Mechanising Theoretical Upper Bounds in Planning

Mohammad Abdulaziz and **Charles Gretton** and **Michael Norrish** [*]
Canberra Research Lab., NICTA
7 London Circuit, Canberra ACT 2601, Australia
{Mohammad.Abdulaziz, Charles.Gretton, Michael.Norrish}@nicta.com.au

Abstract

We examine the problem of computing upper bounds on the lengths of plans. Tractable approaches to calculating such bounds are based on a decomposition of state-variable dependency graphs (causal graphs). Our contribution follows an existing formalisation of concepts in that setting, reporting our efforts to mechanise bounding inequalities in HOL. Our first contribution is to identify and repair an important error in the original formalisation of bounds. More importantly, we also develop novel theoretical bounding results and compare them analytically with existing bounds.

Introduction

This paper develops novel insights and approaches for reasoning about upper bounds on the lengths of plans. Formally, an *upper bound* of N means that, if a plan exists, then an optimal plan comprises no more than N steps. A variety of applications for such upper bounds have been proposed. If an explicit state-based search encounters a state with *upper bound* N and *lower bound* M—if a plan exists, it must be at least of length M—then if $N < M$ the state can safely be pruned. Also, given a tight upper bound N, the plan existence problem can be reduced to a fixed-horizon reachability problem—*i.e.*, is there a plan of length less-than-or-equal-to N? In that case the fixed-horizon problem can be posed as a Boolean SAT(isfiability) problem (Kautz and Selman 1996). More generally, planning problems can be solved using SAT solvers given a query strategy that focuses search effort at important horizon lengths (Rintanen 2004; Streeter and Smith 2007). Tight horizon bounds provide focus in that setting. Lastly, in situations where no plan exists, bounds have been used to identify a small subset of goal facts that cannot be achieved together. Here, plan existence for goal sets which admit short bounds are tested earliest, so that non-existence can be established quickly using relatively little search effort.

Our contributions follow (Rintanen and Gretton 2013), which describes a general procedure for computing upper bounds based on state-variable dependency information. That approach yields useful bounds in problems that exhibit a branching one-way dependency structure. A highlighted example of that type of dependency occurs in the *logistics* benchmark. To change the location of a package, vehicles must be used. The locations of vehicles can be modified irrespective of the package locations, and indeed independently of each other. In other words, each package has a one-way dependency with vehicles and otherwise all objects can be manipulated independently.

This work reports on our efforts so far to obtain mechanized proofs of the correctness of versions of the headline theorems from (Rintanen and Gretton 2013). Our work has exposed a subtle yet important error in the original formalisation of bounds. We correct the original formalisation of upper bounds and summarise the new proof of the bounds from that work. [1] The new proof employs a constructive technique, relying on a function which builds a plan of an appropriate length given an overly long input. Our new proof of correctness is mechanised in the HOL interactive theorem proving system (Slind and Norrish 2008). We provide links to that mechanisation work which is hosted on `github`. Finally, we also develop novel inequalities which yield tighter bounds than existing approaches.

Definitions and Notations

We formalise the planning problem and give definitions of concepts related to computing bounds on solution lengths. Our definitions are functionally equivalent to standard expositions of deterministic propositional planning. We include two minor departures, used to decrease the verbosity of our proofs. In particular, our formalisation supposes there is a single goal-state, rather than a set of goal states. Also, we allow any action to be executed at any state, supposing its effects are only realised if its preconditions are satisfied at the state from which it is executed.

Definition 1 (States and Actions)**.** *A planning problem is defined in terms of* states *and* actions*:*

1. *We model states as finite maps from variables—i.e., state characterizing propositions—to Booleans.*

[*]NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

[1]We note that the algorithm and experimental results from (Rintanen and Gretton 2013) are presumed correct: that part of the work uses cardinality derived bounds rather than the bounding concept presented in the original formalisation.

2. *An action π is a pair of finite maps. Each of those maps is from a subset of the problem variables to the Booleans. The domain of each of these maps does not have to be the same. The first component of the pair ($p(\pi)$) is the* precondition*: each variable in the domain of the map must have the specified value if the action is to affect any state change.*[2] *The second component of the pair ($e(\pi)$) is the* effect*: each variable in the domain of the map takes on the specified value in the resulting state. We say that a variable is an* action precondition *if it is in the domain of the precondition map, and similarly that it is an* action effect *if it is in the domain of the effect map.*
3. *An action sequence $\dot{\pi}$ is a list of actions. We use the notation $\pi :: \dot{\pi}$ (a "cons") to denote the sequence which has π as its first element, followed by the actions in $\dot{\pi}$, and $[\,]$ to denote the empty sequence.*

We will write concrete examples of states and actions as sets of variables that are either bare (mapping to true), or overlined (mapping to false). For example, $\{x, \overline{y}, z\}$ is the state where state variables x and z are true, and y is false.

We now use the above concepts to define a planning *problem*.

Definition 2 (Planning Problems)**.** *A problem Π is a 3-tuple $\Pi = \langle I, A, G\rangle$, with I the initial state of the problem, G the goal state and A a set of permitted actions. When writing about the components of a problem Π, unless explicitly written otherwise we will write I, A and G for $\Pi.I$ etc., leaving the Π implicit. We will write D for the domain of the initial state; this is the domain of the problem.*

Problem Π is valid *if the goal has the same domain as the initial state, and all actions refer exclusively to variables that occur in that set. We only consider valid problems.*

Naturally, a state s is valid with respect to a planning problem Π if its domain is the same as that of the initial state I.

Definition 3 (Action Execution)**.** *When an action π is executed at state s, it causes a transition to a successor state. If the precondition is not satisfied at s, the successor is simply s once more. Otherwise, the action effects hold at the successor. We denote this operation $\mathsf{exec}(s, \pi)$. We lift that definition to sequences of executions taking an action sequence $\dot{\pi}$ as the second argument. So $\mathsf{exec}(s, \dot{\pi})$ denotes the state resulting from successively applying each of $\dot{\pi}$'s actions, starting with s.*

Key concepts in formalising bounds in planning are those of *projection* and *dependency graph*.

Definition 4 (Projection)**.** *Projecting an object (a state s, an action π, a sequence of actions $\dot{\pi}$ or a problem Π) on a set of variables vs refers to restricting the domain of the object or its constituents to vs. We denote these operations as $s\downharpoonright_{vs}$, $\pi\downharpoonright_{vs}$, $\dot{\pi}\downharpoonright_{vs}$ and $\Pi\downharpoonright_{vs}$ for a state, action, action sequence and problem respectively. Note that if an action sequence $\dot{\pi}$ is projected in this way, it may come to include actions with empty preconditions and/or effects, however, actions with empty effects are removed.*

[2]We use the word *domain* in the mathematical sense—*i.e.*, the set from which the function arguments are drawn. To avoid confusion, we shall not use this term to refer to a PDDL model.

Definition 5 (Problem Dependency Graphs)**.** *The dependency graph of a problem Π is a directed graph, written G, describing variable dependencies. This graph was conceived under different guises in (Williams and Nayak 1997) and (Bylander 1994), and is also commonly referred to as a* causal graph*. That graph features one vertex for each variable in D. An edge from v_1 to v_2 records that v_2 is dependent on v_1. A variable v_2 is dependent on v_1 in a planning problem Π iff one of the following statements holds:*

1. *v_1 is the same as v_2.*
2. *There is an action π in A such that v_1 is a precondition of π and v_2 is an effect of π.*
3. *There is an action π in A such that both v_1 and v_2 are effects of π.*

We write $v_1 \to v_2$ if there is a directed arc from v_1 to v_2 in the dependency graph for Π. Also, when we illustrate a dependency graph we do not draw arcs from a variable to itself although it is dependent on itself.

We also lift the concept of dependency graphs and refer to *lifted* dependency graphs (written G_{vs}) in which each vertex represents a distinct set of problem variables. The vertices (set of variables) in lifted graphs will partition the domain D of the original problem.

Definition 6 (Dependencies of Sets of Variables)**.** *An edge from a set of variables vs_1 to set vs_2 records that vs_2 is dependent on vs_1. A set of variables vs_2 is dependent on vs_1 in a planning problem Π (written $vs_1 \to vs_2$) iff all of the following conditions hold:*

1. *vs_1 is disjoint from vs_2.*
2. *There exist variables $v_1 \in vs_1$ and $v_2 \in vs_2$ such that $v_1 \to v_2$.*

In revising previous work, we refer to the strongly connected components (SCCs) of a dependency graph.

Definition 7 (Strongly Connected Component)**.** *An SCC of G is a maximal subgraph in which there is a directed path from each vertex to every other vertex. We write G_S for the lifted graph which has one vertex for each SCC in G, and an edge from component (a set of variables) vs_1 to component vs_2 iff $vs_1 \to vs_2$. Note that G_S will be a DAG.*

Definition 8 (Leaves, Ancestors and Children)**.** *For any DAG G, the set of leaves $\mathcal{L}(G)$ contains those vertices of G from which there are no outgoing edges. We also write $\mathcal{A}_G(n)$ to denote the set of n's ancestors in G. Formally, $\mathcal{A}_G(n) = \{n_0 \mid n_0 \in G \wedge n_0 \to^+ n\}$, where $\to^+$ is the transitive closure of $\to$.We also write $\mathcal{C}_G(n)$ to denote the set $\{n_0 \mid n_0 \in G \wedge n \to n_0\}$ which are the children of n in G.*

Finally, an important relation on lists (of actions) on which our work relies is the *scattered sublist*[3] relation:

Definition 9 (Scattered Sublists)**.** *List l_1 is a scattered sublist of l_2 (written $l_1 \preceq\!\cdot\; l_2$) if all the members of l_1 occur in the same order in l_2.*

[3]Our "scattered sublist" is sometimes referred to as a "subsequence" in the computer science literature. In HOL we require a distinct concept.

$$
\begin{array}{rcl}
[\,]\underset{vs}{\between}\dot{\pi} & = & \dot{\pi}\downharpoonleft_{D-vs} \\
\dot{\pi}_c\underset{vs}{\between}[\,] & = & [\,] \\
\pi_c :: \dot{\pi}_c\underset{vs}{\between}\pi :: \dot{\pi} & = & \begin{cases} \pi :: (\dot{\pi}_c\underset{vs}{\between}\dot{\pi}) & \text{if } \pi\downharpoonleft_{vs} = \pi_c \\ \pi_c :: \dot{\pi}_c\underset{vs}{\between}\dot{\pi} & \text{o/wise} \end{cases}
\end{array}
$$

Figure 1: The definition of the stitching function ($\between$).

$$\begin{tikzcd}
 & w \arrow[dl] \arrow[d] \arrow[dr] & \\
x \arrow[r, leftrightarrow] & y \arrow[r, leftrightarrow] & z
\end{tikzcd}$$

Figure 2: The dependency graph of the problem in Example 2.

Derivation of Upper Bounds

The main contribution of our work is the constructive proof technique we use to derive plan-length bounds. Our approach is the outcome of an exercise in mechanising the proofs of the results from (Rintanen and Gretton 2013) using the HOL interactive theorem proving system (Slind and Norrish 2008)[4]. Working from first principles, that exercise uncovered a subtle yet important error in the original formalisation of bounds.In detail, a plan-length bound was originally formulated as the maximum of the minimum length executions between pairs of states. An important error becomes apparent in the usage of that bound, which follows the algorithm of iterative plan refinement described originally in (Knoblock 1994). Summarising the error, earlier work incorrectly assumed that an abstract plan satisfying that bound could be refined into a complete plan for the problem at hand. However, valid problems exist where no minimum length executions in the abstract model can be refined to create a valid plan. The key assumption is invalid and thus a correction is needed. In what follows we describe the error we discovered in (Rintanen and Gretton 2013), correct that error, and then describe our proofs of important bounding theorems.

Error

The headline result in (Rintanen and Gretton 2013) is described using an upper bound function that was written using the ℓ symbol. Because we are treating both the erroneous and repaired versions of the function in our work, we have chosen to write $\ell_\perp$ for the original erroneous version. We use the subscript $\perp$ symbol to emphasise that it leads to an invalid result. Intuitively, $\ell_\perp$ denotes a function which takes a planning problem Π and returns the length of the longest optimal execution between any two valid states in Π. We now formally review the definition of $\ell_\perp$, identifying and explaining two errors. Following that, we shall repair that definition in support of the upper bounds in (Rintanen and Gretton 2013) . We write $\Pi(s)$ for the set of finite lists of actions in Π that reach s from I, $\mathcal{S}$ for the set of states in Π, and $|\dot{\pi}|$ for the length of execution $\dot{\pi}$.

Definition 10.

$$\ell_\perp(\Pi) = \max_{s\in\mathcal{S}} \min_{\dot{\pi}\in\Pi(s)} |\dot{\pi}|$$

A first negative consequence of the above definition renders $\ell_\perp$ unsuitable for making statements about upper bounds. Specifically, $\ell_\perp$ is not well-defined in situations where there are no valid executions between I and a state s—*i.e.*, the function $\min$ has no well-defined output in that situation. This issue was not dealt with adequately in (Rintanen and Gretton 2013) . We illustrate this error with an example.

Example 1. *Consider the planning problem* Π *such that*

$$\Pi \;=< I = \{\overline{x},\overline{y},\overline{z}\}, A = \{(\{x,\overline{y}\},\{z\})\}, G = \{x,\overline{y},z\} >$$

Now let $s = \{\overline{x},\overline{y},\overline{z}\}$ *and* $s' = \{x,\overline{y},z\}$. *These are two valid states in* Π *but a transition from* s *to* s' *is impossible. So* $\ell_\perp$ *is thus ill-defined for* Π.

We can mitigate this ill-definedness by treating reachability explicitly, as follows. We shall see in a moment however that the proposed correction is still insufficient. Let $\dot{A}$ be the set of finite lists of actions in Π and $\Pi(\dot{\pi}, s) = \{\dot{\pi}' | \mathsf{exec}(s,\dot{\pi}) = \mathsf{exec}(s,\dot{\pi}') \wedge \dot{\pi}' \in \dot{A}\}$, *i.e.*, the set of executions from s equivalent to $\dot{\pi}$.

Definition 11.

$$\ell'_\perp(\Pi) = \max_{s\in\mathcal{S},\dot{\pi}\in\dot{A}} \min_{\dot{\pi}'\in\Pi(\dot{\pi},s)} |\dot{\pi}'|$$

This proposed modification to $\ell'_\perp$ is insufficient to fix a deeper error. A headline results from (Rintanen and Gretton 2013) states:

Not-A-Theorem 1. *If the domain of* Π *is comprised of two disjoint sets of variables* vs_1 *and* vs_2 *satisfying* $vs_2 \not\rightarrow vs_1$*, we have:*

$$\ell'_\perp(\Pi) < (\ell'_\perp(\Pi\downharpoonleft_{vs_1}) + 1)(\ell'_\perp(\Pi\downharpoonleft_{vs_2}) + 1)$$

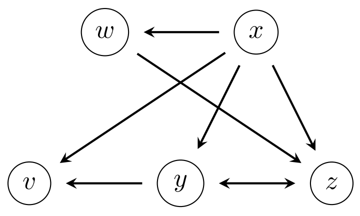
This inequality is invalid with respect to Definition 11 (and Definition 10), as follows.

Example 2. *Consider the problem*

$$\Pi \;=\; \left\langle \begin{array}{l} I = \{\overline{w},\overline{x},\overline{y},\overline{z}\} \\ A = \left\{ \begin{array}{l} a = (\emptyset,\{x\}), b = (\{x\},\{\overline{x},y\}), \\ c = (\{x,y\},\{\overline{x},\overline{y},z\}), d = (\{w\},\{x,y,z\}) \end{array} \right\} \\ G = \{\overline{w},x,y,z\} \end{array} \right\rangle$$

whose dependency graph is shown in Figure 2. Note that here we include the starting and goal conditions, however $\ell'_\perp$ *is independent of those. The domain is comprised of two sets of variables* $S = \{x,y,z\}$*, which is an SCC, and*

[4]The mechanised formalisation and proofs described in this paper can be found online at `https://github.com/mabdula/planning/`.



$$\begin{tikzcd}
 & w & & x \arrow[ll] \arrow[dlll] \arrow[dl] \arrow[dr] & \\
v & & y \arrow[ll] \arrow[rr, leftrightarrow] & & z \arrow[from=1-2]
\end{tikzcd}$$

Figure 3: Dependency graph of the problem in Example 3.

the set $\mathcal{A}_{G_S}(S) = \{\{w\}\}$. The evaluation of $\ell'_{\perp}(\Pi)$ gives 7, as this is the maximal length optimal execution. Specifically, that maximal optimal execution is $[a; b; a; c; a; b; a]$ from $\{\overline{w}, \overline{x}, \overline{y}, \overline{z}\}$ to $\{\overline{w}, x, y, z\}$. Treating the abstract problems $\Pi\downharpoonright_S$ and $\Pi\downharpoonright_{\bigcup \mathcal{A}_{G_S}(S)}$, we get the bounds 1 and 6, respectively. This violates Not-A-Theorem 1.

Mechanisation of Existing Bounds

In this section we provide a corrected definition of ℓ. Adopting that new definition we describe our mechanised proof of the inequality that featured in Not-A-Theorem 1. We also develop novel bounds and compare them to the inequalities suggested in (Rintanen and Gretton 2013) , showing that in some cases our novel bounds dominate.

Our definition of ℓ mitigates the problem exhibited in the definition by (Rintanen and Gretton 2013) by appealing to $\Pi^{\preceq\cdot}(\dot{\pi}, s) = \{\dot{\pi}' | \mathsf{exec}(s, \dot{\pi}) = \mathsf{exec}(s, \dot{\pi}') \wedge \dot{\pi}' \preceq\cdot\, \dot{\pi}\}$ – *i.e.*, the set of executions from s that are equivalent to $\dot{\pi}$ and also scattered sublists of $\dot{\pi}$.

Definition 12.

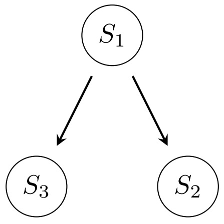
$$\ell(\Pi) = \max_{s \in \mathcal{S}, \dot{\pi} \in \dot{A}} \ \min_{\dot{\pi}' \in \Pi^{\preceq\cdot}(\dot{\pi}, s)} |\dot{\pi}'|$$

It should be clear that $\ell(\Pi)$ is a valid upper bound for Π. Using ℓ we can first prove a corrected version of Not-A-Theorem 1.

Theorem 1. *If the domain of Π is comprised of two disjoint sets of variables vs_1 and vs_2 satisfying $vs_2 \not\rightarrow vs_1$, we have:*

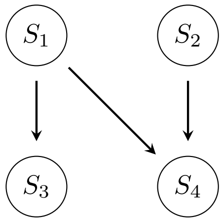
$$\ell(\Pi) < (\ell(\Pi\downharpoonright_{vs_1}) + 1)(\ell(\Pi\downharpoonright_{vs_2}) + 1)$$

Proof. To prove Theorem 1 we use a construction which, given any plan $\dot{\pi}$ for Π violating the stated bound, produces a shorter witness plan $\dot{\pi}'$ satisfying that bound. The premise $vs_2 \not\rightarrow vs_1$ implies that actions with variables from vs_2 in their effects—hereupon we shall call these vs_2-actions—never include vs_1 variables in their effects. Also, because vs_1 and vs_2 capture all problem variables, the effects of vs_1-actions after projection to the set vs_1 are unchanged. Our construction first considers the abstract action sequence $\dot{\pi}\downharpoonright_{vs_2}$. Definition 12 of ℓ provides a scattered sublist $\dot{\pi}'_{vs_2} \preceq\cdot\, \dot{\pi}\downharpoonright_{vs_2}$ satisfying $|\dot{\pi}'_{vs_2}| \leq \ell(\Pi\downharpoonright_{vs_2})$. Moreover, the definition of ℓ can guarantee that $\dot{\pi}'_{vs_2}$ is equivalent, in terms of the execution outcome, to $\dot{\pi}\downharpoonright_{vs_2}$. The stitching function described in Figure 1 is then used to remove the vs_2-actions in $\dot{\pi}$ whose projections on vs_2 are not in $\dot{\pi}'_{vs_2}$. Thus our construction arrives at a plan $\dot{\pi}'' = \dot{\pi}'_{vs_2} \underset{vs_2}{\wr} \dot{\pi}$ with at most $\ell(\Pi\downharpoonright_{vs_2})$ vs_2-actions. We are left to address the continuous lists of vs_1-actions in $\dot{\pi}''$, to ensure that in the constructed plan any such list satisfies the bound $\ell(\Pi\downharpoonright_{vs_1})$. The method by which we obtain $\dot{\pi}''$ guarantees that there are at most $\ell(\Pi\downharpoonright_{vs_2}) + 1$ such lists to address. The definition of ℓ provides that for any abstract list of actions $\dot{\pi}\downharpoonright_{vs_1}$ in $\Pi\downharpoonright_{vs_1}$, there is a list that achieves the same outcome of length at most $\ell(\Pi\downharpoonright_{vs_1})$. Our construction is completed by replacing each continuous sequence of vs_1-actions in $\dot{\pi}''$ with witnesses of appropriate length ($\ell(\Pi\downharpoonright_{vs_1})$). ☐



$$\begin{tikzcd}
 & S_1 \arrow[dl] \arrow[dr] & \\
S_3 & & S_2
\end{tikzcd}$$

Figure 4: An SCC graph with 3 SCCs.



$$\begin{tikzcd}
S_1 \arrow[d] \arrow[dr] & S_2 \arrow[d] \\
S_3 & S_4
\end{tikzcd}$$

Figure 5: An SCC graph with 4 SCCs.

The above construction can be illustrated using the following concrete example.

Example 3. *Consider the valid problem*

$$\Pi = \left\langle \begin{array}{l} I = \{v, \overline{w}, \overline{x}, \overline{y}, \overline{z}\} \\ A = \left\{ \begin{array}{l} a = (\emptyset, \{x\}), b = (\{x\}, \{y\}), \\ c = (\{x\}, \{\overline{v}\}), d = (\{x\}, \{w\}), \\ e = (\{y\}, \{v\}), f = (\{w, y\}, \{z\}), \\ g = (\{\overline{x}\}, \{y, z\}) \end{array} \right\} \\ G = \{v, w, x, y, z\} \end{array} \right\rangle$$

whose dependency graph is shown in Figure 3. The domain of Π has a subset $vs_2 = \{v, y, z\}$ where vs_2 is dependent on the set $vs_1 = \{w, x\}$, and vs_1 is not dependent on vs_2.

In Π, the actions b, c, e, f, g are vs_2-actions, and a, d are vs_1-actions. A plan $\dot{\pi}$ for Π is $[a; a; b; c; d; d; e; f]$. When the plan $\dot{\pi}$ is projected on vs_2 it becomes $[b\downharpoonright_{vs_2}; c\downharpoonright_{vs_2}; e\downharpoonright_{vs_2}; f\downharpoonright_{vs_2}]$, which is a plan for $\Pi\downharpoonright_{vs_2}$. A shorter plan, $\dot{\pi}_c$, for $\Pi\downharpoonright_{vs_2}$ is $[b\downharpoonright_{vs_2}; f\downharpoonright_{vs_2}]$. Since $\dot{\pi}_c$ is a scattered sublist of $as\downharpoonright_{vs_2}$, we can use the stitching function to obtain a shorter plan for Π. In this case, $\dot{\pi}_c \underset{vs_2}{\wr} \dot{\pi}$ is $[a; a; b; d; d; f]$. The second step is to contract the pure vs_1 segments which are $[a; a]$ and $[d; d]$, which are contracted to $[a]$ and $[d]$ respectively. The final constructed witness for our bound is the plan $[a; b; d; f]$.

So far we have seen how to reason about problem bounds by treating abstract subproblems separately. We now review how subexponential bounds for planning problems are achieved by exploiting branching one-way state-variable dependencies. An example of that type of dependency structure is exhibited in Figure 4, where S_i are sets of variables each of which forms an SCC in the dependency graph, and we have both $S_1 \rightarrow S_2$ and $S_1 \rightarrow S_3$. Recall, the latter means that there is at least one edge from a variable in S_1 to one in S_2, and similarly between S_1 and S_3. Importantly, Figure 4 gives $S_2 \not\rightarrow S_1$ and $S_3 \not\rightarrow S_1$, and there is no dependency between any variables in S_2 and S_3. For bounding optimal plan lengths, the following theorem was suggested in (Rintanen and Gretton 2013) to exploit such structures.

Theorem 2. *Following Definition 7, G_S is the DAG of SCCs from the dependency graph for a problem Π. The upper bound $\ell(\Pi)$ satisfies the following inequality:*

$$\ell(\Pi) \leq \Sigma_{S \in \mathcal{L}(G_S)} \ell(\Pi\downharpoonright_{S \cup (\bigcup \mathcal{A}_{G_S}(S))}) \tag{1}$$

In our work we have established sometimes superior bounds by deviating from the above inequality. The theorem that we mechanised can provide tighter bounds for some specific dependency structures; and otherwise it does not dominate and is not dominated by the bound provided by the inequality in Theorem 2. Our theorem exploits planning problems which are partitioned by two sets of state variables that are not connected in the dependency graph.

Theorem 3. *For a problem Π whose domain is partitioned by vs_1 and vs_2 such that $vs_1 \not\rightarrow vs_2$ and $vs_2 \not\rightarrow vs_1$*

$$\ell(\Pi) \leq \ell(\Pi\downharpoonright_{vs_1}) + \ell(\Pi\downharpoonright_{vs_2}) \tag{2}$$

Proof. The premises $vs_1 \not\rightarrow vs_2$ and $vs_2 \not\rightarrow vs_1$ implies that vs_2-actions have no variables from vs_1 in their effects or preconditions and vs_1-actions have no vs_2 variables in their effects or preconditions. This implies that removing vs_1-actions from a plan does not affect the executability of vs_2-actions in that plan. This statement applies in the other direction also, in the case of vs_2-action removal. Our construction first takes the action sequence $\dot{\pi}\downharpoonright_{vs_2}$. Definition 12 of ℓ provides an action sequence $\dot{\pi}'_{vs_2}$ scattered sublist $\dot{\pi}'_{vs_2} \preceq \dot{\pi}\downharpoonright_{vs_2}$ satisfying $|\dot{\pi}'_{vs_2}| \leq \ell(\Pi\downharpoonright_{vs_2})$. Moreover, the definition of ℓ guarantees that $\dot{\pi}'_{vs_2}$ is equivalent, in terms of the execution outcome, to $\dot{\pi}\downharpoonright_{vs_2}$. The stitching function described in Figure 1 is then used to remove the vs_2-actions in $\dot{\pi}$ whose projection on vs_2 is not in $\dot{\pi}'_{vs_2}$. Thus our construction arrives at a plan $\dot{\pi}'' = \dot{\pi}'_{vs_2} \underset{vs_2}{\wr} \dot{\pi}$ whose execution outcome is the same as $\dot{\pi}$ but in which the number of vs_2-action is at most $\ell(\Pi\downharpoonright_{vs_2})$. The next step is to consider $\dot{\pi}''\downharpoonright_{vs_1}$. Then we obtain $\dot{\pi}'_{vs_1}$ whose execution outcome is equivalent to $\dot{\pi}''\downharpoonright_{vs_1}$, which has at most $\ell(\Pi\downharpoonright_{vs_1})$ actions. Then stitching again we obtain the plan $\dot{\pi}'_{vs_1} \underset{vs_1}{\wr} \dot{\pi}''$ which is a plan that has the same outcome as $\dot{\pi}''$ and accordingly $\dot{\pi}$ but with at most $\ell(\Pi\downharpoonright_{vs_1}) + \ell(\Pi\downharpoonright_{vs_2})$ actions. □

Comparison

In this section we compare different ways to decompose dependency graphs based on the results we have so far, and study the upper bounds thus obtained.

Case 1 Following Theorem 2, a bound on the problem in Figure 4 is given by :

$$\ell(\Pi) \leq \ell(\Pi\downharpoonright_{S_2 \cup S_1}) + \ell(\Pi\downharpoonright_{S_3 \cup S_1}) \tag{3}$$

Because $S_2 \not\rightarrow S_1$ and $S_3 \not\rightarrow S_1$ hold, the result from Theorem 1 is applicable:

$$\begin{aligned}\ell(\Pi) \quad \leq \quad & \ell(\Pi\downharpoonright_{S_2})\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_2}) + \ell(\Pi\downharpoonright_{S_3})\ell(\Pi\downharpoonright_{S_1}) \\ & + \ell(\Pi\downharpoonright_{S_3}) + 2\ell(\Pi\downharpoonright_{S_1})\end{aligned} \tag{4}$$

Alternatively, since $S_2 \cup S_3 \not\rightarrow S_1$ holds, Theorem 1 can be used, as follows:

$$\ell(\Pi) \leq \ell(\Pi\downharpoonright_{S_2 \cup S_3})\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_2 \cup S_3}) + \ell(\Pi\downharpoonright_{S_1}) \tag{5}$$

Because $S_2 \not\rightarrow S_3$ and $S_3 \not\rightarrow S_2$ hold, thus application of Theorem 3 is admissible, yielding:

$$\begin{aligned}\ell(\Pi) \quad \leq \quad & \ell(\Pi\downharpoonright_{S_2})\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_3})\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_2}) \\ & + \ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_1})\end{aligned} \tag{6}$$

The bound reached by the second approach is based on our new results, and is the tightest.

Case 2 Decomposing the problem using our novel bounds does not always lead to a better result. Consider the dependency graph in Figure 5. A bound derived with Theorem 2 on a problem with such a dependency graph will be:

$$\ell(\Pi) \leq \ell(\Pi\downharpoonright_{S_3 \cup S_1}) + \ell(\Pi\downharpoonright_{S_4 \cup (S_1 \cup S_2)}) \tag{7}$$

Again, this bound can be decomposed further according to Theorem 1:

$$\begin{aligned}\ell(\Pi) \quad \leq \quad & \ell(\Pi\downharpoonright_{S_3})\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_1}) \\ & + \ell(\Pi\downharpoonright_{S_4})\ell(\Pi\downharpoonright_{S_1 \cup S_2}) + \ell(\Pi\downharpoonright_{S_4}) \\ & + \ell(\Pi\downharpoonright_{S_1 \cup S_2})\end{aligned} \tag{8}$$

Application of Theorem 3 gives:

$$\begin{aligned}\ell(\Pi) \quad \leq \quad & \ell(\Pi\downharpoonright_{S_3})\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_4})\ell(\Pi\downharpoonright_{S_1}) \\ & + \ell(\Pi\downharpoonright_{S_4})\ell(\Pi\downharpoonright_{S_2}) + 2\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_2}) \\ & + \ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_4})\end{aligned} \tag{9}$$

Alternatively, decomposing the same dependency graph using Theorem 1 and Theorem 3 will lead to a different bound. Because in the dependency graph in Figure 5 $(S_3 \cup S_4) \not\rightarrow (S_1 \cup S_2)$ holds, Theorem 1 is applicable as follows:

$$\ell(\Pi) \leq \ell(\Pi\downharpoonright_{S_3 \cup S_4})\ell(\Pi\downharpoonright_{S_1 \cup S_2}) + \ell(\Pi\downharpoonright_{S_3 \cup S_4}) + \ell(\Pi\downharpoonright_{S_1 \cup S_2}) \tag{10}$$

This bound can be decomposed further using Theorem 3:

$$\begin{aligned}\ell(\Pi) \quad \leq \quad & \ell(\Pi\downharpoonright_{S_3})\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_3})\ell(\Pi\downharpoonright_{S_2}) \\ & + \ell(\Pi\downharpoonright_{S_4})\ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_4})\ell(\Pi\downharpoonright_{S_1}) \\ & + \ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_2}) + \ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_4})\end{aligned} \tag{11}$$

Neither of the bounds in Inequality 9 and Inequality 11 dominate. Specifically, the first bound has an extra $\ell(\Pi\downharpoonright_{S_1})$ term while the second one has an extra $\ell(\Pi\downharpoonright_{S_2})\ell(\Pi\downharpoonright_{S_3})$ term.

A Tighter Bound

In this section we present a conjecture and an informal proof of it. To prove our conjecture we require the following lemma:

Lemma 1. *For a planning problem Π for which $\dot{\pi}$ is a solution and for a node S (i.e. an SCC) in G_S of Π, there exists a plan $\dot{\pi}'$ such that:*

- $n(S, \dot{\pi}') \leq \ell(\Pi\downharpoonright_S)(\Sigma_{C \in \mathcal{C}_{G_S}(S)} n(C, \dot{\pi}) + 1)$, *where $n(vs, \dot{\pi})$ is the number of vs-actions in $\dot{\pi}$, and*
- $\dot{\pi}' \preceq\!\cdot\, \dot{\pi}$, *and*
- $\forall\ S' \neq S.\ n(S', \dot{\pi}) = n(S', \dot{\pi}')$.

Proof. The proof of Lemma 1 is a constructive proof. Let $\dot{\pi}_{\overline{\mathcal{C}}}$ be a contiguous fragment of $\dot{\pi}$ that has no $\bigcup \mathcal{C}_{G_S}(S)$-actions in it. Then perform the following steps:

- By the definition of ℓ, there must be a plan $\dot{\pi}_S$ that achieves the same execution result as $\dot{\pi}_{\overline{\mathcal{C}}}\downharpoonright_S$, and satisfies $|\dot{\pi}_S| \leq \ell(\Pi\downharpoonright_S)$ and $\dot{\pi}_S \preceq\!\cdot\, \dot{\pi}_{\overline{\mathcal{C}}}\downharpoonright_S$.
- Because $D - S - \bigcup \mathcal{C}_{G_S}(S) \not\rightarrow S$ holds and using the same argument used in the proof of Theorem 1, $\dot{\pi}'_{\overline{\mathcal{C}}}(= \dot{\pi}_S \underset{S}{\between} \dot{\pi}_{\overline{\mathcal{C}}}\downharpoonright_{D - \bigcup \mathcal{C}_{G_S}(S)})$ achieves the same $D - \bigcup \mathcal{C}_{G_S}(S)$ assignment as $\dot{\pi}_{\overline{\mathcal{C}}}$, and at the same time it is a sublist of $\dot{\pi}_{\overline{\mathcal{C}}}$. Also, $n(S, \dot{\pi}'_{\overline{\mathcal{C}}}) \leq \ell(\Pi\downharpoonright_S)$ holds.
- Finally, because $\dot{\pi}_{\overline{\mathcal{C}}}$ has no $\bigcup \mathcal{C}_{G_S}(S)$-actions, no $\bigcup \mathcal{C}_{G_S}(S)$ variables change along the execution of $\dot{\pi}_{\overline{\mathcal{C}}}$ and accordingly any $\bigcup \mathcal{C}_{G_S}(S)$ variables in preconditions of actions in $\dot{\pi}_{\overline{\mathcal{C}}}$ always have the same assignment. This means that $\dot{\pi}'_{\overline{\mathcal{C}}} \underset{D - \bigcup \mathcal{C}_{G_S}(S)}{\between} \dot{\pi}_{\overline{\mathcal{C}}}$ will achieve the same result as $\dot{\pi}_{\overline{\mathcal{C}}}$, but with at most $\ell(\Pi\downharpoonright_S)$ S-actions.

Repeating the previous steps for each $\dot{\pi}_{\overline{\mathcal{C}}}$ fragment in $\dot{\pi}$ yields an action sequence $\dot{\pi}'$ that has at most $\ell(\Pi\downharpoonright_S)(n(\bigcup \mathcal{C}_{G_S}(S), \dot{\pi}) + 1)$ S-actions. Because $\dot{\pi}'$ is the result of consecutive applications of the stitching function, it is a scattered sublist of $\dot{\pi}$. Lastly, because during the previous steps, only S-actions were removed as necessary, the count of the remaining actions in $\dot{\pi}'$ is the same as their number in $\dot{\pi}$. □

Corollary 1. *Let $F(S, \dot{\pi})$ be the witness plan of Lemma 1 (according to Skolem's theorem(Hodges 1993)). We know then that:*

- $\mathsf{exec}(s, \dot{\pi}) = \mathsf{exec}(s, F(S, \dot{\pi}))$, *and*
- $n(S, F(S, \dot{\pi})) \leq \ell(\Pi\downharpoonright_S)(\Sigma_{C \in \mathcal{C}_{G_S}(S)} n(C, \dot{\pi}) + 1)$, *and*
- $F(S, \dot{\pi}) \preceq\!\cdot\, \dot{\pi}$, *and*
- $\forall\ S' \neq S.\ n(S', \dot{\pi}) = n(S', F(S, \dot{\pi}))$.

We now use Corollary 1 to prove the following theorem:

Theorem 4. *For a planning problem Π, the upper bound $\ell(\Pi)$ satisfies the following inequality:*

$$\ell(\Pi) \leq \Sigma_{S \in G_S} N(S) \tag{12}$$

where $N(S) = \ell(\Pi\downharpoonright_S)(\Sigma_{C \in \mathcal{C}_{G_S}(S)} N(C) + 1)$.

Proof. Again, our proof of this theorem follows a constructive approach where we begin by assuming we have a solution $\dot{\pi}$. The goal of the proof is to find a witness plan $\dot{\pi}'$ such that $\forall\ S \in G_S.\ n(S, \dot{\pi}') \leq N(S)$. We proceed by induction on l_S,the list of nodes in G_S, assuming that it is topologically sorted. As our graph is not empty, the base case is a singleton list $[S]$. In this case the goal reduces to finding a plan $\dot{\pi}_0$ such that $n(S, \dot{\pi}_0) \leq N(S)$ and $\dot{\pi}_0 \preceq\!\cdot\, \dot{\pi}$. Since S has no children (as it is the only node), $N(S) = \ell(\Pi\downharpoonright_S)$ and accordingly the proof follows from the definition of ℓ.

In the step case, we assume the result holds for any problem whose non-empty node list is the topologically sorted l_S. We then show that it also holds for Π, a problem whose node list is $S :: l_S$, where S has no parents (hence its position at the start of the sorted list), and l_S is non-empty. Since the node list of $\Pi\downharpoonright_{D-S}$ is l_S, the induction hypothesis applies. Accordingly, there is a solution $\dot{\pi}_{D-S}$ for $\Pi\downharpoonright_{D-S}$ such that $\dot{\pi}_{D-S} \preceq\!\cdot\, \dot{\pi}\downharpoonright_{D-S}$ and $\forall\ K \in l_S.\ n(K, \dot{\pi}') \leq N(K)$. Since $S :: l_S$ is topologically sorted, $D - S \not\rightarrow S$ holds. Therefore $\dot{\pi}'_{D-S} = \dot{\pi}_{D-S} \underset{D-S}{\between} \dot{\pi}$ is a solution for Π (using the same argument used in the proof of Theorem 1). Furthermore, $\forall K \in l_S.\ n(K, \dot{\pi}'_{D-S}) \leq N(K)$ and $\dot{\pi}'_{D-S} \preceq\!\cdot\, \dot{\pi}$. The last step in this proof is to apply F to $(S, \dot{\pi}'_{D-S})$ to get the required witness. From Corollary 1 and because the operators $=$ and $\preceq\!\cdot$ are transitive, we know that

- $\mathsf{exec}(s, \dot{\pi}) = \mathsf{exec}(s, F(S, \dot{\pi}'_{D-S}))$, and
- $n(S, F(S, \dot{\pi}'_{D-S})) \leq \ell(\Pi\downharpoonright_S)(\Sigma_{C \in \mathcal{C}_{G_S}(S)} n(C, \dot{\pi}'_{D-S}) + 1)$, and
- $F(S, \dot{\pi}'_{D-S}) \preceq\!\cdot\, \dot{\pi}$, and
- $\forall\ K \neq S.\ n(K, \dot{\pi}'_{D-S}) = n(K, F(S, \dot{\pi}'_{D-S}))$.

Since $\forall\ K \in l_S.\ n(K, \dot{\pi}'_{D-S}) \leq N(K)$ holds, then $n(S, F(S, \dot{\pi}'_{D-S})) \leq \ell(\Pi\downharpoonright_S)(\Sigma_{C \in \mathcal{C}_{G_S}(S)} N(C))$ is true. Accordingly the plan demonstrating the needed bound is $F(S, \dot{\pi}'_{D-S})$. □

Bounds Obtained

Using Theorem 4, we can compute bounds that are tighter than the ones obtained using Theorem 2 for the two dependency graphs in Figure 4 and Figure 5. These bounds can be obtained by computing $N(S)$ for every SCC in the graph in the reverse topological order of the SCCs and summing the results.

Case 1 For the dependency graph in Figure 4 we start by computing $N(S_2)$ and $N(S_3)$ which are going to be $\ell(\Pi\downharpoonright_{S_2})$ and $\ell(\Pi\downharpoonright_{S_3})$ because they both have no children. Then we compute $N(S_1)$ which will be $\ell(\Pi\downharpoonright_{S_1})\ell(\Pi\downharpoonright_{S_2}) + \ell(\Pi\downharpoonright_{S_1})\ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_1})$. Accordingly and based on Theorem 4 the bound on the problem is

$$\begin{aligned}\ell(\Pi) \quad \leq \quad & \ell(\Pi\downharpoonright_{S_1})\ell(\Pi\downharpoonright_{S_2}) + \ell(\Pi\downharpoonright_{S_1})\ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_1}) \\ & + \ell(\Pi\downharpoonright_{S_2}) + \ell(\Pi\downharpoonright_{S_3})\end{aligned} \tag{13}$$

This bound is tighter than the one obtained using Theorem 2 shown in Inequality 4.

Case 2 For the dependency graph in Figure 5 we start by computing $N(S_3)$ and $N(S_4)$ which equal $\ell(\Pi\downharpoonright_{S_3})$ and $\ell(\Pi\downharpoonright_{S_4})$, respectively, because they both have no children. Then we compute $N(S_1)$ which will be $\ell(\Pi\downharpoonright_{S_1})\ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_1})\ell(\Pi\downharpoonright_{S_4}) + \ell(\Pi\downharpoonright_{S_1})$. Finally we compute $N(S_2)$ which will be $\ell(\Pi\downharpoonright_{S_2})\ell(\Pi\downharpoonright_{S_4}) + \ell(\Pi\downharpoonright_{S_2})$. Accordingly the bound on the problem is

$$\begin{aligned}\ell(\Pi) \leq\ & \ell(\Pi\downharpoonright_{S_1})\ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_1})\ell(\Pi\downharpoonright_{S_4}) \\ & +\ell(\Pi\downharpoonright_{S_2})\ell(\Pi\downharpoonright_{S_4}) + \ell(\Pi\downharpoonright_{S_1}) + \ell(\Pi\downharpoonright_{S_2}) \\ & +\ell(\Pi\downharpoonright_{S_3}) + \ell(\Pi\downharpoonright_{S_4})\end{aligned} \tag{14}$$

This bound is tighter than the one obtained using Theorem 2 shown in Inequality 9.

Conclusion and Future Work

With this work, we believe we have launched a fruitful interdisciplinary collaboration between the fields of AI planning and mechanised mathematical verification. From the interactive theorem-proving community's point of view, it is gratifying to be able to find and fix errors in the modern research literature. Specifically, we found errors in the existing formalisation of bounds (errors that led to the statement of a false theorem), corrected the errors with a revised definition of the key notion, and then gave a mechanized proof of a key result relating to the calculation of upper bounds.

For planning systems to be deployed in safety critical applications and for autonomous exploration of space, they must not only be efficient, and conservative in their resource consumption, but also correct. We have therefore found it highly gratifying to be able to give the planning community strong assurance of the correctness of the formalisation we treated.

We have only scratched the surface so far. When a tight bound for a planning problem is known, the most effective technique for finding a plan is to reduce that problem to SAT (Rintanen 2012). Proposed reductions are constructive, in the sense that a plan can be constructed in linear time from a satisfying assignment to a formula. A key recent advance in the setting of planning-via-SAT has been the development of compact SAT-representations of planning problems. Such representations facilitate highly efficient plan search (Rintanen 2012; Robinson et al. 2009). In future work, we would like to verify the correctness of both the reductions to SAT, and the algorithms that subsequently construct plans from a satisfying assignment.

References

Bylander, T. 1994. The computational complexity of propositional strips planning. *Artif. Intell.* 69(1-2):165–204.

Hodges, W. 1993. *Model theory*, volume 42. Cambridge University Press Cambridge.

Kautz, H. A., and Selman, B. 1996. Pushing the envelope: Planning, propositional logic and stochastic search. In *Proc. 13th National Conf. on Artificial Intelligence*, 1194–1201. AAAI Press.

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

Rintanen, J., and Gretton, C. O. 2013. Computing upper bounds on lengths of transition sequences. In *IJCAI*.

Rintanen, J. 2004. Evaluation strategies for planning as satisfiability. In *Proc. 16th European Conf. on Artificial Intelligence*, 682–687. IOS Press.

Rintanen, J. 2012. Planning as satisfiability: Heuristics. *Artif. Intell.* 193:45–86.

Robinson, N.; Gretton, C.; Pham, D. N.; and Sattar, A. 2009. SAT-based parallel planning using a split representation of actions. In *ICAPS*.

Slind, K., and Norrish, M. 2008. A brief overview of HOL4. In *Theorem Proving in Higher Order Logics*, volume 5170 of *LNCS*, 28–32. Springer.

Streeter, M. J., and Smith, S. F. 2007. Using decision procedures efficiently for optimization. In *Proc. 17th Intnl. Conference on Automated Planning and Scheduling*, 312–319. AAAI Press.

Williams, B. C., and Nayak, P. P. 1997. A reactive planner for a model-based executive. In *Proc. 15th Intnl. Joint Conference on Artificial Intelligence*, 1178–1185. Morgan Kaufmann Publishers.