Determining Linearity of Optimal Plans by Operator Schema Analysis

Lukáš Chrpa and **Mauro Vallati** and **Thomas Leo McCluskey**
PARK Research Group
School of Computing and Engineering
University of Huddersfield
{l.chrpa, m.vallati, t.l.mccluskey}@hud.ac.uk

Abstract

Analysing the structures of solution plans generated by AI Planning engines is helpful in improving the generative planning process, as well as shedding light in the study of its theoretical foundations. We investigate a specific property of solution plans, that we called linearity, which refers to a situation where each action achieves an atom (or atoms) for a directly following action, or achieves goal atom(s). Similarly, linearity can be defined for parallel plans where each action in a set of actions executed at some time step, achieves either goal atom(s) or atom(s) for some action executed in the directly following time step. In this paper, we present a general and problem-independent theoretical framework focusing on the analysis of planning operator schema, namely relations of achiever, clobberer and independence, in order to determine whether solvable planning problems using a given operator schema have as solutions optimal (parallel) plans which are linear. The findings presented in this paper deepen current theoretical knowledge, provide helpful information to engineers of new planning domain models, and suggest new ways of improving the performance of state-of-the-art (optimal) planning engines.

Introduction

Automated planning, despite being intractable in general (Bylander 1994), is very useful in many real-world applications (e.g. Space exploration, Manufacture planning etc.). Therefore, there is a need for novel approaches to inform heuristic design, as well as deepening theoretical knowledge. Many promising domain-independent techniques have been developed in recent years, for example, heuristic search based planners such as FF (Hoffmann and Nebel 2001) or LAMA (Richter and Westphal 2010), SAT based planners (Kautz, Selman, and Hoffmann 2006; Rintanen 2012) or CSP based planners (Do and Kambhampati 2001).

Complexity results of well known benchmark domains showed that some of these domains are tractable (Helmert 2003; 2006). Despite tractability of some domains, for instance the well known BlocksWorld domain (Slaney and Thiébaux 2001), state-of-the-art planning engines tend to struggle on them. However, when considering domain-independent planners, it has proven useful to focus on analysing structures of planning domains and problems. One of such structures is the Causal Graph (Knoblock 1994) which describes dependencies between state variables in the SAS+ representation (Bäckström and Nebel 1995). The Causal Graph has been studied in terms of computational complexity (Gimenez and Jonsson 2007; Katz and Domshlak 2007) as well as for developing heuristics (Richter and Westphal 2010). Analysing mutex relations between actions, which were exploited in GraphPlan (Blum and Furst 1997), has been studied in order to determine classes of tractable planning problems (Surynek 2008). Impact of various restrictions (e.g. number of atoms in actions' preconditions etc.) on computational complexity has been studied in both STRIPS (Bylander 1994) and SAS+ (Bäckström and Nebel 1995) representation. Recent work (Bäckström et al. 2012) shows that planning under certain restrictions can be tractable in terms of parametrised complexity.

Analysing structures of planning problems is an established area. It is good to mention TIM (Fox and Long 1998), a tool for analysing inference of state invariants, which has been used within the STAN planner (Fox and Long 2001). Structural analysis of planning problems can be also useful in order to prune some unpromising alternatives during the search. Expansion Cores (Chen and Yao 2009) aim to restricting on relevant Domain Transition Graphs rather than all of them in the node expansion phase. Other work (Coles and Coles 2010) prunes irrelevant actions, i.e., actions that change 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 we have to consecutively execute specific actions forming the 'tunnel'.

In this paper we introduce a property of the set of solution plans to a planning problem, called *linearity*. This refers to a situation where there exists an optimal plan in the set of solution plans, which is *linear*, meaning that all actions in the plan achieve an atom (or atoms) for a directly following action, or achieve goal atom(s). The property generalises to parallel plans, where one step may contain a set of independent actions. We provide a theoretical analysis of relations between planning operators (considering the STRIPS representation) in terms of whether one operator is an achiever, is a clobberer, or is independent of another. We use this analysis to determine sets of conditions under which solution plans (both sequential and parallel) of problems within a domain model can be said to satisfy the linearity property. We relate our findings to other theoretical work in the area, and show how the property can provide helpful information to engineers of new planning domain models. Finally, we show how the property can be operationalised as a planning heuristic, for use with planning problems satisfying the linearity property.

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 (Ghallab, Nau, and Traverso 2004). Alternatively, instead of using sequences of (single) actions we may use sequences of sets of actions where actions in each set are independent and can be executed simultaneously or without a predefined order.

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., $a = (\mathit{pre}(a), \mathit{eff}^-(a), \mathit{eff}^+(a))$) and $\mathit{eff}^-(a) \cap \mathit{eff}^+(a) = \emptyset$. An action a is *applicable* in a state s if and only if $\mathit{pre}(a) \subseteq s$. Application of a in s results in a state $(s \setminus \mathit{eff}^-(a)) \cup \mathit{eff}^+(a)$ if a is applicable in s, otherwise the result of the application is undefined. Actions a_1 and a_2 are *independent* if and only if $(\mathit{eff}^-(a_1) \cap (\mathit{pre}(a_2) \cup \mathit{eff}^+(a_2)) = \emptyset) \wedge (\mathit{eff}^-(a_2) \cap (\mathit{pre}(a_1) \cup \mathit{eff}^+(a_1)) = \emptyset)$. A set of independent actions A_x is applicable in a state s if and only if $\bigcup_{a \in A_x} \mathit{pre}(a) \subseteq s$. Application of actions in a set of independent actions A_x in s results in a state $(s \setminus \bigcup_{a \in A_x} \mathit{eff}^-(a)) \cup \bigcup_{a \in A_x} \mathit{eff}^+(a)$ if A_x is applicable in s (undefined otherwise).

In the classical representation atoms are predicates. A *planning operator* $o = (\mathit{name}(o), \mathit{pre}(o), \mathit{eff}^-(o), \mathit{eff}^+(o))$ is a generalised action (i.e. an action is a grounded instance of the operator), where $\mathit{name}(o) = \mathit{op\_name}(x_1, \ldots, x_k)$ (*op_name* is an unique operator name and $x_1, \ldots x_k$ are variable symbols (arguments) appearing in the operator description) and $\mathit{pre}(o), \mathit{eff}^-(o)$ and $\mathit{eff}^+(o)$ are sets of (ungrounded) predicates. The set-theoretic representation can be obtained from the classical representation by grounding. Two predicates are *equal* if and only if they have the same name and identical arguments (including their order). Determining equality of predicates is important for set operations which will be used on the top of sets of ungrounded predicates (e.g. operators' preconditions or effects). Hereinafter, we will assume that different operators have different arguments (unless otherwise stated). Determining which operators' arguments are shared (in other words, having the same variable symbols) is done by substitutions.

Definition 1. *A Substitution is a set of mappings from variable symbols to terms.*
We say that a substitution $\Theta = \{v_1 \rightarrow v'_1, \ldots, v_k \rightarrow v'_k\}$ is relevant w.r.t. planning operators o_1, o_2 if and only if all the variable symbols $v_1, \ldots, v_k$ are different and defined in $\mathit{name}(o_2)$ and all the variable symbols $v'_1, \ldots, v'_k$ are defined in $\mathit{name}(o_1)$.
An inverse substitution to Θ is denoted as Θ^{-1} and is obtained from Θ by swapping all the variable symbol mappings. ■

In other words, relevant substitutions are used to determine which arguments (variable symbols) operators share. For example, having operators unstack(?x ?y) and put-down(?z) substitutions $\emptyset, \{?z \rightarrow ?x\}, \{?z \rightarrow ?y\}$.

A *planning domain* is specified via sets of predicates and planning operators (alternatively propositions and actions). A *planning problem* is specified via a planning domain, initial state and set of goal atoms. A *plan* is a sequence of actions. A *parallel plan* is a sequence of sets of independent actions. A (parallel) plan is a *solution* of a planning problem if and only if a consecutive application of the (sets of independent) actions in the plan (starting in the initial state) results in a state, where all the goal atoms are satisfied. A solution π of a given problem is *optimal* if for any solution π' of the given problem $|\pi| \leq |\pi'|$ (length of the parallel plan is determined by the number of sets of independent actions).

Relation between Planning Operators

An insight into how actions in plans can be ordered is given by the fact that some planning operator might achieve an atom (a predicate) which is required by another planning operator. Following Chapman's terminology (Chapman 1987) we define the relation of being an achiever defined between planning operators in the following way.

Definition 2. *Let o_1 and o_2 be planning operators. We say that o_1 is an* ***achiever*** *for o_2 w.r.t a relevant substitution Θ if and only if $\mathit{eff}^+(o_1) \cap \mathit{pre}(o_2\Theta) \neq \emptyset$. We denote this as $o_1 \rhd_\Theta o_2$.* ■

Analogously, planning operators influence each other in a negative way, that is, one operator might 'clobber' an atom (or atoms) required by another operator. We define the relation of being a clobberer defined between planning operators in the following way.

Definition 3. *Let o_1 and o_2 be planning operators. We say that o_1 is a* ***clobberer*** *for o_2 w.r.t a relevant substitution Θ if and only if $\mathit{eff}^-(o_1) \cap \mathit{pre}(o_2\Theta) \neq \emptyset$. We denote this as $o_1 \triangledown_\Theta o_2$.* ■

We extend the independence relation which we have already defined for actions (see the previous section) also for planning operators.

Definition 4. *Let o_1 and o_2 be planning operators. We say that o_1 is* ***independent*** *on o_2 w.r.t a relevant substitution Θ if and only if $\mathit{eff}^-(o_1) \cap (\mathit{pre}(o_2\Theta) \cup \mathit{eff}^+(o_2\Theta)) = \emptyset$ and $\mathit{eff}^-(o_2\Theta) \cap (\mathit{pre}(o_1) \cup \mathit{eff}^+(o_1)) = \emptyset$. We denote this as $o_1 \diamond_\Theta o_2$.* ■

Keep in mind that the above relations might hold with respect to more (relevant) substitutions even for a single pair of operators.

Linearity of Solution Plans

Achievers or clobberers can be analogously defined for actions as well. When analysing solution plans, we may observe that some action provides atoms which are preconditions for some other actions. Hence, a possible achiever becomes necessary achiever in a given plan (Chapman 1987). Analysing plans by exploring a relation of necessary achievement between actions has already been successfully applied in post-planning optimisation (Chrpa, McCluskey, and Osborne 2012) or macro-operator generation (Chrpa 2010). However, instead of doing post-planning plan analysis it may be useful to determine specific properties of optimal plans in advance by analysing planning domains, which can be useful in improving the planning process, for instance, by pruning some unpromising alternatives. An interesting property of a plan is when an action achieves goal atom(s) or atom(s) to the immediately following action. Such a plan is denoted as *linear*.

Definition 5. *Let $\pi = \langle a_1, \ldots, a_n \rangle$ be a plan, a solution of some problem P. We say that π is* ***linear*** *if and only if for every i such that $1 \leq i \leq n$ is the case that $\mathit{eff}^+(a_i) \cap \mathit{pre}(a_{i+1}) \neq \emptyset$ or $(\mathit{eff}^+(a_i) \setminus \bigcup_{j=i+1}^{n} \mathit{eff}^+(a_j)) \cap g \neq \emptyset$ (g is a set of goal atoms defined in P).* ■

Similarly, we can introduce linearity for parallel plans.

Definition 6. *Let $\pi = \langle A_1, \ldots, A_n \rangle$ be a parallel plan, a solution of some problem P. We say that π is* ***linear*** *if and only if for every i such that $1 \leq i \leq n$ is the case that for every $a \in A_i$ there is $a' \in A_{i+1}$ such that $\mathit{eff}^+(a) \cap \mathit{pre}(a') \neq \emptyset$ or $(\mathit{eff}^+(a) \setminus \bigcup_{j=i+1}^{n} \bigcup_{a' \in A_j} \mathit{eff}^+(a')) \cap g \neq \emptyset$ (g is a set of goal atoms defined in P).* ■

Analysing Planning Operator Schema

Structures that are formed by the achiever ($\triangleright$), clobberer ($\triangledown$) or independent ($\diamond$) relations provide an abstract insight into how optimal (parallel) solution plans might look like. We can identify when there exists an optimal solution plan which is linear. Intuitively, it is when an operator is either achiever or clobberer for another operator (including itself) with respect to all relevant substitutions. We formalise it in the following theorem.

Theorem 1. *Let O be the set of planning operators defined in some planning domain Σ. Assume that for every $o_1, o_2 \in O$ and every relevant substitution Θ is the case that $o_1 \triangleright_\Theta o_2$ or $o_1 \triangledown_\Theta o_2$. Then, for every solvable planning problem P defined over Σ, there exists an optimal solution plan which is linear.*

Proof. Let a, a' be instances of operators o, o'. For any solution plan of P, a can be followed by a' only if a does not delete any of the preconditions of a', otherwise the plan is not well formed. From the assumption we can deduce that for any relevant substitution ξ, if not $o \triangledown_\xi o'$, then $o \triangleright_\xi o'$. It follows that for such an action a' following a in any solution plan, is the case that a achieves atom(s) for a', hence the linearity condition is met. □

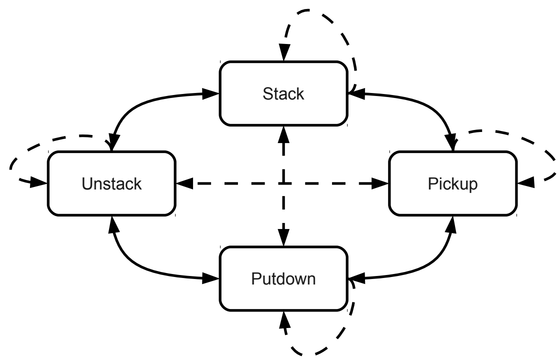

Figure 1: The BlocksWorld planning domain, enhanced with the handfull atom. Full arrows indicate achiever relations, dashed arrows indicate clobberer relations between the involved operators.

Recall the well known BlocksWorld domain (Slaney and Thiébaux 2001) with 4 planning operators: pickup(?b), putdown(?b), unstack(?b1 ?b2) and stack(?b1 ?b2). There is only one robotic hand which can carry blocks. Hence, an atom handempty is in preconditions of pickup(?b) and unstack(?b1 ?b2) and in positive effects of putdown(?b) and stack(?b1 ?b2). We can also use atom handfull in the other way round (at each point of the planning process either handempty or handfull is true). Note that the atom handfull is not present in the original domain but it is used for 'highlighting' the fact that in well-defined problems only one block can be held by the robotic hand at the same time. Otherwise, if a robotic hand initially holds more than one block, then, for instance, putdown(?b) is neither an achiever nor clobberer for stack(?b1 ?b2) w.r.t. empty substitution. If the supplementary handfull is used we can see that, for instance, pickup(?b) is an achiever for stack(?b1 ?b2) and putdown(?b) (w.r.t. every relevant substitution), and a clobberer for unstack(?b1 ?b2) and itself (w.r.t. every relevant substitution). Similar observations can be done for the other operators. Hence, we can see that according to Theorem 1 we can find optimal solution plans which are linear. In Figure 1 the structure of the BlocksWorld domain, enhanced with the handfull atom is shown.

However, we can straightforwardly deduce that for situations described by Theorem 1 it holds that every solution plan follows the linearity conditions. Therefore, it does not bring any improvement to the planning process since we cannot prune any of alternatives during the search. On the other hand, it might reveal that for some problems the goal atoms have to be achieved in a certain order, which is the case of BlocksWorld.

It might be observed that different achievers for a given operator might achieve the same predicate(s) or the given operator needs more achievers for its precondition. Hence, we can consider *OR achievers* which stand for different (sets of) pairs (operator, substitution) such that we need only one pair (or set) to achieve atoms for a certain operator. For example, the putdown(?b) operator requires a predicate holding(?b) (we do not consider the atom handfull in this example). holding(?b) can be achieved by unstack(?b ?b2) or pickup(?b). unstack(?b ?b2) and pickup(?b) can be therefore understood as OR achievers. We can consider also *AND* achievers which stand for a set of pairs (operator, substitution) such that we need all the pairs to achieve atoms for a certain operator. For example, the stack(?b1 ?b2) operator requires predicates holding(?b1) and clear(?b2). These predicates can be achieved by pickup(?b1) and unstack(?b ?b2) and therefore these operators (with the corresponding substitutions) can be understood as AND achievers. Formally:

Definition 7. *Let o be a planning operator. Let $\mathcal{A}_o = \{(o', \Theta) \mid o' \rhd_\Theta o\}$ be a set of o's achievers. We say that $\mathcal{A}_o^+$ is a **sufficient set of** o**'s achievers** if and only if $\mathcal{A}_o^+ \subseteq \mathcal{A}_o$ and $\bigcup_{(o',\Theta)\in\mathcal{A}_o^+}(\mathit{eff}^+(o'\Theta^{-1}) \supseteq \mathit{pre}(o)$. We say that a sufficient set o's of achievers $\mathcal{A}_o^*$ is **compact** if and only if $|\mathcal{A}_o^*| \leq |\mathit{pre}(o)|$ and there exists a total mapping $\chi : \mathit{pre}(o) \rightarrow \mathcal{A}_o^*$ such that $\chi(p) = (o', \Theta)$ if and only if $p \in \mathit{eff}^+(o\Theta^{-1}) \cap \mathit{pre}(o)$. We say that all the possible compact sets of o's achievers are o's **OR sets of achievers**. Elements in $\mathcal{A}_o^*$, which is a compact set of o's achievers, are o's **AND achievers**.* ■

The above definition might be intuitively understood as some sort of DNF where literals are couples (operator, substitution) representing achievers.

The above definition also considers that all precondition predicates for a given operator must be achieved by other operators. It is not always true, since some of the predicates might be only achieved by an initial state. However, such predicates only limit the number of instances of the operators (for static predicates) or the number of instances (if such predicates appear only in preconditions and negative effects). Static predicates are not important for our analysis because they do not affect the achiever, clobberer and independence relations. However, the latter kind of predicates might influence clobberer and independence relations. Since, the achiever relation is not affected by such predicates we will not consider them while determining OR and AND achievers.

Situations where each operator in a given operator set does not need more than one achiever intuitively leads to the conclusion that using such an operator set leads to an existence of linear optimal solution plans. However, clobberers influence action ordering in plans but if operators delete only predicates which are in their preconditions, then as the following theorem shows that linearity is not affected.

Theorem 2. *Let O be the set of planning operators defined in some planning domain Σ. Assume the following conditions:*

1) For every $o \in O$, o's OR sets of achievers $\mathcal{A}_o^{*1}, \ldots, \mathcal{A}_o^{*k}$ are the case that $\forall i \in \{1, \ldots, k\} : |\mathcal{A}_o^{*i}| \leq 1$.

2) For every $o \in O$, $\mathit{eff}^-(o) \subseteq \mathit{pre}(o)$.

Then, for every solvable planning problem P defined over Σ, there exists an optimal solution plan which is linear.

Proof. To achieve a precondition for any instance of any operator $o \in O$ we need at most one action, an instance of some operator from O. This is because according to the condition 1) any of o's OR sets of achievers contains at most one element. For problems with a single goal atom, we can perform the backward search where we keep selecting nondeterministically an achiever for an operator the current action is an instance, until there is no achiever (the current action has an empty precondition), or the initial state has been reached. Clearly, we can find optimal plans which are linear.

For problems with more goal atoms, we can nondeterministically choose the order in which the goal atoms must be achieved in order to sustain optimality of the solutions. However, 'subplans' (each 'subplan' achieves some goal atom(s)) must not be interleaving otherwise linearity might be violated. From this perspective we have to investigate clobberers. From the assumption (both the conditions 1) and 2)) we can see that $(o'' \rhd_\Theta o \wedge o \nabla_{(\Theta'\Theta^{-1})} o') \Rightarrow o'' \rhd_{\Theta'} o'$ (note that o may be a clobberer for itself as well). In other words, an operator (o) might be clobberer to another operator (o') only if they have the same achiever (o''). Let o_k be an operator such that $o_k \rhd_{\Theta_k^1} o_k^1$ and $o_k \rhd_{\Theta_k^2} o_k^2$. If both o_k^1 and o_k^2 are clobberers to each other, then we cannot execute o_k^2 after o_k^1 and vice versa unless o_k (or some other achiever is executed). Hence, we have to achieve goal(s) by a 'subplan' containing o_k^1 (or o_k^2) and after that we can execute a 'subplan' containing o_k^2 (or o_k^1). So, linearity is not affected. If both o_k^1 and o_k^2 are not clobberers to each other, then we might execute o_k^2 after o_k^1 and vice versa. However, executing the whole 'subplan' containing o_k^1 (or o_k^2) cannot delete atom(s) required by o_k^2 (or o_k^1). This is because according to the assumption if some operator is a clobberer for o_k^2 (or o_k^1), then o_k is its achiever. Thus no instance of such operator can occur after o_k^1 (or o_k^2) since otherwise the solution plan containing such a 'subplan' is not optimal. Hence, we can execute the 'subplan' containing o_k^1 (or o_k^2) and after that we can execute a part of the 'subplan' commencing o_k^2 (or o_k^1). So, linearity is not affected as well. If o_k^1 is a clobberer for o_k^2 but not the other way round, then from the above we can deduce that linearity is also not affected. Note that in this case the 'subplan' containing o_k^2 is executed before a part of the 'subplan' commencing o_k^1. This can be analogously extended to situations where o_k is an achiever for more than two operators. □

The previous theorem concerns situations where we do not need more than one action in order to achieve atoms needed by another action. A specific situation of this kind, where operators have at most one atom (predicate) in their preconditions, is formalised in the following corollary.

Corollary 1. *Let O be the set of planning operators defined in some planning domain Σ. If $\forall o \in O : |\mathit{pre}(o)| \leq 1 \wedge \mathit{eff}^-(o) \subseteq \mathit{pre}(o)$, then every solvable planning problem P defined over Σ has an optimal solution plan which is linear.*

Proof. Straightforwardly, if $|\mathit{pre}(o)| \leq 1$ for any operator o, then none of o's OR sets of achievers can contain more than one element. □

Theorem 2 can be also applied for optimal parallel plans, which is proved in the following corollary.

Corollary 2. *Following the assumption from Theorem 2 it also holds that for every solvable planning problem P defined over Σ, there exists a linear optimal parallel plan which is a solution of P.*

Proof. Following the proof of Theorem 2 we have to consider executing 'subplans' (sequences of actions achieving particular goal atoms) in parallel. Clearly, there are only two possibilities where operators o and o' may not be independent w.r.t. some substitution ξ. That is, if $o \nabla_\xi o'$ (or vice versa), or if $o \rhd_\xi o'$ (or vice versa). If $o \rhd_\xi o'$ (or vice versa), then, straightforwardly, instances of o and o' can be placed in adjacent action sets. This does not violate linearity nor optimality. For the clobberer situations, we recall the situation from the proof of Theorem 2. Let o_k be an operator such that $o_k \rhd_{\Theta_k^1} o_k^1$ and $o_k \rhd_{\Theta_k^2} o_k^2$. If neither o_k^1 nor o_k^2 is a clobberer for each other, then they are independent and can be present in the same set of actions. Hence, after a prefix which is the same for both 'subplans' the rest of 'subplans' can be executed in parallel. So, both optimality and linearity are not affected. If o_k^1 and o_k^2 are clobberers for each other, then they are not independent and cannot be thus present in the same set of actions. We can observe that o_k^1 can be present in at least a second action set before the action set containing o_k^2 (or vice versa). However, this cannot affect optimality since there is no other option how these 'subplans' can be ordered. Linearity is not affected as well because both the 'subplans' do not have to be 'shredded'. If o_k^1 is a clobberer for o_k^2 but not the other way round, then for optimal solutions it might happen that o_k^1 is present in an action set following the set containing o_k^2. In such a situation we do not have to execute the prefix the 'subplans' share but we move o_k^1 by one (i.e. to the action set following the one containing o_k^2) rather than by two (as in the previous situation). However, in this situation linearity is not violated as well because even if o_k^1 is not present in the action set following the set containing o_k, o_k is an achiever for o_k^2 which can be present in the action set following the set containing o_k. This idea can be analogously extended for more operators o_k is an achiever. □

Having at most one element sets of OR achievers for every operator in a given operator set we can guarantee for every solvable problem an existence of an optimal (parallel) solution plan which is linear operator. The main drawback of this is that such a situation is very uncommon for some more practical domains. To the best of our knowledge, only the nPuzzle domain follows these criteria. Nevertheless, Theorem 2 with its corollaries can be applied in further analysis of AND achievers and also we believe that it might be extended for problem-specific analysis of linearity (e.g. in some problems AND achievers might not be necessary).

Dealing with AND achievers, which is very common in planning domains, brings another dimension into the problem of determining linearity of optimal (parallel) plans. Intuitively, we can see that we have to execute more than one action (e.g a_1, a_2) in order to achieve atom(s) required by another action a. If neither a_1 nor a_2 achieves atom(s) for each other, then the linearity conditions are violated. On the other hand, if we consider optimal parallel plans, a_1 and a_2 if independent can be executed simultaneously. This idea is formalised in the following theorem.

Theorem 3. *Let O be the set of planning operators defined in some planning domain Σ. Assume the following conditions:*

1) For every $o \in O$, $eff^-(o) \subseteq pre(o)$.

2) If $o \nabla_\xi o'$, then $o \equiv o'\xi$.

Then, for every solvable planning problem P defined over Σ, there exists an optimal parallel plan, a solution of P, which is linear.

Proof. The proof mainly derives from the proofs of Theorem 2 and Corollary 2. We will therefore focus on AND achievers. From the assumption (2) we can easily obtain that no action (an instance of some operator) can clobber for another action. Therefore, the clobberer relation (∇) can only cause that an action might not be present twice (or more times) in the same action set.

For any operator o and its AND achievers, for instance o_1 and o_2, we have two possibilities. Firstly, o_1 and o_2 can be in the same action set because they are independent with respect to the corresponding substitution. Secondly, o_1 is an achiever for o_2 (or vice versa) with respect to the corresponding substitution and, therefore, o_1 might be placed into the action set preceding the action set containing o_2 (or vice versa). However, this does not affect optimality nor linearity since o_1 is an achiever for o_2, and o_2 is (together with o_1) an achiever for o. Since o_1 and o_2 can have their achievers and so on, there might be a situation, where to achieve atoms for an instance of o we have to execute sequences $\langle a, a_1^0, \ldots, a_1^k \rangle$ and $\langle a, a_2^0, \ldots, a_2^l \rangle$ commencing by the same action a but having a different lengths (i.e. $k \neq l$). Without loss of generality let $k > l$. A (part of a) solution parallel plan can be constructed as $\langle .., \{a\}, \{a_1^0\}, \ldots, \{a_1^{k-l-1}\}, \{a_1^{k-l}, a_2^0\}, \ldots, \{a_1^k, a_2^l\}, .. \rangle$. Optimality is not violated since the (part of the) solution plan is not greater than the length of the longest sequence. Linearity is not violated as well because a achieves atom(s) for a_1^0 which is in the successive action set and therefore a_2^0 might be placed in some later action set. This idea can be analogously extended for more o's AND achievers.

□

Theorem 3 covers also a fact that delete-relaxed planning problems (negative effects are omitted) have optimal parallel solution plans (if solvable) which are linear. It is formally summarised in the following corollary (note that it directly follows Theorem 3)

Corollary 3. *Let O be the set of planning operators defined in a planning domain Σ. If $\forall o \in O : eff^-(o) = \emptyset$, then for every solvable planning problem P defined over Σ, there exists an optimal parallel plan, a solution of P, which is linear.*

Delete-relaxed Planning Graphs are widely used for determining heuristic estimation from a given state to some goal state. It is good to mention the well known FF heuristic (Hoffmann and Nebel 2001). Although Theorem 3 is quite constrained in its current we believe that it can be extended to allow clobberers which might delete also atoms for other actions than itself. Clobberers which might compromise linearity of optimal parallel plans are these between AND achievers. For example, if operators o_1 and o_2 are AND achievers for an operator o and o_1 is a clobberer for o_2 (w.r.t. a corresponding substitution), then instances of o_1 and o_2 are not independent and therefore cannot be in the same action set. If an instance an instance of o_1 is present in an action set A_i, then an instance of o_2 must be present at most in an action set A_{i-1} while an instance of o must be present at least in an action set A_{i+1}. If o_2 is not an achiever for o_1 (w.r.t a corresponding substitution), then the linearity is compromised. However, to ensure linearity of optimal parallel plans we have to consider that none of AND achievers for all independent operators can be clobberer to some other AND achiever unless it is also and achiever to it (w.r.t a corresponding substitution).

In sequential planning (as stated before) AND achievers threaten linearity of plans. Recalling the situation where actions a_1 and a_2 must be executed in order to achieve atoms for some other action a. However, in a specific case it might also hold that a_1 achieves atom(s) for a_2. Then, a (part of) plan $\langle \dots, a_1, a_2, a, \dots \rangle$ follows the linearity conditions. We can generalise this observation for planning operators. We denote this property as "ordered AND achievers" with respect to a relation $\prec$ which is defined in the following definition. However, having o_1 and o_2, AND achievers for an operator o, we have to consider 'argument matching' between o_1,o_2 and o, and 'parity' of instances of o_1,o_2 and o (e.g. multiple instances of o_1 and a single instance of o_2 are achievers for multiple instances of o). This is explained in the proof of Theorem 4.

Definition 8. *Let o be a planning operator and $\mathcal{A}^*_o$ its AND achievers. We say that $(o_i, \Theta_i) \prec (o_j, \Theta_j)$ if and only if all the following holds:*

- $(o_i, \Theta_i), (o_j, \Theta_j) \in \mathcal{A}^*_o$,
- $o_i \rhd_\xi o_j$ *such that* $\xi \supseteq \{(v_j \to v_i) \mid (v' \to v_j) \in \Theta_j \wedge (v' \to v_i) \in \Theta_i\}$,
- $o_i \nabla_{\sigma_i} o_i$ *such that* $\{v \mid v \neq v' \wedge (v \to v') \in \sigma_i \wedge (v'' \to v) \in \xi\} = \emptyset$,
- $o_j \nabla_{\sigma_j} o_j$ *such that* $\{v \mid v \neq v' \wedge (v \to v') \in \sigma_j \wedge (v \to v'') \in \xi\} = \emptyset$

*We say that $\mathcal{A}^*_o = \{(o_1, \Theta_1), \dots, (o_k, \Theta_k)\}$ is **ordered** if and only if there exists a permutation λ (over the sequence $\langle 1, \dots, k\rangle$) such that $(o_{\lambda(1)}, \Theta_{\lambda(1)}) \prec (o_{\lambda(2)}, \Theta_{\lambda(2)}) \prec \dots \prec (o_{\lambda(k)}, \Theta_{\lambda(k)})$.* ■

Intuitively, if AND achievers can be ordered, then linearity may not be affected. The following theorem extends Theorem 2 considering ordered AND achievers.

Theorem 4. *Let O be the set of planning operators defined in some planning domain Σ. Assume the following conditions:*

1) *For every $o \in O$, and every o's AND achievers $\mathcal{A}^{*i}_o$ it is the case that $\mathcal{A}^{*i}_o$ is ordered.*
2) *For every $o \in O$, eff$^-(o) \subseteq$ pre(o).*

Then, for every solvable planning problem P defined over Σ, there exists a linear optimal plan which is a solution of P.

Proof. This proof extend the proof of Theorem 2 by discussing AND achievers. For some operator o, let $\mathcal{A}^{*i}_o$ be o's AND achievers. From the assumption we know that $\mathcal{A}^{*i}_o$ is ordered and, therefore, there exists a way how pairs (operator,substitution) can be ordered with respect to the relation $\prec$ (see Definition 8). Without loss of generality, assume that $(o_x, \Theta_x), (o_y, \Theta_y) \in \mathcal{A}^{*i}_o$ (i.e., $o_x \rhd_{\Theta_x} o$ and $o_y \rhd_{\Theta_y} o$) and $(o_x, \Theta_x) \prec (o_y, \Theta_y)$. According to $\prec$ it holds that $o_x \rhd_\xi o_y$, $\xi \supseteq \{v_y \to v_x \mid v' \to v_y \in \Theta_y \wedge v' \to v_x \in \Theta_x\}$, $o_x \nabla_{\sigma_x} o_x$ ($\{v \mid v \neq v' \wedge v \to v' \in \sigma_x \wedge v'' \to v \in \xi\} = \emptyset$) and $o_y \nabla_{\sigma_y} o_y$ ($\{v \mid v \neq v' \wedge v \to v' \in \sigma_y \wedge v \to v'' \in \xi\} = \emptyset$) Since o_x is an achiever for o_y and both are achievers for o, it is straightforward how we can order their corresponding instances in solution plans. However, linearity depends on the substitutions Θ_x, Θ_y and ξ. $\{v_y \to v_x \mid v' \to v_y \in \Theta_y \wedge v' \to v_x \in \Theta_x\}$ determines an 'argument matching' substitution for operators o_x and o_y according to atom(s) they achieve for o. If the 'argument matching' substitution is supported by ξ, i.e., ξ is a superset of the 'argument matching', we can see that for all instances of o_x and o_y achieving for a certain instance of o, is the case that an instance of o_x is an achiever for the corresponding instance of o_y. Hence, the linearity conditions are not violated. However, optimality might be affected if more instances of o_x and a certain instance of o_y (or vice versa) are achievers for more instances of o. This situation is avoided by the fact that o_x and o_y are 'self'-clobberers in such a way that only one instance of o_x which achieves for a certain instance of o_y is allowed and similarly only one instance of o_y is allowed after a certain instance of o_x achieves atom(s) for it. □

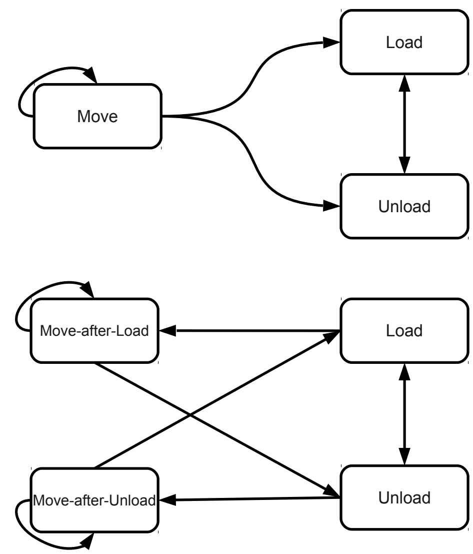

Figure 2: The original Logistics domain (upper graph) and the modified Logistic domain (lower graph).

Let us have a (simpler) Logistics domain where we have only one truck which can carry at most one package. The domain consists of three operators, move(?l1 ?l2), load(?p ?l) and unload(?p ?l). We can see that for unload we have two AND achievers, move and load. Although move is an achiever for load, by analysing particular substitution we can find out that it does not follow the ordering condition (Def. 8) and, hence, we cannot use Theorem 4 to determine linearity (intuitively, linearity does not hold). However, 'splitting' move into operators move-after-load and move-after-unload might result in fulfilling the conditions of Theorem 4. This also requires to 'split' a predicate at(?l), which refers to a position of the truck, into at-empty(?l) and at-full(?l). Note that load and unload must be updated according to this as well. The described Logistics domains are shown in Figure 2.

Discussion

Linearity is useful along several dimensions: it can be operationalised as a static test for problems and used as a search heuristic, it can be used as a target property for the (possibly automated) reconfiguration of domain models, or it can be taken as a theoretical contribution to the study of planning problems and planning domain encoding.

Application to Planning Engines

Finding that problems in a domain model have solutions which are linear results in a pruning heuristic for a planner. Planning engines based on forward search (e.g. FF (Hoffmann and Nebel 2001)) can be augmented as follows. In the node expansion phase we must allow only such neighbour nodes which comply with the linearity conditions. Let a be an action which led to the current state. Then, if $\mathit{eff}^+(a) \cap g = \emptyset$ (a does not achieve any goal atom), then we can allow only such an action a' such that $\mathit{eff}^+(a) \cap \mathit{pre}(a') \neq \emptyset$. Otherwise, that is, $\mathit{eff}^+(a) \cap g \neq \emptyset$ we might allow also actions which do not require atoms from a. However, after that we have to ensure that at least one atom from $\mathit{eff}^+(a) \cap g$ is not consumed and/or re-achieved by any of the following actions.

Planning engines based on backward search, which are not that common as the forward search ones, can be augmented to follow linearity in the node expansion phase as well. Let a be an action which led to the current state s and g_s be a set of 'open goals' in s ('open goals', similarly to plan-space planning terminology, are atoms which have not yet been achieved in s). In the node expansion phase we can allow only actions a' such that $\mathit{eff}^+(a') \cap (\mathit{pre}(a) \cup g_s) \neq \emptyset$. Note that in the following state, the set of 'open goals' is updated as $g_s \setminus \mathit{eff}^+(a')$. From this perspective, we intuitively believe that applying linearity in backward search planning is easier than in forward search planning.

For SAT and CSP based planning, linearity can be encoded directly in SAT formulae, or constraints. Linearity can be also exploited in some planning focused SAT solvers (Rintanen 2012). In the case of parallel planning, enabling linearity can be done analogously.

Application to Modelling and Reformulation

How planning domains are modelled affects whether or not solutions of problems might be linear. For example, the easiest way how to provide a domain following the linearity conditions is to introduce a supplementary atom (proposition) and place it in the preconditions and positive effects of all the operators in the domain. However, such a blind reformulation might not reduce the potential search in a planner.

Recalling the BlocksWorld example we can see that we modified the original operator schema by introducing a supplementary atom handfull. This has not affected the domain in terms of losing solvability or losing a possibility to find an optimal solution. However, it enables to (directly) use Theorem 1 to determine linearity. Recalling the simple Logistic example we can see that 'splitting' an operator move into operators move-after-load and move-after-unload can enable to use Theorem 4 to determine linearity.

Therefore, it seems to be very reasonable to study possibilities of modelling or reformulating planning domains in order to support linearity. As the examples above indicated one possibility how to reformulate domains is to create an achiever relation between some operators. Another possibility is to use macro-operators (Botea et al. 2005; Chrpa 2010), for instance, load-move.

Finally, we believe that reformulating domains in order to support linearity can be done automatically. A statical analysis of domain structures, eventually problem structures (if our general theoretical framework is extended), would be sufficient to (i) identify if a given domain (and problem) structure supports linearity and, (ii) propose reformulations. In case there exists different possible reformulations for a given domain (problem), it will be important to select the "best" one, with respect to some properties that are related to the performance of domain-independent planning systems.

Theoretical Aspects of Linearity

Linearity provides no *guarantee* of improving the time complexity of planning, as illustrated in the example given above, where the same proposition was added to the preconditions and positive effects of all operator schema. Although there is no relation between linearity and complexity of planning, analysing operator schema reveals some interesting aspects. There is a connection between OR and AND achievers and OR and AND Landmarks (Hoffmann, Porteous, and Sebastia 2004). Landmarks are atoms that, given any solution plan π, must become true in some state during the execution of π. If each operator has at most one OR set of achievers and the operator schema follows Theorem 2, 3 or 4, then we believe that finding optimal solution for some planning problems might be tractable. To justify this hypothesis we can see that if we apply backward search, then the search can directly go towards the initial state. However, determining which instances of which operators can achieve goal atoms, or the order in which goal atoms have to be achieved might hinder the tractability. Also, one possible outcome from analysis of operator schema is determining Shortcut Rules (Karpas and Domshlak 2012) which stand for mappings from 'longer' operator sequences to 'shorter' ones such that the sequences achieve the same predicates. Shortcut Rules might be useful in optimal planning as well as in post-planning optimisation.

Conclusions

In this paper we introduced a property of sequential and parallel solution plans to a planning problem, called linearity. This refers to situations where the set of optimal plans includes a linear plan, that is one in which every action achieves goal atom(s) or atom(s) for the following action. In the parallel version, it is where actions from a set of independent actions at one step in the parallel plan, all achieve atoms for the next set of independent actions at the next step.

We analysed planning operator schema by using relations of being an achiever, a clobberer or being independent of one another. According to specific properties of these relations we were able to determine on a problem-independent basis in which cases we are able to produce optimal (parallel) solution plans which are linear. Though the main focus of the paper is on a theoretical analysis of the property, we have shown how linearity can be applied as a heuristic within typical state space and goal directed planning engines, and how planning domains can be modified in order to enable linearity. In the future, we plan to focus on i) augmenting existing state-of-the-art planning engines in order to exploit linearity, ii) providing methods for reformulating planning domains in order to enable linearity, iii) outlining guidelines for designing new planning domains and iv) extending our theoretical study, for instance, by further investigating similarities and differences between our work and research into Landmarks and Learning Shortcut Rules.

Acknowledgements The research was funded by the UK EPSRC Autonomous and Intelligent Systems Programme (grant no. EP/J011991/1).

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS+ planning. *Computational Intelligence* 11:625–656.

Bäckström, C.; Chen, Y.; Jonsson, P.; Ordyniak, S.; and Szeider, S. 2012. The complexity of planning revisited - a parameterized analysis. In *Proceedings of AAAI*, 1735–1741.

Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90(1-2):281–300.

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.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *Artificial Intelligence* 69:165–204.

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

Chen, Y., and Yao, G. 2009. Completeness and optimality preserving reduction for planning. In *Proceedings of IJCAI*, 1659–1664.

Chrpa, L.; McCluskey, T. L.; and Osborne, H. 2012. Optimizing plans through analysis of action dependencies and independencies. In *Proceedings of ICAPS*, 338–342.

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

Coles, A. J., and Coles, A. I. 2010. Completeness-preserving pruning for optimal planning. In *Proceedings of ECAI*, 965–966.

Do, M. B., and Kambhampati, S. 2001. Planning as constraint satisfaction: Solving the planning graph by compiling it into CSP. *Artificial Intelligence* 132:151–182.

Fox, M., and Long, D. 1998. The automatic inference of state invariants in TIM. *Journal of Artificial Intelligence Research (JAIR)* 9:367–421.

Fox, M., and Long, D. 2001. STAN4: a hybrid planning strategy based on subproblem abstraction. *AI Magazine* 22(3):81–84.

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

Gimenez, O., and Jonsson, A. 2007. On the hardeness of planning problems with simple causal graphs. In *Proceedings of ICAPS*, 152–159.

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

Helmert, M. 2006. New complexity results for classical planning benchmarks. In *Proceedings of ICAPS*, 52–62.

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.

Karpas, E., and Domshlak, C. 2012. Optimal search with inadmissible heuristics. In *Proceedings of ICAPS*, 92–100.

Katz, M., and Domshlak, C. 2007. Structural patterns of tracable sequentialy-optimal planning. In *Proceedings of ICAPS*, 200–207.

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

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

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

Rintanen, J. 2012. Engineering efficient planners with SAT. In *Proceedings of ECAI*, 684–689.

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

Surynek, P. 2008. Tractable class of a problem of goal satisfaction in mutual exclusion network. In *Proceedings of FLAIRS Conference*, 561–566.