Revisiting Inner Entanglements in Classical Planning

Lukáš CHRPA [a,1], Mauro VALLATI [a]
[a] *PARK research group, School of Computing and Engineering, University of Huddersfield, UK*

Abstract. In Automated Planning, learning and exploiting structural patterns of plans, domain models and/or problem models, in order to improve plan generation speed-up and increase the scope of problems solved, has attracted much research. Reformulation techniques such as those based on macro-operators or entanglements are very promising, mainly because they are planner-independent. This paper aims to extend and revisit the recent work on *inner entanglements*, relations between pairs of planning operators and predicates encapsulating exclusivity of predicate 'achievements' or 'requirements', in order to bring new theoretical results (PSPACE-completeness of deciding inner entanglements), present a new way of encoding of inner entanglements and empirical comparison between different kinds of inner entanglements.

Keywords. Classical Planning, Inner Entanglements, Problem Reformulation

1. Introduction

Automated planning, which deals with the problem of finding a totally or partially ordered sequences of actions transforming the environment from an initial state to a desired goal state, has been studied extensively for several decades and lead to development of many advanced planning techniques [7]. The International Planning Competition (IPC) provides a standard environment for comparing automated planners that, since the first edition of such competition, have been developed and significantly enhanced.[2] A promising way for improving the planning process is to learn some domain characteristics which can narrow the search space. One well known approach is based on macro-operators which encapsulate a sequences of (ordinary) operators [2,4].

A recent technique introduces *inner entanglements* [5], relations between pairs of planning operators and predicates, denoting exclusivity of predicate 'achievement' (entanglements by succeeding) or 'requirement' (entanglements by preceding). That is, one operator achieves a predicate exclusively for another operator, or an operator requires a predicate exclusively from another operator. Inner entanglements in fact eliminate some alternatives in the search space. Enforcing inner entanglements in a planner-independent way can be done by reformulating planning domain/problem models which is a com-

[1] Corresponding Author: Lukáš Chrpa, School of Computing and Engineering, University of Huddersfield, Queensgate, HD1 3DH, Huddersfield, United Kingdom; E-mail: l.chrpa@hud.ac.uk.

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

plementary approach to other pruning techniques, often incorporated in state-of-the-art planning engines [9,13].

In this paper, we revisit and extend the recent work on inner entanglements, i.e., entanglements by preceding and succeeding [5]. We will extend the theoretical results by proving that deciding inner entanglements is generally intractable (PSPACE-complete), i.e., as hard as classical planning itself. We will also propose a new compact encoding of inner entanglements, for cases where both entanglements by preceding and succeeding hold for a pair of operators and a predicate. Moreover, we will empirically evaluate the impact that different kinds of entanglements have on the performance of several state-of-the-art planning systems.

The rest of this paper is organized as follows. We first provide the necessary background information on classical planning, and the basic terminology. Next, we introduce the theoretical properties of inner entanglements and we describe how planning problems can be reformulated by introducing inner entanglements. Finally, we show the results of an empirical analysis and we give conclusions and discuss some avenues for future work.

2. Preliminaries

Classical planning (in state space) deals with finding a sequence of actions transforming the static, deterministic and fully observable environment from some initial state to a desired goal state [7].

In the set-theoretic representation *atoms*, which describe the environment, are propositions. *States* are defined as sets of propositions. *Actions* are specified via sets of atoms specifying their preconditions, negative and positive effects (i.e., $a = (pre(a), eff^-(a), eff^+(a))$). An action a is *applicable* in a state s if and only if $pre(a) \subseteq s$. Application of a in s (if possible) results in a state $(s \setminus eff^-(a)) \cup eff^+(a)$.

In the classical representation atoms are predicates. A *planning operator* $o = (name(o), pre(o), eff^-(o), eff^+(o))$ is a generalized action (i.e. action is a grounded instance of the operator), where $name(o) = op\_name(x_1, \ldots, x_k)$ (op_name is an unique operator name and $x_1, \ldots x_k$ are variable symbols (arguments) appearing in the operator) and $pre(o), eff^-(o)$ and $eff^+(o)$ are sets of (unground) predicates. A *planning problem* is specified via a planning domain, initial state and set of goal atoms. The details can be found in [7].

The set-theoretic representation can be obtained from the classical representation by grounding. Note that comparing predicates (needed for set operations) is done such that predicates are equal if it has the same name and their arguments (including their order) are identical. Hereinafter, we will assume that different operators have different arguments (unless otherwise stated). A *Substitution* $\Theta = \{v_1 \to t_1, \ldots, v_k \to t_k\}$ is a set of mappings where $v_1, \ldots, v_k$ are variable symbols and $t_1, \ldots, t_k$ are terms. Substitutions are used to rename or ground operators' or predicates' arguments. We will use a postfix notation, e.g $o\Theta$ means that a substitution Θ is applied on an operator o.

3. Basic Terminology

By analysing action or operator schema we can identify how these influence each other. As discussed in Chapman's earlier work [3], an action having some atom in its positive

effects is a possible achiever of that atom for some other action having that atom in its precondition. An action achieving an atom for another action in a given plan is a necessary achiever. Note that being 'achiever' refers to a notion "causal link" in plan-space planning. A notion of being a possible achiever can be easily extended for planning operators. Formally:

Definition 1. *Let a_i and a_j be actions. We say that a_i **possibly achieves** an atom p for a_j if and only if $p \in eff^+(a_i) \cap pre(a_j)$.*
*Let o_i and o_j be planning operators and Θ be a substitution. We say that o_i **possibly achieves** an atom (predicate) p for o_j with respect to Θ if and only if $p \in eff^+(o_i) \cap pre(o_j\Theta)$.*
*Let $\langle a_1, a_2, \ldots a_n \rangle$ be a plan. We say that an action a_i **necessarily achieves** (hereinafter achieves only) an atom p for an action a_j if and only if $i < j$, $p \in eff^+(a_i) \cap pre(a_j)$ and $\forall k \in \{i+1, \ldots, j-1\} : p \notin eff^+(a_k)$.* ■

Hereinafter, the well known BlocksWorld domain will be used as a running example we use. It consists of four operators: pickup(?x) — a robotic hand picks-up a block ?x from the table, putdown(?x) — a robotic hand puts-down the block ?x it is holding to the table, unstack(?x,?y) — a robotic hand unstacks a block ?x from ?y, and stack(?x,?y) — a robotic hand stacks a block ?x to ?y.

3.1. Inner Entanglements

Inner Entanglements have been recently introduced as relations between pairs of planning operators and predicates [5]. Inner entanglements stand for operator exclusivity of 'achieving' or 'requiring' predicates. In the BlocksWorld [16] it may be observed, for instance, that operator pickup(?x) achieves predicate holding(?x) exclusively for operator stack(?x,?y) (and not for operator putdown(?x) since it only reverses effects of pickup(?x)). This relation is denoted as an 'entanglement by succeeding'. Similarly, it may be observed that predicate holding(?x) for operator putdown(?x) is exclusively achieved by operator unstack(?x,?y) (and not by operator pickup(?x) since it only reverses effects of putdown(?x)). This relation is denoted as an 'entanglement by preceding'. This is formalized in the following definition [5].

Definition 2. *Let P be a planning problem. Let o_1 and o_2 be planning operators and p be a predicate (o_1, o_2 and p are defined in a planning domain related to P) and Θ be a substitution such that $p \in eff^+(o_1)$ and $p \in pre(o_2\Theta)$.*
*We say that o_1 is **entangled by succeeding** o_2 with p if and only if there exists a plan π solving P and $\forall a_1, a_2 \in \pi$ such that a_1 achieves p_{gnd} (p_{gnd} is a grounded instance of p) for a_2 it holds that if a_1 is an instance of o_1 then a_2 is an instance of o_2.*
*We also say that o_2 is **entangled by preceding** o_1 with p if and only if there exists a plan π solving P and $\forall a_1, a_2 \in \pi$ such that a_1 achieves p_{gnd} (p_{gnd} is a grounded instance of p) for a_2 it holds that if a_2 is an instance of o_2 then a_1 is an instance of o_1.*
*Henceforth, entanglements by preceding and succeeding are denoted as **inner entanglements**.* ■

Informally speaking, inner entanglements provide constraints affecting ordering of operators' instances in solution plans. If an operator o_1 is entangled by a succeeding

operator o_2 with a predicate p in a given planning problem, then in some solution plan instances of o_1 are at some point followed by corresponding instances of o_2 and no corresponding instance of other operator having p in its precondition can be placed in between them. Similarly, if an operator o_2 is entangled by a preceding operator o_1 with a predicate p in a given planning problem, then in some solution plan instances of o_2 are at some point preceded by corresponding instances of o_1 and no corresponding instance of other operator having p in its positive effects can be placed in between them. Note that if there is a plan in which no instance of p is achieved or required by any action, it does not violate the entanglement conditions.

Situations, where an instance of the predicate (e.g. holding(a)) is already present in the initial state and thus not exclusively achieved by an instance of a certain operator (e.g. unstack(a,b)), or is present in the goal state and thus not exclusively achieved for an instance of a certain operator (e.g. stack(a,b)), do not break entanglements according to Definition 2. To enforce the exclusivity of 'providing' and 'requiring' predicates only between given operators, Definition 2 must be strengthened as follows.

Remark 1. *Let P be a planning problem, I be the initial state in P and $\pi = \langle a_1, \ldots, a_n \rangle$ be a plan solving P. Let $a_I = (\emptyset, \emptyset, I)$ and $a_G = (s_G, \emptyset, \emptyset)$ be actions where s_G is a state obtained by executing π in I. Universal quantifier ($\forall a_1, a_2 \in \pi$) used for defining both entanglement by succeeding and preceding can be modified to $\forall a_1, a_2 \in \langle a_I, a_1, \ldots, a_n, a_G \rangle$ (in both cases). Then we say that entanglement by succeeding (or preceding) is **strict**.*

A single (inner) entanglement requires only the existence of one plan solving the given planning problem where the entanglement conditions are met and, therefore, different entanglements might be met in different solution plans. A *set of compatible entanglements* ensures existence of at least one solution plan following all the entanglements in the set [5]. For example, both the BlocksWorld related entanglements mentioned throughout this section forms a set of compatible entanglements. Hereinafter, we will assume that multiple entanglements are a set of compatible entanglements unless stated otherwise.

We can also define *conflicting* entanglements, a pair of entanglements, that cannot be present together in any set of compatible entanglements (of the (set of) problem(s)), i.e., there is no solution plan for the problem (or for some problem from the set of problems) following both the entanglements.

4. Theoretical Properties of Inner Entanglements

In general, deciding inner entanglements is intractable, which will be formally proved in the following theorem. For this purpose we will use the landmark theory [10]. A *landmark* is a proposition (atom) which must become true at some point during execution of every valid solution plan (of a given problem). A *greedy necessary ordering of landmarks* $p \rightarrow_g q$ (p and q are landmarks) refers to situation where in every solution plan p is achieved before q is achieved at the first time. Problems of deciding a landmark or greedy necessary ordering of landmarks is PSPACE-complete [10].

Theorem 1. *Deciding entanglements by succeeding is PSPACE-complete. Deciding entanglements by preceding is PSPACE-complete as well.*

Proof. We reduce the problem of deciding whether landmarks p and q are greedily necessarily ordered, i.e., $p \rightarrow_g q$, which is PSPACE-complete to the problem of deciding entanglements by succeeding or preceding. Without loss of generality, let p and q defined in some problem P be nulary and landmarks. Let O be a set of planning operators defined in P. Let $O_p = \{o \mid o \in O, p \in \mathit{eff}^+(o)\}$ be set of operators achieving p and $O_q = \{o \mid o \in O, q \in \mathit{eff}^+(o)\}$ be set of operators achieving q. We extend the description of P by adding predicates p', p'', q' and q'' (without loss of generality we assume that none of the predicates is defined in P). We modify operators in O_p and O_q as follows. $\forall o \in O_p$: replace p by p' in $\mathit{eff}^+(o)$. $\forall o \in O_q$: add q' into $\mathit{eff}^+(o)$ and add q'' into $\mathit{eff}^-(o)$. The initial state of P is modified by replacing p by p' if p is present in the initial state, and adding q' if q is present in the initial state, or q'' if q is not present in the initial state. Hence, we can observe that q' is true after q has been achieved and q'' is true only before q is achieved (if q is true in the initial state, q'' is never true).

To prove PSPACE-completeness of the problem of deciding entanglements by succeeding we introduce the following operators into the modified problem P (without loss of generality we assume that none of the operators is defined in P), i.e, an operator $o_{p'} = (\{p'\},\{p'\},\{p''\})$, and operators $o_{q'} = (\{p'',q'\},\{p''\},\{p\})$ and $o_{q''} = (\{p'',q''\},\{p''\},\{p\})$. We can observe that if $o_{p'}$ is entangled by succeeding $o_{q'}$ with p'' (in the modification of P), then there exists a solution plan of P where q is achieved before p (if p has to be achieved before q, $o_{q''}$ must be applied which is in the contradiction with the entanglement). Hence, $o_{p'}$ is entangled by succeeding $o_{q'}$ with p'' if and only if $p \rightarrow_g q$ does not hold. So, the problem of deciding entanglements by succeeding is co-PSPACE-complete = PSPACE-complete.

To prove PSPACE-completeness of the problem of deciding entanglements by preceding we introduce the following operators into the modified problem P (without loss of generality we assume that none of the operators is defined in P), i.e, operators $o_{p'} = (\{p',q'\},\{p'\},\{p''\})$ and $o_{p''} = (\{p',q''\},\{p'\},\{p''\})$, and an operator $o_{q'} = (\{p''\},\{p''\},\{p\})$. We can observe that if $o_{q'}$ is entangled by preceding $o_{p'}$ with p'' (in the modification of P), then there exists a solution plan of P where q is achieved before p (if p has to be achieved before q, $o_{p''}$ must be applied which is in the contradiction with the entanglement). Hence, $o_{q'}$ is entangled by preceding $o_{p'}$ with p'' if and only if $p \rightarrow_g q$ does not hold. So, the problem of deciding entanglements by preceding is co-PSPACE-complete = PSPACE-complete.

□

Deciding whether a pair of (inner) entanglements is conflicting is PSPACE-complete as well. Since we can reformulate planning problems in order to enforce inner entanglements (see Section 5), we can apply the previous theorem for the reformulated problem where one of the entanglements is involved.

Despite the complexity results we can identify (inner) entanglements in some trivial cases, for instance, if there is only one operator achieving some predicate, or if there is only one operator having some predicate in its precondition [5]. These trivial cases, however, do not lead to pruning of some unpromising alternatives in the search space, since the exclusivity of predicate 'achievement' or 'requirement' what inner entanglements capture is trivially met by existence of only one predicate achiever or 'requirer'. On the other hand, we believe that it is possible identify some non-trivial inner entanglements in polynomial time. For instance, the example of the BlocksWorld domain mentioned before we can observe that after picking up the block from the table we can either stack

it on some other block or put it down on the table. We can easily find out that putting the block down on the table after picking it up from the table results in the same state as before the block is picked up and hence applying such actions consequently is meaningless. From this observation we can derive the entanglement by succeeding between the pickup and stack operators.

4.1. Approximative Detection of Inner Entanglements

Because of general intractability, we can use an approximation method for detecting compatible sets of inner entanglements which has recently been published [5]. This method analyses a set of training plans, solutions of simpler planning problems, in order to identify a set of compatible (inner) entanglements which holds for every training problem; it is then assumed that this set of compatible (inner) entanglements holds for a whole class of planning problems using the same domain model. Although such an approach is generally incomplete it has been empirically shown in [5] and also will be shown in this paper that on IPC benchmarks only in a very few cases incorrect entanglements have been detected.

5. Problem Reformulation

To exploit entanglements by any existing planning engine, the original domain and problem models must be reformulated as discussed in [5].

Let P be a planning problem and an operator o_1 is entangled by a succeeding operator o_2 with a predicate p. Then the problem P is reformulated as follows:

1. Create a predicate p' (not defined in the domain of P) having the same arguments as p and add p' to the domain of P.
2. Modify the operator o_1 by adding p' into its negative effects. p' has the same arguments as $p \in \mathit{eff}^{+}(o_1)$.
3. Modify the operator o_2 by adding p' into its positive effects. p' has the same arguments as $p \in \mathit{pre}(o_2)$.
4. Modify all operators o such that $o \neq o_2$ and o is a possible achiever for o_1 by p by adding p' into its precondition. p' has the same arguments as $p \in \mathit{pre}(o)$.
5. Add all possible instances of p' into the initial state of P and if the entanglement is strict, then also to the goal situation of P.

Let P be a planning problem and an operator o_2 is entangled by a preceding operator o_1 with a predicate p. Then the problem P is reformulated as follows:

1. Create a predicate p' (not defined in the domain of P) having the same arguments as p and add p' to the domain of P.
2. Modify the operator o_1 by adding p' into its positive effects. p' has the same arguments as $p \in \mathit{eff}^{+}(o_1)$.
3. Modify the operator o_2 by adding p' into its precondition and negative effects. p' has the same arguments as $p \in \mathit{pre}(o_2)$.
4. Modify all operators o such that $o \neq o_2$ and $p \in \mathit{eff}^{-}(o)$ by adding p' into its negative effects. p' has the same arguments as p.

5. Modify all operators o such that $o \neq o_1$ and $p \in \mathit{eff}^+(o)$ by adding p' into its negative effects (p' has the same arguments as p).
6. If the entanglement is not strict, then i) add all possible instances of p' to the initial state of P.

There are also situations where both the (strict) entanglements by preceding and succeeding hold for operators o_1, o_2 and a predicate p. Of course, we can reformulate the problem according to previous reformulation approaches. On the other hand, it requires more supplementary predicates which might not be very desirable. Therefore, we introduce a more compact reformulation approach for such situations. Let P be a planning problem, o_1 is strictly entangled by succeeding o_2 with p, and o_2 is strictly entangled by preceding o_1 with p. Then the problem P is reformulated as follows:

1. Create a predicate p' (not defined in the domain of P) having the same arguments as p and add p' to the domain of P.
2. Modify the operator o_1 by replacing p by p' in o_1's positive effects.
3. Modify the operator o_2 by replacing p by p' in o_1's precondition and (possibly) negative effects.

Given the strict entanglement relations by preceding and succeeding between o_1, o_2 and p, we can see that there is no other operator which either achieves p for o_2 or requires p from o_1. Replacing p by a new predicate p' (having the same arguments) in o_1's positive effects and o_2's precondition enforces the entanglements, but does not affect solvability of the problem.

6. Experimental Evaluation

The aim of the experimental evaluation is to demonstrate how different types of inner entanglements and different domain/problem reformulation strategies influence the planning process. For evaluation purposes we chose several IPC benchmark domains (typed strips) from IPC-3, IPC-6 and 7 (learning track), where it was clear that this kind of reformulation would be applicable (for example, it would not be applicable to domains with one operator). The domains are BlocksWorld (BW), Depots, Zeno, DriverLog, GoldMiner, Parking and Matching-BW. As benchmarking planners we chose Metric-FF [8], LPG-td [6], Probe [12], LAMA 2011 [13], SatPlan [11] (using SAT-MAX-PLAN encoding [15] and the Precosat [1] SAT solver) and Mp [14]. All the planners successfully competed in the IPCs. Timeout was set to 900 seconds, as in IPC learning tracks. The experiment was performed on Intel Xeon™ 3 GHz, 2 GB RAM. For each benchmark we selected 5-7 easy problems as training problems and produced training plans by Metric-FF which were used to learn inner entanglements and generate macros from them. Time spent on learning was in the order of tenths of seconds per one domain.

Cumulative results of the evaluation are presented in Table 1, with the original problem formulation compared to the existing reformulation technique of inner entanglements, considering both the kind of entanglements, only entanglements by preceding and only by succeeding. Values are computed according to rules used in IPC-7 learning track.[3] Score for every solved problem is computed according to the formula

[3] http://www.plg.inf.uc3m.es/ipc2011-learning/Rules

Planner	Model	BW (60)	Depots (60)	Zeno (20)	DriverLog (20)	GoldM (60)	Parking (60)	MatchingBW (60)
Metric-FF	Orig	0.0	17.7	17.6	**17.2**	20.1	**16.9**	**20.6**
	IE	0.0	20.3	17.8	15.9	29.2	15.1	0.4
	ES	0.0	20.1	**19.7**	–	43.0	–	0.4
	EP	0.0	**22.7**	16.0	15.9	**54.4**	15.1	19.5
LPG	Orig	26.8	30.8	**19.2**	**18.5**	44.0	0.0	26.0
	IE	**59.7**	**42.6**	12.9	18.4	42.4	0.0	23.4
	ES	47.6	32.8	13.1	–	**57.5**	–	**29.9**
	EP	35.4	26.7	17.5	18.4	56.9	0.0	19.1
Probe	Orig	39.1	**58.5**	16.9	**19.6**	35.4	10.3	20.0
	IE	47.0	36.5	18.6	17.9	33.6	**12.3**	23.0
	ES	**49.4**	55.3	**19.6**	–	53.1	–	16.6
	EP	17.8	37.3	15.0	17.9	**60.0**	**12.3**	**23.4**
LAMA	Orig	41.1	15.5	18.4	19.1	23.0	**9.8**	**42.3**
	IE	23.8	22.4	18.9	19.1	53.6	7.6	10.1
	ES	**42.3**	23.7	**19.7**	–	55.2	–	9.9
	EP	17.5	**27.1**	17.8	19.1	**56.3**	7.6	39.8
SatPlan	Orig	0.0	7.9	14.0	**14.8**	**59.5**	0.0	**37.0**
	IE	0.0	8.1	**15.3**	13.4	58.9	0.0	30.0
	ES	0.0	8.4	15.2	–	58.5	–	29.5
	EP	0.0	**8.7**	14.6	13.4	59.1	0.0	0.0
Mp	Orig	0.0	30.8	18.8	18.5	**59.8**	**5.0**	0.0
	IE	0.0	**49.6**	19.3	**18.8**	32.6	3.2	0.6
	ES	0.0	42.9	**19.9**	–	44.6	–	**2.0**
	EP	0.0	33.1	18.6	18.5	54.4	3.2	0.5

Table 1. Time score (max score per domain is shown in brackets) on selected domains. Values are computed by considering each planner separately. "–" indicates that no entanglements of that type were generated. Orig – Original domain model, IE – Both types of Inner Entanglements, EP – Entanglements by Preceding, ES – Entanglements by Succeeding.

$(1/(1+\log_{10}(T/T^*)))$. T is the running time of the certain planner for a certain (original or reformulated) problem and T^* is the minimum running time achieved by a certain planner on either original problem or any of its reformulation. The score for unsolved problems is zero.

The technique of using inner entanglements to reformulate domains can reduce the search branching factor, but does so at the cost of introducing supplementary predicates. Good results were achieved for this technique in the Gold-miner domain because supplementary predicates had only a few instances. Contrary to this, many supplementary predicates were needed in DriverLog, which caused poor performance for the planners generally. In the previous work [5] only both types of entanglements have been considered for experiments, while in this paper entanglements by preceding and succeeding were considered individually. In about half of cases, using just one type of inner entanglements brought the best results.

Metric-FF, Probe and LAMA usually achieve best results while exploiting either preceding or succeeding inner entanglements. In the Metric-FF case, entanglements by preceding usually perform better than entanglements by succeeding or both of them. The reason is in the FF heuristic used by Metric-FF which is based on delete relaxation (all negative effects are omitted while the heuristic value is being computed). Entanglements by succeeding are enforced by removing a supplementary predicate (when the first operator is executed) to prevent applicability of 'unwanted' operators (those which are not in the entanglement relation). Hence, the FF heuristic cannot efficiently propagate entanglements by succeeding. In the Probe and LAMA case results are mixed, however, using both types of entanglements never outperforms using only one type of them (ei-

	Metric-FF		LPG		Probe		LAMA		SatPlan		Mp	
Domain	IE	nIE	IE	nIE	IE	nIE	IE	nIE	IE	nIE	IE	nIE
BW	0.0	0.0	**57.5**	50.7	41.6	**50.0**	28.3	**44.2**	0.0	0.0	0.0	0.0
GoldM	31.2	**60.0**	45.8	**59.3**	37.4	**60.0**	53.5	**60.0**	58.2	**60.0**	35.0	**60.0**
Depots	20.6	**26.3**	37.6	**47.1**	35.7	**60.0**	21.9	**34.0**	8.1	**10.0**	43.2	**46.6**

Table 2. Time IPC score on selected domains. Values are computed by considering each planner separately. IE – existing (old) encoding, nIE – New encoding.

ther entanglements by preceding or succeeding). This might be caused by a significant increase of the number of atoms considered during the search. SatPlan and Mp, even if both are based on SAT, have different behaviours. SatPlan usually has better performance on original domain models, while Mp is able to exploit the reformulated version of domain models that include both the type of entanglements, or only preceding ones. We believe that the SAT-MAX-PLAN encoding strategy used by SatPlan for translating the planning problem in a SAT formula is more sensitive to the increase of the number of atoms than the one used by Mp. The other encoding strategies that are included in the SatPlan framework are generally generating too large formulas for most of the considered testing problems, leading to a dramatically small number of solved problems. LPG's performance are usually best while exploiting both the preceding and succeeding entanglements or only preceding ones (except in Zeno and DriverLog), intuitively we can argue that this is due to the fact that LPG uses greedy local search on the Planning Graph, hence the reduction of the branching factor improves its performance.

Regarding the quality, in terms of number of involved actions of solution plans, we noticed that the exploitation of inner entanglements by the planners do not result in significant modification of the plans quality. This is interesting, since inner entanglements were designed for improving the performance of planners by reducing the branching factor. Given these results, we can derive that by exploiting inner entanglements we are able to improve the runtime of planners, in most of the considered domains, without decreasing the quality of solution plans.

Table 2 shows the results of the comparison between Both types of Inner Entanglements, inner entanglements encoded using the existing technique, and New encoding, which are encoded using the new technique proposed in this paper. These techniques lead to different domain reformulations in three of the considered domains, namely BlocksWorld, Gold-Miner and Depots. The results clearly indicate that the new encoding are able to further improve the performance of the considered planners. The main reason for such improvement is in the significantly smaller number of atoms considered during the search.

7. Conclusions

In this paper, we revisited and extended the work on inner entanglements, relations capturing the exclusivity of predicate 'achievement' and 'requirement' between planning operators [5]. We formally proved PSPACE-completeness of deciding both types of inner entanglements, entanglements by preceding and succeeding. We also proposed a new compact encoding for situations where a pair of operators is in both types of (strict) inner entanglements. We empirically showed that the proposed compact encoding outper-

forms the existing one. We also compared how different types of inner entanglements influence the planning process. The results showed that in many cases, using just one type of inner entanglement was the best option. This result is interesting, since it points to the problem of utility of different inner entanglement relations. Even though inner entanglements prune some unpromising alternatives in the search space, introducing supplementary predicates causes overheads in the planning process. Therefore, a given inner entanglement relation is useful only if the overheads are smaller than the time saved by avoiding exploration of unpromising search alternatives.

In future, we will investigate in which situations deciding inner entanglements can be tractable. We will also investigate how to efficiently estimate the utility of particular inner entanglement relations, since it might significantly improve the planning process.

Acknowledgements The research was funded by the UK EPSRC Autonomous and Intelligent Systems Programme (grant no. EP/J011991/1). The authors would like to acknowledge the use of the University of Huddersfield Queensgate Grid in carrying out this work.

References

[1] A. Biere. P{re,i}cosat@sc'09. In *SAT Competition 2009*, 2009.

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

[3] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.

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

[5] L. Chrpa and T. L. McCluskey. On exploiting structures of classical planning problems: Generalizing entanglements. In *Proceedings of ECAI*, pages 240–245, 2012.

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

[7] M. Ghallab, D. Nau, and P. Traverso. *Automated planning, theory and practice*. Morgan Kaufmann Publishers, 2004.

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

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

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

[11] H. Kautz, B. Selman, and J. Hoffmann. Satplan: Planning as satisfiability. In *Proceedings of the fifth IPC*, 2006.

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

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

[14] J. Rintanen. Engineering efficient planners with sat. In *Proceedings of ECAI*, pages 684–689, 2012.

[15] A. Sideris and Y. Dimopoulos. Constraint propagation in propositional planning. In *Proceedings of ICAPS*, pages 153–160, 2010.

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