Stackelberg Planning:
Towards Effective Leader-Follower State Space Search

Patrick Speicher and **Marcel Steinmetz** and **Michael Backes** and
Jörg Hoffmann and **Robert Künnemann**
CISPA, Saarland University
Saarland Informatics Campus
Saarbrücken, Germany
{patrick.speicher,robert.kuennemann}@cispa.saarland, backes@mpi-sws.org, {steinmetz,hoffmann}@cs.uni-saarland.de

Abstract

Inspired by work on Stackelberg security games, we introduce Stackelberg planning, where a *leader* player in a classical planning task chooses a minimum-cost action sequence aimed at maximizing the plan cost of a *follower* player in the same task. Such Stackelberg planning can provide useful analyses not only in planning-based security applications like network penetration testing, but also to measure robustness against perturbances in more traditional planning applications (e. g. with a leader sabotaging road network connections in transportation-type domains). To identify all equilibria – exhibiting the leader's own-cost-vs.-follower-cost tradeoff – we design *leader-follower search*, a state space search at the leader level which calls in each state an optimal planner at the follower level. We devise simple heuristic guidance, branch-and-bound style pruning, and partial-order reduction techniques for this setting. We run experiments on Stackelberg variants of IPC and pentesting benchmarks. In several domains, Stackelberg planning is quite feasible in practice.

Introduction

Stackelberg security games have been extremely successful in the formalization and solution of defensive tasks regarding physical infrastructures, such as patrolling an airport (Tambe 2011). In such games, the *leader* (the defender) moves first, followed by the *follower* (the attacker). A solution (a Stackelberg equilibirum) is a leader/follower move pair that minimizes the defender's loss given optimal attacker play. The same principle has recently been applied to planning-based network security penetration testing, short *pentesting* (e. g. (Boddy et al. 2005; Lucangeli, Sarraute, and Richarte 2010; Durkota and Lisý 2014; Hoffmann 2015)): the defender chooses the placement of honeypots (fake hosts), in a way minimizing the expected reward of an attacker whose attack is thwarted in case it runs into a honeypot (Durkota et al. 2015b; 2015a; 2016). In scenarios like these, the leader may benefit from randomizing his strategy.

Here, we introduce *Stackelberg planning* games, where each of the two players is modelled as a full-scale classical planning agent. We conjecture that Stackelberg planning can be used in various applications beyond simulated pentesting as a way to measure robustness against worst-case perturbances. For example, in transportation, the "leader" may be a saboteur, or a malicious environment, planning to disrupt the road network, while the "follower" may be the agent that plans transportation in a (damaged) network. Optimal leader behavior then captures minimal damage to the road network causing maximum transportation efficacy loss, thus measuring the road network's robustness against damage given the desired transportation objectives. Similar analyses can make sense, e. g., for warehouse robotics, Mars rovers, airport ground traffic, power supply networks, pipeline networks, greenhouse conveyor belts, etc. Interest in domains like these is amply reflected in the International Planning Competition (IPC) benchmarks. We thus focus on the case where the leader's strategy is pure and can be observed by the follower.

In our Stackelberg planning formalization, the leader and follower control different actions in a classical planning task with non-negative action costs. The follower has a goal; the leader does not have a goal, and instead pursues the objective to maximize the follower's plan cost. Hence an equilibrium is a pair of leader/defender plans where the leader cannot decrease own cost without also decreasing optimal follower-plan cost. We aim at producing *all* such equilibria (akin to a Pareto frontier). We believe this is useful – compared to aggregating leader and follower costs into a single utility – as it exhibits the actual trade-off (software updates vs. data loss; damaged road segments vs. transportation cost; etc). Different variants of Stackelberg planning may, of course, be useful too and are left for future research.

Our framework per se is somewhat novel in that prior work on planning games considered general self-interested planning agents each pursueing an own goal (Bowling, Jensen, and Veloso 2003; Larbi, Konieczny, and Marquis 2007; Brafman et al. 2009).[1] Our main contribution is on the algorithmic and experimental side, however, tackling leader-follower move combinations where each "move" is a planning problem in its own right. We design *leader-follower*

[1] General game playing (e. g. (Genesereth, Love, and Pell 2005; Love, Hinrichs, and Genesereth 2008; Thielscher 2010; Haufe, Schiffel, and Thielscher 2012)) is also related, but more remotely still due to the modelling differences between planning vs. the logic-pogramming based Game Description Language.

search, interleaving two search levels where each state in the leader level of the search spawns a planning problem at the follower level. Towards efficiency at the leader level, we devise simple heuristic guidance, branch-and-bound style pruning against incumbent solutions, as well as partial-order reduction techniques. We run experiments on Stackelberg variants of IPC and pentesting benchmarks. Expectedly, a crucial performance factor in Stackelberg planning is the number of leader actions considered. In several domains, reasonably large numbers are feasible in practice.

Stackelberg Planning

We define *Stackelberg planning tasks* following the classical STRIPS framework. A Stackelberg planning task is a tuple $\Pi = (\mathcal{P}, \mathcal{A}^L, \mathcal{A}^F, \mathcal{I}, \mathcal{G}^F)$ of a set of *facts* $\mathcal{P}$, a set of *leader actions* $\mathcal{A}^L$, a set of *follower actions* $\mathcal{A}^F$, an *initial state* $\mathcal{I} \subseteq \mathcal{P}$, and the *follower goal* $\mathcal{G}^F \subseteq \mathcal{P}$. We require that $\mathcal{A}^L \cap \mathcal{A}^F = \emptyset$, and we denote by $\mathcal{A} = \mathcal{A}^L \cup \mathcal{A}^F$ the set of all actions. A *state* of Π is a subset of facts $s \subseteq \mathcal{P}$. S denotes the set of all states. Every action $a \in \mathcal{A}$ is associated with a *precondition* $pre_a \subseteq \mathcal{P}$, an *add list* $add_a \subseteq \mathcal{P}$, a *delete list* $del_a \subseteq \mathcal{P}$, and a non-negative cost $c_a \in \mathbb{R}_0^+$. An action a is *applicable* in a state s if $pre_a \subseteq s$. In that case, the state resulting from applying a to s is $s[[a]] := (s \setminus del_a) \cup add_a$. An action sequence $\langle a_1, \dots, a_n \rangle$ is applicable in a state s if a_1 is applicable in s, and $\langle a_2, \dots, a_n \rangle$ is applicable in $s[[a_1]]$. The resulting state is denoted $s[[\langle a_1, \dots, a_n \rangle]]$. The cost of an action sequence is $c_{\langle a_1, \dots, a_n \rangle} = \sum_{i=1}^{n} c_{a_i}$.

The syntax and state transition semantics just specified is standard classical planning except for the partitioning of actions across the leader L and follower F. The particularities of our Stackelberg planning framework pertain to the task's solutions, which we define as follows.

A *leader strategy* is a sequence of leader actions π^L applicable in $\mathcal{I}$. We denote by $S^L \subseteq S$ the set of all states reachable from $\mathcal{I}$ through a leader strategy. Let $s \in S^L$ be any such state. The leader's *best move* to s is a leader strategy π^{L*} to s whose cost is minimal among all leader strategies ending in s; we denote that cost by $L^*(s)$. A *follower strategy* in s is a sequence of follower actions π^F that is applicable in s, and that achieves the follower goal, i. e., $\mathcal{G}^F \subseteq s[[\pi^F]]$. The follower's *best response* in s is a follower strategy π^{F*} in s with minimal cost; we denote that cost by $F^*(s)$, or $F^*(s) = \infty$ if no follower strategy exists.

The leader's objective is to minimize her own cost L^*, while maximizing the cost F^* of the follower's best response. We capture the trade-off between these two objectives through the set of *equilibria* where L^* cannot be reduced without also reducing F^*. Precisely, we say that a state $s \in S^L$ is an equilibrium if it is not *dominated* by any other state $t \in S^L$, where t dominates s if $(L^*(t), F^*(t))$ dominates $(L^*(s), F^*(s))$, and a pair (L, F) dominates a pair (L', F') if $L \leq L'$ and $F \geq F'$ and at least one of these two inequalities is strict. The *solution* to a Stackelberg planning task is the set $S^* \subseteq S^L$ of all equilibria.

Observe that each equilibrium state s characterizes a subset of equlibrium strategy pairs, namely the pairs (π^{L*}, π^{F*}) of best move/response pertaining to s. In this sense, our definition of equilibria in terms of states is equivalent to one in terms of strategy pairs, but is more compact.

```
procedure leader-follower-search($\Pi$)
  $\hat{S} := \emptyset$;
  let Open be an empty queue;
  push node $N$ with $N.s := \mathcal{I}$ and $N.L := 0$ onto Open;
  while Open $\neq \emptyset$ do
    pop $N$ from Open;
    /* Follower-Search Pruning (next Section) */
    run an optimal planner to compute $F^*(N.s)$;
    if $N$ is not dominated by any $\hat{N} \in \hat{S}$
      remove from $\hat{S}$ every $\hat{N} \in \hat{S}$ dominated by $N$;
      $\hat{S} := \hat{S} \cup \{N\}$;
    for every $a \in \mathcal{A}^L$, $pre_a \subseteq N.s$ do
      /* Partial-Order Reduction (Section after next) */
      let $N'$ be a new search node;
      $N'.s := N.s[[a]]$; $N'.L := N.L + c_a$;
      if already generated $N'.s$ with path cost $\leq N'.L$
        continue
      /* Leader-Search Pruning (next Section) */
      push $N'$ onto Open;
  return $\{\hat{N}.s \mid \hat{N} \in \hat{S}\}$;
```

Figure 1: Leader-follower search. Search nodes N contain the state $N.s$ and the leader path cost $N.L$ to s. They also cache the follower's best-response cost $F^*(N.s)$.

We remark that previous work on Stackelberg security games (Tambe 2011; Durkota et al. 2015b), like the original definition of Stackelberg games (von Stackelberg 1934), defines overall utility as a function of both, the leader and follower strategies. An equilibrium is then a strategy pair where the follower strategy is a best response, and the overall leader utility is optimal among such strategy pairs. As discussed in the introduction, our definition aims instead at avoiding the aggregation of leader costs with follower costs, to exhibit the full trade-off S^* between these two functions.

Leader-Follower Search

As previously indicated, our base search algorithm, *leader-follower search*, is a two-layer search solving a classical planning task optimally in every node of a state space search at the leader level. We will devise pruning and partial-order reduction techniques in the following sections. The basic algorithm is depicted in Figure 1.

The algorithm is straightforward. It explores the entire leader state space, calling an optimal planner at the follower level for each candidate state. The only major addition is the maintenance of the incumbent solution $\hat{S}$: the subset of non-dominated search nodes found thus far. Dominance over search nodes here is defined in terms of $N.L$ and $F^*(N.s)$: the former replaces $L^*(N.s)$ which is (in general) not known during the search, while the latter is already cached in the respective search nodes at the time required.

The algorithm guarantees, for every $s \in S^L$, that eventually a search node N is expanded where $N.s = s$ and $N.L = L^*(s)$. Correctness follows immediately from that:

Proposition 1. *Upon termination of leader-follower search,* $\{\hat{N}.s \mid \hat{N} \in \hat{S}\} = S^*$.

Observe that, to attain this guarantee, the search is exhaustive: prior to termination, *every* state $s \in S^L$ is expanded at least once, regardless of the expansion order. However, a good expansion order is crucial for anytime behavior and for the efficacy of the pruning techniques we introduce below. (Near-)equilibria states should be found early on.

The latter poses an interesting challenge to the generation of heuristic functions, namely a new type of estimation that one may baptize *Stackelberg heuristic functions*, based on relaxed Stackelberg planning problems that combine an optimistic estimation of leader costs with a pessimistic estimation of associated follower costs. There are at least two desirable outcomes of such estimation processes: a) for guiding leader-follower search, an estimate of the distance to the nearest equilibirum state; b) for admissible pruning, bounds on the leader/follower cost pairs still achievable below a given state. The comprehensive investigation of these questions has, in our view, the potential for an entire sub-area of AI Planning. We discuss b) in the next section, devising first pruning techniques; towards a) we implemented simple heuristics without any lookahead on the leader side, preferring leader states with higher follower plan cost estimates.

Branch-and-Bound Style Pruning

We define two pruning criteria for leader-follower search:

1. Follower-search pruning prunes unnecessary calls to the follower level, and can be done given the availability of an upper bound on follower cost.
2. Leader-search pruning prunes entire unnecessary search branches at the leader level. In order to identify such branches, bounds on the trade-off between leader and follower cost are required.

Follower-Search Pruning

Follower-search pruning in Figure 1 identifies N that are necessarily dominated by some $\hat{N} \in \hat{S}$. For such N, we do not need to compute $F^*(N.s)$. This is beneficial given the complexity of such a computation (optimal planning), and the number of times the computation will need to be performed without pruning (for every search node N).

To permit the desired pruning, it is enough to have an upper bound $\mathsf{up}^F(s)$ on the follower's best-response cost in a state s, i. e., $\mathsf{up}^F(s) \geq F^*(s)$. Namely, a given search node N is necessarily dominated by some $\hat{N} \in \hat{S}$ if N is dominated by $\hat{N}$ even under the optimistic assumption (from the leader's point of view) that $F^*(N.s) = \mathsf{up}^F(N.s)$.

Upper bounds $\mathsf{up}^F(s)$ can be computed, for example, by running a satisficing planner to compute any (not necessarily optimal) follower strategy. A much cheaper method, that we use in our experiments, is to derive $\mathsf{up}^F(s)$ from $F^*(t)$ for a parent state t of s, $s = t[[a]]$, using the following observation: If the follower's best-response π^{F*} in state t is not affected by a, i. e., π^{F*} still constitutes a follower strategy in s, then $F^*(s) \leq F^*(t)$, and hence we can use $\mathsf{up}^F(s) := F^*(t)$ as the upper bound. If a does invalidate π^{F*}, we use the trivial upper bound $\mathsf{up}^F(s) = \infty$.

Leader-Search Pruning

Leader-search pruning is more ambitious than follower-search pruning. Rather than identifying nodes necessarily dominated by $\hat{S}$, it identifies nodes whose *descendant nodes* are necessarily dominated by $\hat{S}$. Given some node N' (the new search node in Figure 1), we need to show that *for every descendant node N_0 of N', there exists a node $\hat{N} \in \hat{S}$ that dominates N_0*. If we succeed in showing this, then N' can be pruned.

Observe that assessing this property necessarily requires information about the trade-off between leader-cost vs. follower-cost, below N': How much does the leader still need to spend in order to increase follower cost, and by how much? Is it possible to obtain a new trade-off not dominated by any of the trade-offs currently contained in $\hat{S}$?

A conceptually simple approach could be derived through Stackelberg heuristic functions of type b) discussed in the previous section. Precisely, one could strive to design a function h^{Stackel} mapping N' to a set H of (L^+, F^+) pairs optimistically approximating the available trade-offs below N', in that, for every descendant node N_0 of N', there exists $(L^+, F^+) \in H$ that dominates $(N_0.L, F^*(N_0.s))$. Given such an estimation $h^{\text{Stackel}}(N') = H$, one can prune N' if, for every pair $(L^+, F^+) \in H$, there exists $\hat{N} \in \hat{S}$ such that $(\hat{N}.L, F^*(\hat{N}.s))$ dominates (L^+, F^+).

While natural, the computational feasibility of this approach appears questionable, as an entire Pareto frontier needs to be estimated. To derive a more feasible approach, we instead perform the assessment for the concrete incumbent solution $\hat{S}$, estimating whether or not it will be possible to beat these trade-offs. We formulate such estimation through an upper bound $\mathsf{up}^L(N', B)$ on the follower's best-response in any node reachable, for the leader, from N' within cost B. The advantage of this formulation is that suitable values for B can be gleaned directly from $\hat{S}$:

Theorem 1. *Every descendant of N' is dominated by some $\hat{N} \in \hat{S}$ if the following conditions are satisfied:*

(i) *$\hat{S}$ is not empty,*

(ii) $\mathsf{up}^L(N', \infty) \leq \max_{\hat{N} \in \hat{S}} F^*(\hat{N}.s)$,

(iii) $N'.L > 0$ *or* $\mathsf{up}^L(N', 0) < F^*(\hat{N}.s)$ *where $\hat{N} \in \hat{S}$ is such that $\hat{N}.L = 0$, and*

(iv) *for every $\hat{N}_1 \in \hat{S}$ with $\hat{N}_1.L > 0$ and $\hat{N}_1.L \geq N'.L$, there exists $\hat{N}_2 \in \hat{S}$ such that $\hat{N}_2.L < \hat{N}_1.L$ and* $F^*(\hat{N}_2.s) \geq \mathsf{up}^L(N', \hat{N}_1.L - N'.L)$.

Proof. Assume for contradiction that N' can reach a node N_0 that is not dominated by any $\hat{N} \in \hat{S}$, even though the conditions (i) – (iv) are all satisfied. If $\hat{S}$ is not empty, then observe that $\hat{S}$ is guaranteed to contain at least one node $\hat{N} \in \hat{S}$ with $\hat{N}.L = 0$. This holds because the algorithm in Figure 1 always adds the node with $N.s = \mathcal{I}$ and $N.L = 0$ to $\hat{S}$, and this node may only be replaced by some other node reached via a path with cost 0. If $N_0.L = 0$, then in particular $N'.L = 0$, and since N_0 is not dominated by $\hat{N}$, it holds that $F^*(N_0.s) \geq F^*(\hat{N}.s)$. However as (iii) is satisfied, either $N'_0.L > 0$ for all nodes N'_0 reachable from N', or $F^*(N'_0.s) \leq \mathsf{up}^L(N', 0) < F^*(\hat{N}.s)$ for all nodes N'_0 reachable from N' with $N'_0.L = 0$, i. e., N'_0 is dominated by $\hat{N}$. Therefore, $N_0.L$ must be larger than 0. On the other hand, since (ii) is satisfied, $\hat{S}$ must contain a node $\hat{N}$ such that $F^*(N_0.s) \leq \mathsf{up}^L(N', \infty) \leq F^*(\hat{N}.s)$. Since N_0 is not dominated by any such $\hat{N}$, it immediately follows that $N_0.L \leq \hat{N}.L$. Let $\hat{N}_1 \in \hat{S}$ be the node with minimal $\hat{N}_1.L$ value among the nodes with $\hat{N}_1.L \geq N_0.L$. As we have shown before, $N_0.L > 0$, and therefore $\hat{N}_1.L > 0$. Because of (iv), there must hence exist $\hat{N}_2 \in \hat{S}$ such that $\hat{N}_2.L < \hat{N}_1.L$ and $F^*(\hat{N}_2.s) \geq \mathsf{up}^L(N', \hat{N}_1.L - N'.L) \geq F^*(N_0.s)$. However, the selection of $\hat{N}_1$ implies that $\hat{N}_2.L < N_0.L$. Thus, $\hat{N}_2$ actually dominates N_0, what is a contradiction to the assumption. We conclude that if N' can reach a node N_0 that is not dominated by any $\hat{N} \in \hat{S}$, then one of the conditions (i) – (iv) must be violated. □

At first glance, one might wonder why condition (iv) requires the consideration of two nodes $\hat{N}_1, \hat{N}_2 \in \hat{S}$. To see why, assume a node N' that can only reach a single node N_0 whose L and F^* values are both either strictly smaller or strictly larger than those of $\hat{N}$, for every $\hat{N} \in \hat{S}$. When considering only a single node $\hat{N} \in \hat{S}$ in the reachability approximation, all we will find out is that within a cost budget of $\hat{N}.L$, N' cannot reach any node where the follower's best-response cost is at least as large as $F^*(\hat{N}.s)$. Nevertheless, N_0 is not dominated by any $\hat{N} \in \hat{S}$, and thus N' should be considered for expansion.

As of now, we use a very simple upper bound $\mathsf{up}^L(N', B)$: let $\mathsf{up}^L(N', B) := \mathsf{up}^L_\forall$ where $\mathsf{up}^L_\forall$ is an upper bound on the follower's best-response in *every* state. For example, using the trivial such upper bound $\mathsf{up}^L_\forall := \infty$, as soon as $\hat{S}$ contains a node $\hat{N}$ with $F^*(\hat{N}.s) = \infty$, i. e., a state in which the follower can no longer achieve her goal, all nodes N' with $N'.L > \hat{N}.L$ will be pruned. To derive tighter upper bounds $\mathsf{up}^L_\forall$, in Stackelberg planning tasks where the leader can never entirely preclude the follower from reaching the goal, one can over-approximate the overall harm the leader can do to the follower. In our experiments, we use a simple technique based on the delete relaxation. We first perform a delete-relaxed reachability fixed point for the initial state, considering only leader actions. We collect the facts p deleted by at least one leader action applicable in the fixed point. We remove all these facts p from the initial state, and we compute a follower best-response (an optimal follower strategy) given this reduced initial state.

Partial-Order Reduction

As an additional means to reduce the leader search space, we adapt a well known partial-order reduction technique, *strong stubborn sets (SSS)* (Valmari 1989; Wehrle and Helmert 2012; Wehrle et al. 2013; Wehrle and Helmert 2014). The basic idea behind SSS is to exploit action independency to identify, for a given search state s, a subset A of applicable actions so that expanding s using only A suffices to retain the standard notions of completeness and optimality. Here, we adapt this idea to preserve the guarantee given by Proposition 1, i. e., that $\{N.s \mid N \in \hat{S}\} = S^*$ upon termination of leader-follower search.

The construction of an SSS follows three steps: (i) one starts with a *disjunctive action landmark*, i. e., a set of actions one of which is needed to achieve the goal; (ii) one backchains over open action preconditions to actions applicable in s; and (iii) for each applicable action one includes all interfering actions. (ii) and (iii) remain the same in our context, and we specify them formally below. The main issue here is (i), because in Stackelberg planning the goal of the leader is not given explicitly. So we need to find a different way to seed the SSS with "something the leader will definitely have to do in order to make progress".

To that end, say that $s = N.s$ is the leader search state about to be expanded in leader-follower search as per Figure 1. Say that π^{F*} is the follower's best-response in s. Consider any minimal-cost path $\mathcal{I} = s_0, a_1, s_1, \dots, a_n, s_n$ through $s = s_i$ to an equilibrium $s_n \neq s$. Observe that *either the path behind s_i must contain a leader action that renders π^{F*} inapplicable, or all actions on that path must have cost 0:* clearly, if both these conditions are false, then s dominates s_n in contradiction. We thus seed our SSS with the set of all leader actions that may disable π^{F*} and handle 0-cost actions separately.

In detail, we define SSS for leader-follower search as follows. Let $a_1, a_2 \in \mathcal{A}^L$, $a_1 \neq a_2$, be two leader actions. Following classical planning definitions of SSS we say that a_1 *disables* a_2 if $pre_{a_2} \cap del_{a_1} \neq \emptyset$; a_1 *enables* a_2 if $add_{a_1} \cap pre_{a_2} \neq \emptyset$ and a_1 does not disable a_2; a_1 and a_2 *conflict* with each other if $add_{a_1} \cap del_{a_2} \neq \emptyset$ or vice versa $add_{a_2} \cap del_{a_1} \neq \emptyset$; a_1 and a_2 *interfere* if they conflict with each other, or either disables the other. Furthermore, we define regression in the usual manner, namely for a set of facts $G \subseteq \mathcal{P}$ and an action $a \in \mathcal{A}$, the regression of G through a is defined if $add_a \cap G \neq \emptyset$ and $del_a \cap G = \emptyset$. If defined, the regression is given by the set $\text{Regress}(G, a) = (G \setminus add_a) \cup pre_a$.

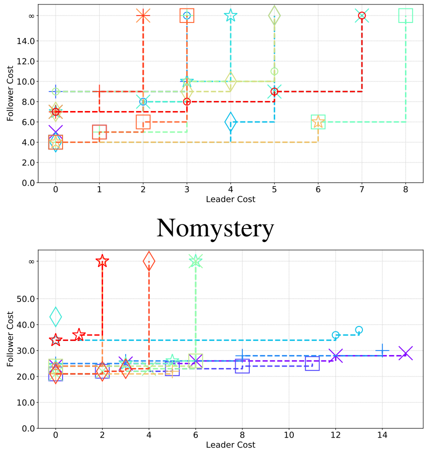
Now, $A \subseteq \mathcal{A}^L$ is an *SSS in s for π^{F*}* if the following conditions are satisfied:

(i) A contains every leader action $a \in \mathcal{A}^L$ where $del_a \cap \text{Regress}(\mathcal{G}^F, \pi^{F*}) \neq \emptyset$;

(ii) for every $a \in A$ not applicable in s, A contains all actions that enable a; and

(iii) for every $a \in A$ applicable in s, A contains all actions that interfere with a.

Note that $\text{Regress}(\mathcal{G}^F, \pi^{F*})$ gives the minimal set of facts that is required for the follower to achieve her goal through π^{F*}. Facts appearing e. g. in the precondition of some action a in π^{F*}, but not in $\text{Regress}(\mathcal{G}^F, \pi^{F*})$, are not relevant for the applicability of π^{F*} because, by definition of regression, these are added by some other action appearing before a anyway. Considering also such facts would not harm the



Nomystery

Visitall

Figure 2: Visualization of Pareto frontiers in Nomystery and Visitall. Each line shows the Pareto frontier of an instance, each data point an entry in the frontier.

correctness of the SSS construction, but may lead to larger sets A, and hence less pruning. With the same arguments as in classical planning (Wehrle and Helmert 2012), we obtain:

Theorem 2. *Let A be an SSS in s for π^{F*}. Then, for every leader strategy $a_1, \dots, a_n$ in s where π^{F*} is no longer applicable in $s[[a_1, \dots, a_n]]$, there exists $1 \leq i \leq n$ such that $a_i \in A$, and $a_i, a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n$ is applicable in s, and $s[[a_1, \dots, a_n]] = s[[a_i, a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n]]$.*

As an immediate consequence, we can ignore in the expansion of s all actions $a \in \mathcal{A}^L$ whose cost is larger than 0 and which are not contained in A.

Experiments

To evaluate the scalability of our Stackelberg planning algorithms, we extended several benchmark domains from the International Planning Competition (IPC). Due to the intrinsic difficulty of solving classical planning tasks and hence Stackelberg planning tasks optimally, we also report results for a *satisficing* variant of the leader-follower search algorithm, where we used an inadmissible search configuration to find follower strategies. Our implementation is based on Fast Downward (Helmert 2006). We ran all experiments on 2.20 GHz Intel Xeon machines, with runtime/memory limits of 30 mins/4 GB (as typically used in IPC setups).

Benchmarks

To design our benchmarks, we adapted three domains relating to transportation, namely *Logistics* (IPC'98), *Nomystery* (IPC'11), and *Visitall* (IPC'14); the puzzle game *Sokoban* (IPC'14); and a simulated pentesting domain *Pentest* adapted from prior work on this application.

Given an individual classical planning base task (a benchmark domain instance), we obtain Stackelberg planning tasks as follows. We set the follower's actions and goal to be that of the base task. We design a set of leader actions suitable for the domain and task (see details below). To control the complexity stemming from the number of leader actions, we generate multiple Stackelberg planning instances, setting that number within $\{1, 2, 3, 4, 5, 6, 8, 10, 12, 16, 25, 32, 50, 64, 128, 256, 512, 1024, 2048, 4096\}$ up to the number of leader actions present in the task. We hence get up to 20 different Stackelberg planning instances per base task. In total, our benchmark suite contains 3263 instances.

In the three transportation-like domains, the leader actions are designed based on the idea of evaluating road-network robustness.[2] Any one leader action removes one road connection (an edge in the network graph) from the network. We design two different variants (that we will refer to by their number throughout this section): (i) the leader can execute any such action at any time; (ii) in order to remove the connection between X and Y, the leader must be located at either X or Y. In (ii), the leader can change her location via additional leader actions moving on the same road network; once a connection is removed, the leader cannot use it anymore, rendering even the leader planning problem on its own non-trivial, requiring to reason about interferences.

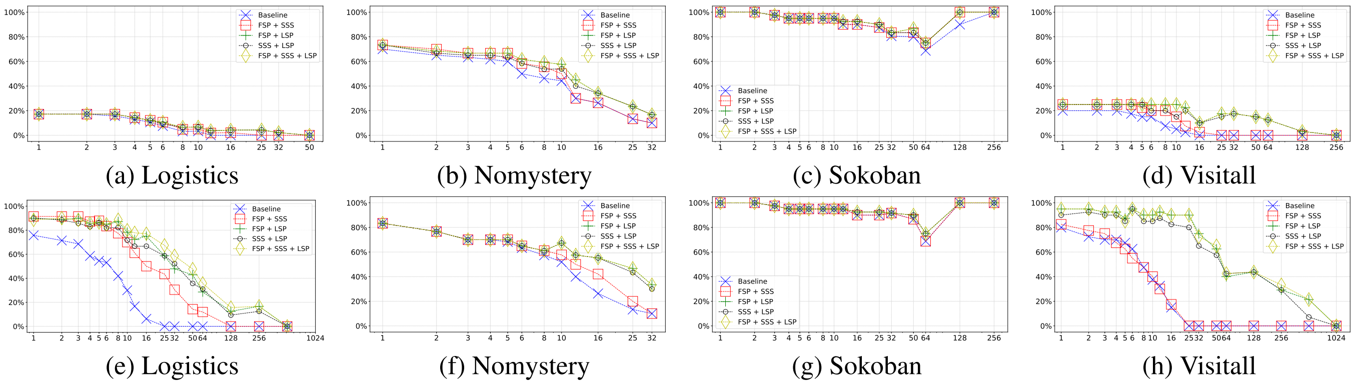
In Sokoban, we test for robustness against modifications to the board. Here, each leader action allows to block one cell of the grid by placing a wall. We again consider two variants: (i) there is no precondition to placing a wall; and (ii) walls may only be placed in a cell adjacent to other walls.

For generating the Pentest base-instances, we adapted the MDP generator by Steinmetz et al. (2016) to output classical planning tasks. We applied two changes: (i) each exploit action is now guaranteed to succeed, and (ii) the cost of an exploit action is set to the negative logarithm of the exploit success probability. This changes the overall objective from the MDP problem of finding a *policy* maximizing the probability of achieving the goal, to the classical planning problem of finding an *action sequence* maximizing that probability. Steinmetz et al.'s generator allows to scale both the network size N and the number of exploits E in the network. We used E := N for simplicity, and scaled N from 50 to 600 in steps of 50. For each N we generated three instances using a different random seed, resulting in a total number of 36 base-instances. We designed the leader actions to correspond to the activities of an administrator trying to secure the network. Namely, we consider two possible *fixes*: closing a specific port on a specific host as a firewall rule; and patching a software on a specific host, effectively removing the associated exploits.

Pareto Frontier Examples

The solution for a Stackelberg planning instance is a Pareto frontier S^*, which visualizes the leader/follower-cost tradeoff. To exemplarily illustrate that this S^* can truly be of interest, we show frontiers of several Nomystery and Visitall instances in Figure 2. We used variant (ii) for both domains, so the leader must be located at an adjacent loca-

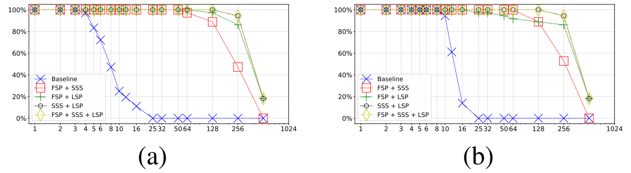
[2] We remark on the side that this application was inspired by a road-network break-down in our own city, blocking reasonable access to the university campus for 3 weeks.



(a) Logistics (b) Nomystery (c) Sokoban (d) Visitall

(e) Logistics (f) Nomystery (g) Sokoban (h) Visitall

Figure 3: Percentage of solved instances, as a function of the number of blockable cells (Sokoban) respectively removable road-map connections (other domains). Plots (a) – (d) report the results for optimal Stackelberg planning, plots (e) – (h) for greedy Stackelberg planning.



(a) (b)

Figure 4: Percentage of solved Pentest instances, as a function of the number of fixes. Optimal (a), greedy (b).

tion to remove a connection and there are leader move actions. Each individual line represents a complete frontier and every marker represents an entry in the frontier, i.e. a leader/follower-cost pair (L, F). One can observe many instances where it is possible to increase F to ∞ with very little L, but also instances where one cannot increase F at all, only by a small margin, or only with high L.

Optimal Stackelberg Planning

We next shed light on the feasibility of optimal Stackelberg planning, and on the impact of our pruning methods. To find optimal follower responses, we run A* with the state-of-the-art admissible heuristic LM-cut (Helmert and Domshlak 2009). Due to the nature of Stackelberg planning, and our benchmarks in particular, there is a high chance to exclude all follower strategies. To be able to perform well in such cases – i. e., to effectively prove the follower problem unsolvable – we use, in addition to LM-cut, the PDB heuristic that was part of Aidos, the winning portfolio in the 2016 Unsolvability IPC (Seipp et al. 2016). We noticed early on in our experiments that the PDB heuristic helps a lot in a few instances, although the overall impact is rather small.

At the leader level, we run iterative deepening depth-first search. We order open states by decreasing follower optimal response cost (preferring states with higher follower cost). If there is no follower strategy for a state and no 0-cost leader actions, then we can trivially discard this state. We refer to the branch-and-bound follower-search pruning, as discussed before, by *FSP*; to leader-search pruning based on the upper bound $\text{up}^L_\forall$ by *LSP*; and to our strong stubborn sets technique by *SSS*. We report results for different combinations of these techniques (including none, as a baseline) to examine their impact.

The coverage results are shown in Figures 3 and 4. Note that we aggregated both types of leader actions in Figures 3, because the results were quite similar. In almost all Sokoban instances (Figure 3 (c)), few changes to the board suffice to prevent the follower from reaching her goal, yielding a shallow leader-level search tree. All our configurations tackle such cases effectively, and relative coverage is between 80% and 100% throughout. (The increase from 64 to 128 leader actions is an artifact of a decreasing number of instances for an increasing number of leader actions.)

In contrast, in Nomystery (Figure 3 (b)) and Pentest (Figure 4 (a)), the impact of the number of leader actions is large, as one would expect. In Logistics and Visitall (Figure 3 (a) and (d)), coverage is extremely low even with few leader actions. This is due to the computational cost of computing optimal follower responses, which is prohibitive in these two domains.

Comparing performance across different configurations, the coverage differences are small. The largest difference occurs in Pentest, where the baseline configuration without any pruning falls substantially short starting from 4 leader actions. The overall best-performing configuration is the one where all pruning techniques are enabled. The configuration without SSS performs almost equally well except in Pentest.

A closer look at the performance of the different configurations, beyond the coarse measurement afforded by coverage, shows that actually all our pruning methods yield vast runtime advantages over the baseline. Figure 5 shows per-instance runtime scatter plots. Clearly, runtime reductions by orders of magnitude are common for all three methods.

Comparing LSP vs. SSS in Figure 5(d), we see that LSP has a noticeable advantage in larger instances, which also explains the slight edge in coverage. It is worth pointing out here that LSP can only be useful if the computed $\text{up}^L_\forall$ indeed

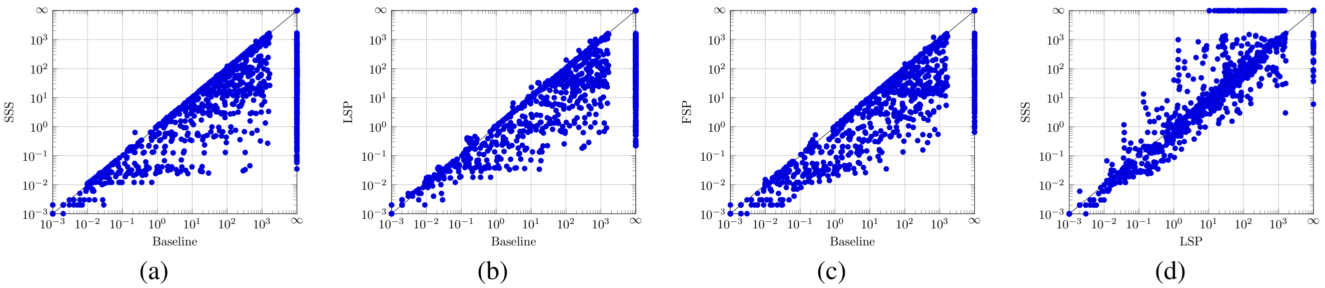


Figure 5: Per instance comparison of runtime for (a) no pruning (x) vs. SSS (y); (b) no pruning (x) vs. LSP (y); (c) no pruning (x) vs. FSP (y); (d) LSP (x) vs. SSS (y).

corresponds to the maximal follower response cost in S^*. In our experiments, this was the case in all instances where S^* could be fully constructed. This is, however, not guaranteed to happen in general; in other cases the other two pruning techniques may become more important. Regarding FSP, we remark that this technique is strongest in our transportation domains of type (ii), i.e., those with leader actions moving along the road map. As such actions never invalidate the follower's best response, this leads to perfect upper bounds and thus frequent follower-search pruning.

Greedy Stackelberg Planning

We noticed that the computational cost of the optimal follower search is critical in many cases, and thus ran experiments with a greedy variant of our algorithms. The only change is that we use a satisficing instead of an optimal planner at the follower level. This makes the follower-level search way more effective, at the price of no longer giving a guarantee on the correctness of the output (the pareto frontier may be faulty as some leader states may have been assigned an overly high follower cost). Note that the curve induced by the greedy pareto frontier solution can only be on or above the curve induced by the optimal solution.

Specifically, at the follower level we now run the commonly used baseline satisficing planner, Fast Downward's greedy best-first search (GBFS) with a dual queue for preferred operators, using the inadmissible heuristic h^{FF} (Hoffmann and Nebel 2001). To effectively prove follower problems unsolvable, as before we use Aidos' PDB heuristic.

The coverage results are included with the ones for optimal Stackelberg planning in Figures 3 and 4. We observe a dramatic coverage increase in those domains – Logistics and Visitall– where computing optimal follower responses is prohibitively costly. In the other domains, the difference to optimal Stackelberg planning is relatively small. For Pentest and Nomystery, the follower level is not the bottleneck of the overall search; for Sokoban, even optimal Stackelberg planning has close to maximal coverage anyway.

Switching from optimal to satisficing follower search in Nomystery increases coverage by about 10 percentage points. The influence of the follower level is marginal in this domain, and LSP pruning has a much higher impact (FSP + SSS and the baseline fall short of the other configurations). Similarly for Pentest, where the coverage gain from optimal to greedy Stackelberg planning is even less pronounced.

That difference is huge though in Logistics and Visitall. In both domains, the coverage of the best greedy configuration (FSP + SSS + LSP) starts at about 90% and decreases linearly in the number of leader actions. Again, only the baseline and FSP + SSS configurations are considerably weaker.

For the Visitall domain, we discovered that the suboptimality of greedy follower-level planning positively influences the impact of LSP pruning: with a satisficing follower planner, it can happen that the upper bound $\mathsf{up}^L_\forall$, though obtained from a very coarse approximation, is *smaller* than the follower costs computed by the satisficing planner. This is indeed often the case in Visitall, where our follower planner is prone to choose highly sub-optimal routes across the map. To our leader-level search, it then appears that many of the follower responses encountered in search are worse than the already known upper bound, so that LSP pruning cuts off the search very early. Computational performance thrives on this, yet the returned Pareto frontiers are highly incomplete.

Importantly, Visitall is the only domain in which we observed this phenomenon. To back this up, we compared the quality of the greedy vs the optimal solutions for all domains. We measured the size of the areas beneath the optimal and greedy pareto frontier curves as a comparison. With the most competitive configuration (FSP + SSS + LSP), the areas under the greedy curves are on average about 10% larger than the areas under the optimal ones. An exception is the visitall domain where the area size is doubled.

Conclusion

Stackelberg planning is, we believe, an exciting new variant of planning that may be of high practical value in a large range of potential planning applications. First, it allows modelling defenses against attackers encoded as planning agents. Second, it subsumes a notion of robustness against sabotage and against worst-case perturbances by an environment. From an algorithmic point of view, it provides an interesting middle ground between full-scale game-theoretic planning and classical planning, generalizing the latter to a single adversarial exchange of action sequences. We expect that many of the successful algorithmic techniques from classical planning can be lifted to this setting, and indeed our first results presented herein suggest that this is so.

Acknowledgments. This work was supported by the German Federal Ministry of Education and Research (BMBF) through funding for the Center for IT-Security, Privacy and Accountability (CISPA, grant no. 16KIS0656).

References

Boddy, M.; Gohde, J.; Haigh, T.; and Harp, S. 2005. Course of action generation for cyber security using classical planning. In Biundo, S.; Myers, K.; and Rajan, K., eds., *Proceedings of the 15th International Conference on Automated Planning and Scheduling (ICAPS-05)*, 12–21. Monterey, CA, USA: Morgan Kaufmann.

Bowling, M. H.; Jensen, R. M.; and Veloso, M. M. 2003. A formalization of equilibria for multiagent planning. In Gottlob, G., ed., *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, 1460–1462. Acapulco, Mexico: Morgan Kaufmann.

Brafman, R. I.; Domshlak, C.; Engel, Y.; and Tennenholtz, M. 2009. Planning games. In Boutilier, C., ed., *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, 73–78. Pasadena, California, USA: Morgan Kaufmann.

Durkota, K., and Lisý, V. 2014. Computing optimal policies for attack graphs with action failures and costs. In *7th European Starting AI Researcher Symposium (STAIRS'14)*.

Durkota, K.; Lisy, V.; Kiekintveld, C.; and Bosansky, B. 2015a. Game-theoretic algorithms for optimal network security hardening using attack graphs. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems (AAMAS'15)*, 1773–1774.

Durkota, K.; Lisy, V.; Kiekintveld, C.; and Bosansky, B. 2015b. Optimal network security hardening using attack graph games. In Yang, Q., ed., *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*. AAAI Press/IJCAI.

Durkota, K.; Lisý, V.; Kiekintveld, C.; Bosanský, B.; and Pechoucek, M. 2016. Case studies of network defense with attack graph games. *IEEE Intelligent Systems* 31(5):24–30.

Genesereth, M. R.; Love, N.; and Pell, B. 2005. General game playing: Overview of the AAAI competition. *AI Magazine* 26(2):62–72.

Haufe, S.; Schiffel, S.; and Thielscher, M. 2012. Automated verification of state sequence invariants in general game playing. *Artificial Intelligence* 187:1–30.

Helmert, M., and Domshlak, C. 2009. Landmarks, critical paths and abstractions: What's the difference anyway? In Gerevini, A.; Howe, A.; Cesta, A.; and Refanidis, I., eds., *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS'09)*, 162–169. AAAI Press.

Helmert, M. 2006. The Fast Downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

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. 2015. Simulated penetration testing: From "Dijkstra" to "Turing Test++". In Brafman, R.; Domshlak, C.; Haslum, P.; and Zilberstein, S., eds., *Proceedings of the 25th International Conference on Automated Planning and Scheduling (ICAPS'15)*. AAAI Press.

Larbi, R. B.; Konieczny, S.; and Marquis, P. 2007. Extending classical planning to the multi-agent case: A game-theoretic approach. In *Proceedings of the 9th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'07)*, 731–742.

Love, N. C.; Hinrichs, T. L.; and Genesereth, M. R. 2008. General game playing: Game description language specification. Technical Report LG-2006-01, Stanford Logic Group.

Lucangeli, J.; Sarraute, C.; and Richarte, G. 2010. Attack planning in the real world. In *Proceedings of the 2nd Workshop on Intelligent Security (SecArt'10)*.

Seipp, J.; Pommerening, F.; Sievers, S.; and Wehrle, M. 2016. Fast downward aidos. In *UIPC 2016 planner abstracts*, 28–38.

Steinmetz, M.; Hoffmann, J.; and Buffet, O. 2016. Goal probability analysis in mdp probabilistic planning: Exploring and enhancing the state of the art. *Journal of Artificial Intelligence Research* 57:229–271.

Tambe, M. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

Thielscher, M. 2010. A general game description language for incomplete information games. In Fox, M., and Poole, D., eds., *Proceedings of the 24th National Conference of the American Association for Artificial Intelligence (AAAI'10)*, 994–999. Atlanta, GA, USA: AAAI Press.

Valmari, A. 1989. Stubborn sets for reduced state space generation. In *Proceedings of the 10th International Conference on Applications and Theory of Petri Nets*, 491–515.

von Stackelberg, H. 1934. *Market Structure and Equilibrium*. Springer-Verlag.

Wehrle, M., and Helmert, M. 2012. About partial order reduction in planning and computer aided verification. In Bonet, B.; McCluskey, L.; Silva, J. R.; and Williams, B., eds., *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS'12)*. AAAI Press.

Wehrle, M., and Helmert, M. 2014. Efficient stubborn sets: Generalized algorithms and selection strategies. In Chien, S.; Do, M.; Fern, A.; and Ruml, W., eds., *Proceedings of the 24th International Conference on Automated Planning and Scheduling (ICAPS'14)*. AAAI Press.

Wehrle, M.; Helmert, M.; Alkhazraji, Y.; and Mattmüller, R. 2013. The relative pruning power of strong stubborn sets and expansion core. In Borrajo, D.; Fratini, S.; Kambhampati, S.; and Oddi, A., eds., *Proceedings of the 23rd International Conference on Automated Planning and Scheduling (ICAPS'13)*. Rome, Italy: AAAI Press.