripper, Matching-bw and TPP. In these domains, the I sets of outer entanglements had the strongest pruning power, in particular, they eliminated more than 90% of actions (see Table 2). Also, the number of atoms was apart of the Gripper domain reduced by at least 60%. The results (see Tables 3 and 4) demonstrate that the performance gain of the planning engines is often considerable. Therefore, outer entanglements learnt in these domain do efficiently both – eliminating unpromising operators' instances and (considerably) reducing the state space.

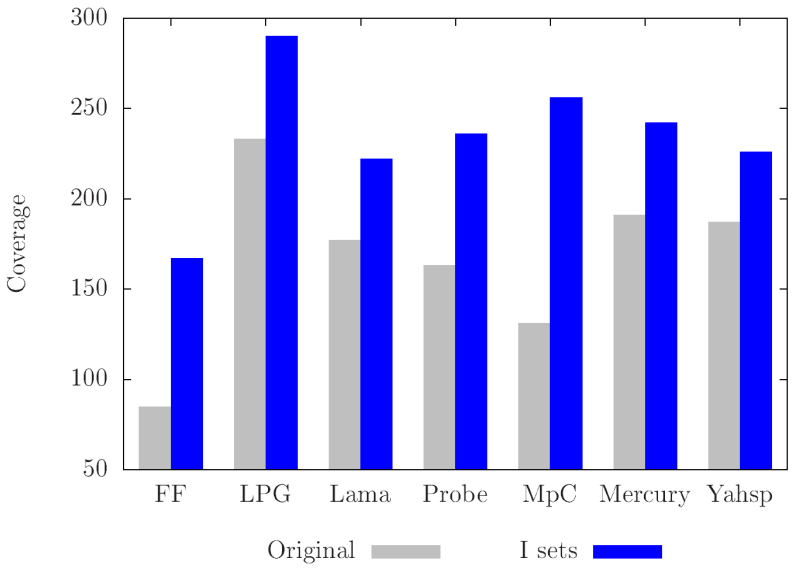

Figure 3. Cumulative coverage performance of the considered planners exploiting the original encodings and the encodings enhanced with the I sets of outer entanglements, across all the testing benchmarks.

By analyzing the impact of outer entanglements in particular domains, we can see the following. In Bw and its variant Matching-bw, there are two possibilities how blocks can be temporarily put aside – putting them on the table, or stacking them on other blocks. Outer entanglements enforce the former option. This drastically reduces the size of the state space because blocks can be only in their initial or goal positions, on the table, or being held by the robotic hand. Moreover, temporarily stacking blocks on other (non-goal) blocks introduces further constraints, i.e., a block on which we temporarily stack another block cannot be moved without taking that other block out. A possible drawback is in introducing dead-ends. If a stack of blocks is incorrectly built from "the middle", the planner cannot repair it (it is impossible to unstack blocks from other than initial configurations) and has to backtrack. However, the results clearly indicate that introducing dead-ends in these domains does not negatively affect planners' performance. The Gripper domain describes the problem of moving balls between rooms by robots with two grippers. Outer entanglements in this domain prevent to pickup a ball in other than its initial location as well as to drop the ball in other than its goal location. The planners thus do not have to consider to temporarily leave balls in non-goal locations, which as the results indicate is beneficial for some planners. Similar observations can be made in Depots and TPP.

Thoughtful is a variant of the well known freecell card game (Bjarnason et al., 2007). There are a number of strategies that can be exploited in order to achieve a goal configuration of cards. Interestingly, each planner exploited a different strategy while solving training problems which led to extraction of different sets of outer entanglements most of which are incomparable to each other (see Table 1). As mentioned before, the sets VI and VII do not hold for 4 and 6 testing problems respectively. This indicates that the training problems were too constrained and that some strategies feasible for solving them might not generalize well for wider range of (testing) problems. On the bright side, the set III brought a considerable performance improvement among the planners (except LPG that benefited from the VI set). A closer analysis of the IPC speed scores indicates that testing problems do not unanimously benefit from a single encoding (i.e., there is a gap between coverage and the IPC speed score). Also, given the discrepancies between

learnt sets of entanglements for each planner (in spite of the fact that training problems were same for all the planners), planners' "sensitivity" might very vary for particular sets of outer entanglements. In such cases determining the most promising set of entanglements can be done by cross-validating their performances on several tasks which are more complex than the training ones.

In Barman and Parking, outer entanglements underperformed the original encodings. Table 3 shows that the results are more mixed, i.e., some planners benefit from outer entanglements, some do not. In Barman, outer entanglements enforce cocktails to be poured only into the "goal" shots. Probe considerably benefits from such a restriction, while it has a very detrimental impact for Mercury. The reason for the latter seems to be in Mercury's inefficient handling of situations where the "goal" shot is not clean while the cocktail is being prepared. Parking, which deals with a problem of rearranging cars on a parking lot, is a combinatorial domain. The sets of outer entanglements seem to be beneficial only for some planning techniques and a limited number of testing problems. Such results point to the fact that despite their reasonable pruning power (around 50% of actions and atoms were pruned in the Barman domain) outer entanglements can have detrimental effects on some planning techniques (such as Red-Black heuristics accommodated in the Mercury planner (Domshlak et al., 2015)).

In Gold-miner, Rovers, Satellite, Sokoban and Spanner, outer entanglements slightly outperformed the original encodings. In these domains, however, outer entanglements have limited pruning power and hence their impact on planners' performance is limited.

In spite of a few cases where outer entanglements have rather detrimental effects on planning engines, the results demonstrated, as summarized in Figure 3, that the use of outer entanglements improve performance of planning engines regardless planning techniques they exploit across a number of different domains. Hence, outer entanglements can be considered a fruitful technique to be exploited both in domain-independent and planner-independent fashion.

7. Discussion

This section is devoted to discussion of benefits and drawbacks of outer entanglements and provide general recommendation for their extraction and use.

7.1. *Extracting "Good" Sets of Outer Entanglements*

The outcome of the outer entanglement learning process depends on training problems and solution plans of these problems (i.e., training plans). According to a study of Chrpa et al. (2013) if the number and complexity of training problems, which is determined by length of solution plans, increase above certain thresholds, the impact on the outcome of the outer entanglement learning process is negligible. On the other hand, using different planners to generate training plans might lead to considerably different results.

Our experimental results, where the number of training problems were set to 5 and whose solution (or training) plans consisted of at least 20 actions in average per domain, have shown that these thresholds are sufficient for generating sets of outer entanglements that hold also for testing problems and improve planners' performance. The only exception has been observed in the Thoughtful domain, where Mercury- and Probe-sets did not hold for some of the testing problems. Since the Thoughtful domain is complex (containing more than 20 planning operators), the used thresholds might be too low. Hence, when setting up the training problems, it is important to consider complexity of

the domain model and adjust the thresholds accordingly, i.e., the number and complexity of training problems should be higher for more complex domain models.

For generating training plans, we have used 7 different planners. In 5 domains, the extracted sets of outer entanglements were identical regardless of the used planner. On the other hand, in Matching-bw and Thoughtful, the extracted sets of outer entanglements considerably differed (for details, see Tables 1 and 2). Also, the impact of different sets of outer entanglements on planners' performance often varied considerably (see Tables 3 and 4). Except Parking and Thoughtful, we were able to identify a set of outer entanglements that subsumes the other sets. Since such a set always has the strongest pruning power, in our experiments the set was denoted as I. With a few notable exceptions (e.g. Barman), the I sets outperformed, often considerably, the other sets as well as the original encodings. In Thoughtful, the Lama-set (set III) was the best performing set, while in Parking, the Lama-set (set I) was the worst performing set. Interestingly, in both cases Lama generated the best quality training plans. Moreover, the Lama-set in Thoughtful does not have the strongest pruning power, it is the Yahsp-set (set I) that underperformed even the original encoding. It should be, however, noted that training plans in Thoughtful generated by Yahsp were of a low quality (the worst among the planners).

Lessons learnt from analyzing the experimental results indicate that i) poor quality training plans lead to empty or "poor" sets of outer entanglements, ii) the best quality training plans do not necessarily lead to "good" sets of outer entanglements, iii) sets containing all extracted outer entanglements tend to be the best performing ones, and iv) incomparable sets of outer entanglements indicate lack of training data (i.e., a small number and low complexity of training problems). Notice that the best quality training plans are not necessarily optimal (unless an optimal planner is used for their extraction). As poor quality training plans we consider those that are at least 50% longer than the best quality ones. We believe that taking into account these lessons provides a general guidance for performing effective and efficient outer entanglement learning process.

7.2. *Heuristic Nature of the Learning Method*

Because deciding whether an outer entanglement holds in a given planning task is PSPACE-complete, we have developed an approximation method that learns outer entanglements from training plans, solutions of simple planning tasks. Our method, therefore, follows an assumption that a learnt set of outer entanglements holds for every task in a given class (or domain). There is, however, no theoretical guarantee that the assumption will hold for every (non-training) planning task. Although the experiments have demonstrated a strong support for the assumption, in a few cases we have observed that the assumption did not hold, and to solve a problem the system would have to revert to its original formulation. Theoretically, after a reformulated task is proved to be unsolvable, the original task has to be solved (or proven unsolvable too). Such an approach might be practically reasonable only in cases in which the unsolvability of the reformulated task is proved quickly (e.g., goals are not reachable). Another possibility to alleviate the heuristic issue is to integrate outer entanglement reformulated tasks within planning portfolios. Also, an engineer who has developed a domain model might manually decide whether a learnt set of outer entanglements holds for planning tasks using that domain model. In cases such as BlocksWorld, it might be easy.

7.3. *Optimality*

The quality of plans has improved in many cases when outer entanglements were used. Intuitively, pruning the search space may force planners to find better solution plans. On the other hand, outer entanglements do not guarantee optimality in general. Strengthening definitions of outer entanglements to guarantee plans optimality is, of course, theoretically possible. Given the complexity results of "normal" entanglements, we can expect the same for "optimal" entanglements. Using the approximation algorithm for learning outer entanglements on optimal training plans with zero flaw ratio might extract some useful "optimal" outer entanglements. However, we believe that there is a high risk of extracting "suboptimal" outer entanglements that prune optimal solution plans. For example, in Logistic-like domain, it is often an optimal strategy to pick up packages from their initial locations and deliver them to their goal locations. This can be captured by outer entanglements. However, if driving between some locations is very expensive, it might be better to move some packages from other trucks to one truck which will perform the "expensive" journey. Here, the outer entanglements will prevent to do so and thus will prune optimal solutions (although the task will remain solvable).

8. Conclusions and Future Work

In this paper we presented outer entanglements, relations between planning operators and predicates whose instances are in the initial state or the goal. Outer entanglements are used to eliminate unpromising instances of planning operators and thus reduce branching factor in the state space as well as the size of problem representation (and, consequently, the size of the search space). To deal with the intractability of deciding whether a given outer entanglement holds for a given planning task (see Section 4.2), we used a learning method for extracting "domain-specific" sets of outer entanglements from training plans, solution plans of simple tasks. Outer entanglements can be encoded into domain models without extending the input language of a planner (see Section 5.2) and, therefore, they can be understood and exploited as planner-independent knowledge.

The extensive experimental analysis in this paper demonstrates that outer entanglements improve the planning process considerably within a wide range of competition domains and state of-the-art planning engine combinations. Our experiments using 7 state of the art planning engines, and 14 benchmark domain models, have given a good indication of the planner and domain independence, and the effectiveness of the method: in the overwhelming majority of the cases, outer entanglements caused a substantial improvement in plan generation speed and solution plan quality.

We identified several avenues for future research. Firstly, we plan to incorporate outer entanglements into well known planning frameworks such as Fast Downward (Helmert, 2006) or LAPKT (Ramirez et al., 2015) so they can be exploited, for instance, for computing heuristics. Secondly, given the encouraging spread of results among sets of planners and domains, we intend to work towards including an entanglements generating facility as part of a knowledge engineering workbench. Finally, we plan to extend the concept of outer entanglements for non-classical planning which will potentially improve performance of real-world planning applications (preliminary work has been started on this in numerical planning (Chrpa et al., 2015a)).

Acknowledgments

The authors would like to acknowledge the use of the University of Huddersfield Queensgate Grid in carrying out this work.

This Research was partly funded by the Czech Science Foundation (project no. 18-07252S) and by the UK EPSRC Autonomous and Intelligent Systems Programme (grant no. EP/J011991/1).

References

Baier, J. A., Fritz, C., and McIlraith, S. A. (2007). Exploiting procedural domain control knowledge in state-of-the-art planners. In *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling, ICAPS*, pages 26–33.

Bernard, D., Gamble, E., Rouquette, N., Smith, B., Tung, Y., Muscettola, N., Dorias, G., Kanefsky, B., Kurien, J., Millar, W., Nayal, P., Rajan, K., and Taylor, W. (2000). Remote agent experiment ds1 technology validation report. Technical report, Ames Research Center and JPL.

Bjarnason, R., Tadepalli, P., and Fern, A. (2007). Searching solitaire in real time. *International Computer Games Association Journal*, 30(3):131–142.

Bonet, B. and Geffner, H. (1999). Planning as heuristic search: New results. In *European Conference on Planning, ECP*, pages 360–372.

Botea, A., Enzenberger, M., Müller, M., and Schaeffer, J. (2005). Macro-ff: Improving ai planning with automatically learned macro-operators. *Journal of Artificial Intelligence Research (JAIR)*, 24:581–621.

Brafman, R. I. and Domshlak, C. (2013). On the complexity of planning for agent teams and its implications for single agent planning. *Artificial Intelligence*, 198:52–71.

Chen, Y. and Yao, G. (2009). Completeness and optimality preserving reduction for planning. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 1659–1664.

Chrpa, L. (2010). Generation of macro-operators via investigation of action dependencies in plans. *Knowledge Engineering Review*, 25(3):281–297.

Chrpa, L. and Barták, R. (2009). Reformulating planning problems by eliminating unpromising actions. In *Symposium on Abstraction, Reformulation, and Approximation, SARA*, pages 50–57.

Chrpa, L. and McCluskey, T. L. (2012). On exploiting structures of classical planning problems: Generalizing entanglements. In *European Conference on Artificial Intelligence, ECAI*, pages 240–245.

Chrpa, L., McCluskey, T. L., and Osborne, H. (2012). Reformulating planning problems: A theoretical point of view. In *International Florida Artificial Intelligence Research Society Conference, FLAIRS*, pages 14–19.

Chrpa, L., Scala, E., and Vallati, M. (2015a). Towards a reformulation based approach for efficient numeric planning: Numeric outer entanglements. In *Proceedings of the Eighth Annual Symposium on Combinatorial Search, SOCS 2015, 11-13 June 2015, Ein Gedi, the Dead Sea, Israel.*, pages 166–170.

Chrpa, L., Vallati, M., and McCluskey, T. (2015b). On the online generation of effective macro-operators. In *International Joint Conference on Artificial Intelligence, IJCAI 2015*, pages 1704–1711.

Chrpa, L., Vallati, M., and McCluskey, T. L. (2014). MUM: a technique for maximising the utility of macro-operators by constrained generation and use. In *the International Conference on Automated Planning and Scheduling, ICAPS*, pages 65–73.

Chrpa, L., Vallati, M., and Osborne, H. (2013). Learnability of specific structural patterns of planning problems. In *International Conference on Tools with Artificial Intelligence, ICTAI*, pages 18–23.

Coles, A., Coles, A., García Olaya, A., Jiménez, S., Linares López, C., Sanner, S., and Yoon, S. (2012). A survey of the seventh international planning competition. *AI Magazine*, 33(1):83–88.

Coles, A., Fox, M., and Smith, A. (2007). Online identification of useful macro-actions for planning. In *the International Conference on Automated Planning and Scheduling, ICAPS*, pages 97–104.

Coles, A. J. and Coles, A. I. (2010). Completeness-preserving pruning for optimal planning. In *European Conference on Artificial Intelligence, ECAI*, pages 965–966.

Dawson, C. and Siklóssy, L. (1977). The role of preprocessing in problem solving systems. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 465–471.

de la Rosa, T., Celorrio, S. J., Fuentetaja, R., and Borrajo, D. (2011). Scaling up heuristic planning with relational decision trees. *Journal of Artificial Intelligence Research (JAIR)*, 40:767–813.

Domshlak, C., Hoffmann, J., and Katz, M. (2015). Red-black planning: A new systematic approach to partial delete relaxation. *Artificial Intelligence*, 221:73–114.

Fox, M. and Long, D. (2003). PDDL2.1: an extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research (JAIR)*, 20:61–124.

Gerevini, A., Saetti, A., and Serina, I. (2003). Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research (JAIR)*, 20:239 – 290.

Ghallab, M., Isi, C. K., Penberthy, S., Smith, D. E., Sun, Y., and Weld, D. (1998). Pddl - the planning domain definition language. Technical report.

Ghallab, M., Nau, D., and Traverso, P. (2004). *Automated planning, theory and practice*. Morgan Kaufmann Publishers.

Gnad, D., Wehrle, M., and Hoffmann, J. (2016). Decoupled strong stubborn sets. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 3110–3116.

Haslum, P. (2007). Reducing accidental complexity in planning problems. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 1898–1903.

Haslum, P. and Jonsson, P. (2000). Planning with reduced operator sets. In *International Conference on Artificial Intelligence Planning Systems, AIPS*, pages 150–158.

Haslum, P., M.Helmert, and Jonsson, A. (2013). Safe, strong, and tractable relevance analysis for planning. In *Proceedings of the International Conference on Automated Planning and Scheduling, ICAPS*, pages 317–321.

Helmert, M. (2006). The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.

Hoffmann, J. (2003). The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. *Journal Artificial Intelligence Research (JAIR)*, 20:291–341.

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

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

Kautz, H. and Selman, B. (1992). Planning as satisfiability. In *European Conference on Artificial Intelligence, ECAI*, pages 359–363.

Knoblock, C. A. (1994). Automatically generating abstractions for planning. *Artificial Intelligence*, 68(2):243–302.

Korf, R. (1985). Macro-operators: A weak method for learning. *Artificial Intelligence*, 26(1):35–77.

Kvarnström, J. and Doherty, P. (2000). Talplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):119–169.

Lipovetzky, N. and Geffner, H. (2011). Searching for plans with carefully designed probes. In *the 21st International Conference on Automated Planning and Scheduling (ICAPS-11)*. AAAI press.

Lipovetzky, N., Ramirez, M., Muise, C., and Geffner, H. (2014). Width and inference based planners: Siw, bfs(f), and probe. In *The Eighth International Planning Competition. Description of Participant Planners of the Deterministic Track*, pages 6–7.

McCluskey, T. L. (1987). Combining weak learning heuristics in general problem solvers. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 331–333.

McCluskey, T. L. and Porteous, J. M. (1997). Engineering and compiling planning domain models to promote validity and efficiency. *Artificial Intelligence*, 95(1):1–65.

Minton, S. and Carbonell, J. G. (1987). Strategies for learning search control rules: An explanation-based approach. In *International Joint Conference on Artificial Intelligence, IJCAI*, pages 228–235.

Newton, M. A. H., Levine, J., Fox, M., and Long, D. (2007). Learning macro-actions for arbitrary planners and domains. In *the International Conference on Automated Planning and Scheduling, ICAPS*, pages 256–263.

Núñez, S., Borrajo, D., and Linares López, C. (2012). Performance analysis of planning portfolios. In *Proceedings of the Fifth Annual Symposium on Combinatorial Search, SOCS*.

Ramirez, M., Lipovetzky, N., and Muise, C. (2015). Lightweight Automated Planning ToolKiT. http://lapkt.org/. Accessed: 2016-11-1.

Richter, S. and Westphal, M. (2010). The lama planner: guiding cost-based anytime planning with landmarks. *Journal Artificial Intelligence Research (JAIR)*, 39:127–177.

Richter, S., Westphal, M., and Helmert, M. (2011). Lama 2008 and 2011. In *Booklet of the 7th International Planning Competition*.

Rintanen, J. (2012). Engineering efficient planners with sat. In *European Conference on Artificial Intelligence, ECAI*, pages 684–689.

Rintanen, J. (2014). Madagascar: Scalable planning with sat. In *The Eighth International Planning Competition. Description of Participant Planners of the Deterministic Track*, pages 66–70.

Scholz, U. (2004). *Reducing planning problems by path reduction*. PhD thesis, Darmstadt University of Technology.

Slaney, J. and Thiébaux, S. (2001). Blocks world revisited. *Artificial Intelligence*, 125(1-2):119–153.

Vallati, M., Chrpa, L., Grzes, M., McCluskey, T. L., Roberts, M., and Sanner, S. (2015). The 2014 international planning competition: Progress and trends. *AI Magazine*, 36(3):90–98.

Vidal, V. (2014). Yahsp3 and yahsp3-mt in the 8th international planning competition. In *The Eighth International Planning Competition. Description of Participant Planners of the Deterministic Track*, pages 64–65.

Wehrle, M., Helmert, M., Alkhazraji, Y., and Mattmüller, R. (2013). The relative pruning power of strong stubborn sets and expansion core. In *the International Conference on Automated Planning and Scheduling, ICAPS*.

Yoon, S. W., Fern, A., and Givan, R. (2008). Learning control knowledge for forward search planning. *Journal of Machine Learning Research*, 9:683–718.