Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)

Optimal Mixed Strategies for Cost-Adversarial Planning Games

Rostislav Horčík[1], **Álvaro Torralba**[2], **Pavel Rytíř**[1], **Lukáš Chrpa**[1], **Stefan Edelkamp**[1]

[1] Czech Technical University in Prague, Faculty of Electrical Engineering, Prague, Czech Republic
[2] Aalborg University, Denmark
{xhorcik, pavel.rytir, chrpaluk, edelkste}@fel.cvut.cz, alto@cs.aau.dk

Abstract

This paper shows that domain-independent tools from classical planning can be used to model and solve a broad class of game-theoretic problems we call Cost-Adversarial Planning Games (CAPGs). We define CAPGs as 2-player normal-form games specified by a planning task and a finite collection of cost functions. The first player (a planning agent) strives to solve a planning task optimally but has limited knowledge about its action costs. The second player (an adversary agent) controls the actual action costs. Even though CAPGs need not be zero-sum, every CAPG has an associated zero-sum game whose Nash equilibrium provides the optimal randomized strategy for the planning agent in the original CAPG. We show how to find the Nash equilibrium of the associated zero-sum game using a cost-optimal planner via the Double Oracle algorithm. To demonstrate the expressivity of CAPGs, we formalize a patrolling security game and several IPC domains as CAPGs.

Introduction

Research in classical planning focuses on developing domain-independent methods and tools to model and solve a broad range of planning tasks, where an agent chooses a sequence of actions to achieve a set of goals. This paper shows that these domain-independent tools can also be applied to game-theoretic problems studied within AI, particularly to security games (Tambe 2012). Many applications of security games use the language of the normal-form games working with an elementary indecomposable notion of a pure strategy, even though pure strategies can be expressed as a complex sequence of actions. Consequently, each application introduces a domain-dependent formalism capturing the inner structures of pure strategies. We demonstrate that it is possible to model this internal structure as a classical planning task. As a result, we obtain a general framework to model a broad class of security games and solve them using any cost-optimal planner combined with game-theory techniques.

We define a class of cost-adversarial planning games (CAPGs) specified by a planning task that the first player (P-player) strives to solve optimally but the action costs are influenced by the second player (C-player). More precisely, CAPGs are 2-player normal-form games, where the P-player chooses a plan and the C-player chooses a cost function from a given collection of cost functions. Even though, this interaction is simpler than other forms of multi-agent adversarial planning, where both agents can select arbitrary plans (Brafman et al. 2009; Speicher et al. 2018), it can still capture many relevant scenarios. For example, by increasing the cost of certain actions, the C-player can force the P-player to choose alternative plans. Furthermore, this allows us to consider randomized strategies, which is very relevant in some practical applications.

Cost-adversarial planning games are almost zero-sum games. Every almost zero-sum game is best-response equivalent to a zero-sum game. This allows computing the optimal mixed strategy for the P-player by computing a Nash equilibrium in the equivalent zero-sum game. Nash equilibrium (NE) is a standard solution concept for normal-form games; see e.g. (Shoham and Leyton-Brown 2009). A NE of a zero-sum game can be computed in polynomial time via a transformation to a linear program. However, this approach is unsuitable for CAPGs because the resulting linear program can be too large. Moreover, the transformation is computationally demanding. Thus to find a NE of a CAPG, we employ the Double Oracle algorithm (McMahan, Gordon, and Blum 2003) which might be seen as a combination of the column and the constraint generation method used to solve large linear programs. This approach can leverage domain-independent cost-optimal planning to solve CAPGs with large state spaces efficiently.

The paper is organized as follows. After we recall all necessary definitions and results from planning and game theory, we introduce cost-adversarial planning games (CAPGs). To demonstrate the expressivity of CAPGs, we discuss two case studies showing what kind of problems one can model as CAPGs. In particular, we introduce a domain inspired by security games, which models wildlife protection against illegal poachers (Yang et al. 2014). Our experimental evaluation shows that this approach is viable, and NEs can be found in most cases where current cost-optimal planners are capable of solving the underlying planning task of the P-player.

Background

We assume the reader is familiar with optimal classical planning. To keep our definition general, we define cost-adversarial planning games over labelled transition systems

so that the definition is independent of the formalism used to specify the underlying planning tasks.

Every planning task Π induces a **labeled transition system** (LTS) that is a tuple $\Theta_\Pi = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$, where $\mathcal{S}$ is a finite set of **states**, $\mathcal{L}$ is a finite set of **labels**, $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{L} \times \mathcal{S}$ is a set of **transitions**, $s_I \in \mathcal{S}$ is an **initial state**, and $S_\star \subseteq \mathcal{S}$ is a set of **goal states**. We write $s \xrightarrow{l} s'$ to refer to a transition from s to s' with the label l. A sequence of labels $\pi = \langle l_1, \dots, l_n \rangle$ is called a **plan** for Θ_Π if there exists $s_{i-1} \xrightarrow{l_i} s_i \in \mathcal{T}$ for every $i \in \{1, \dots, n\}$ such that $s_0 = s_I$ and $s_n \in S_\star$. The set of all plans for Θ_Π is denoted $\mathcal{P}(\Pi)$. In optimal classical planning, there is also a cost function $c \colon \mathcal{L} \to \mathbb{R}_0^+$ assigning each label l its non-negative real-valued cost $c(l)$. Given a cost function c, we can define the cost of the plan π by $c(\pi) = \sum_{i=1}^{n} c(l_i)$. A plan π is an **optimal plan** if its cost is minimal among all plans.

We need to recall a few definitions and basic facts from game theory; for details see e.g. (Shoham and Leyton-Brown 2009). A **2-player normal-form game** (shortly game) is a quadruple $\mathcal{G} = \langle X, Y, u_1, u_2 \rangle$ where X (resp. Y) is a finite set of **pure strategies** of Player 1 (resp. Player 2), $u_1 \colon X \times Y \to \mathbb{R}$ (resp. $u_2 \colon X \times Y \to \mathbb{R}$) is a utility function of Player 1 (resp. Player 2). When the game $\mathcal{G}$ is played, both players choose simultaneously a strategy from their respective sets of pure strategies X, Y. The outcome of $\mathcal{G}$ for Player i is given by the utility function u_i. The players strive to maximize their utilities.

If a game is played repeatedly, it might be reasonable for the players to randomize their strategies in order to increase their expected utilities. A **mixed strategy** for Player 1 is a probabilistic distribution $p \colon X \to [0, 1]$. The set of all mixed strategies of Player 1 is denoted Δ_X. For a mixed strategy $p \in \Delta_X$, we define its support $\mathrm{spt}(p) = \{x \in X \mid p(x) > 0\}$. Note that a pure strategy x corresponds to a mixed strategy δ_x such that $\mathrm{spt}(\delta_x) = \{x\}$. Mixed strategies Δ_Y for Player 2 and their supports are defined analogously.

The utility function u_i can be extended to $\Delta := \Delta_X \times \Delta_Y$ by $u_i(p, q) = \sum_{x \in \mathrm{spt}(p)} \sum_{y \in \mathrm{spt}(q)} p(x) q(y) u_i(x, y)$. In other words, $u_i(p, q)$ is the expected utility of Player i if Player 1 plays p and Player 2 plays q.

An important class of games are **zero-sum** games, i.e., games where $u_1 + u_2 = 0$. In zero-sum games, it suffices to consider only a single utility function $u = -u_1 = u_2$ that Player 1 strives to minimize and Player 2 maximize.

A standard solution concept for games is Nash equilibrium defining stable pairs of mixed strategies.

Definition 1. Let $G = \langle X, Y, u_1, u_2 \rangle$ be a game. A pair of mixed strategies $\langle p^*, q^* \rangle \in \Delta$ is called a **Nash equilibrium** (NE) if for all $\langle p, q \rangle \in \Delta$ we have

$$u_1(p, q^*) \leq u_1(p^*, q^*), \qquad u_2(p^*, q) \leq u_2(p^*, q^*). \quad (1)$$

In other words, none of the players would change her strategy knowing the strategy of the opponent because they mutually play best responses against each other.

If $\mathcal{G}$ is zero-sum and we have only a single utility function $u = -u_1 = u_2$, the defining condition (1) becomes $u(p^*, q) \leq u(p^*, q^*) \leq u(p, q^*)$.

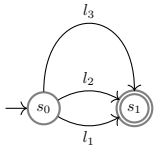


Figure 1: LTS for a simple game.

Nash equilibrium is a reasonable solution concept, especially in zero-sum games. Nash (1951) proved that each game has at least one Nash equilibrium. In fact, a game can have more than one NE. In general games, it is problematic for the players to select one among all NEs because the outcome could be different for different NEs (consider e.g. the well-known prisoners' dilemma). Nevertheless, this problem does not occur in zero-sum games. The value $u(p^*, q^*)$ is always the same for any NE $\langle p^*, q^* \rangle$. Moreover, p^* is a minimax strategy whereas q^* is maximin strategy as follows from Neumann's theorem (von Neumann 1928).

Theorem 2. *Let $\mathcal{G} = \langle X, Y, u \rangle$ be a zero-sum game and $\langle p^*, q^* \rangle$ its NE. Then*

$$u(p^*, q^*) = \min_{p \in \Delta_X} \max_{q \in \Delta_Y} u(p, q) = \max_{q \in \Delta_Y} \min_{p \in \Delta_X} u(p, q).$$

Thus to solve a zero-sum game, both players need to find strategies preparing them best for the worst opponent's strategy. The value $v_\mathcal{G} = u(p^*, q^*)$ is called the **value of the game**.

Example 3. Consider a planning task Π inducing the LTS in the Figure 1 with an initial state s_0 and a goal state s_1. There are three labels $\{l_1, l_2, l_3\}$ with respective costs $1, 5, 53$. The set of plans $\mathcal{P}(\Pi)$ consists of three plans of length 1. We define a game over Π. Player 1 is supposed to choose a plan getting her to the goal state. Player 2 (a robber) is waiting for Player 1 either along the transition $s_0 \xrightarrow{l_1} s_1$ or $s_0 \xrightarrow{l_2} s_1$. If Player 1 encounters Player 2, Player 2 steals her 100 cost units. Formally, we have a game $\mathcal{G} = \langle \{l_1, l_2, l_3\}, \{l_1, l_2\}, u_1, u_2 \rangle$ where $u_2(l_i, l_j) = 100$ if $i = j$ and 0 otherwise, and $u_1(l_i, l_j) = -c(l_i) - u_2(l_i, l_j)$. Player 1 seeks a mixed strategy over plans that would maximize her expected utility (equivalently minimize her expected costs) provided that Player 2 behaves rationally maximizing his utility.

Note that the game is "almost" zero-sum as the sum $u_1(l_i, l_j) + u_2(l_i, l_j) = -c(l_i)$ does not depend on Player 2's strategy. In this kind of game, we are interested in finding the optimal strategy for Player 1, which will consist of a mixed strategy over the possible plans to reach the goal.

Definition 4. We call a game $\mathcal{G} = \langle X, Y, u_1, u_2 \rangle$ **almost zero-sum** if $u_1(x, y) + u_2(x, y) = f(x)$ for some function $f \colon X \to \mathbb{R}$.

To each almost zero-sum game $\mathcal{G} = \langle X, Y, u_1, u_2 \rangle$, we associate its equivalent zero-sum game $\mathcal{G}_0 = \langle X, Y, u \rangle$ where $u(x, y) = -u_1(x, y) = u_2(x, y) - f(x)$. An almost zero-sum game $\mathcal{G}$ is best-response equivalent to its associated zero-sum game $\mathcal{G}_0$; see (Dritsoula, Loiseau, and Musacchio 2017). Thus they have the same NEs as is stated in the next theorem.

Theorem 5. *Let $\mathcal{G} = \langle X, Y, u_1, u_2\rangle$ be an almost zero-sum game such that $u_1(x,y) + u_2(x,y) = f(x)$ and $\mathcal{G}_0 = \langle X, Y, u\rangle$ its associated zero-sum game where $u(x,y) = -u_1(x,y) = u_2(x,y) - f(x)$. Then $\langle p^*, q^*\rangle$ is a NE of $\mathcal{G}$ iff it is a NE of $\mathcal{G}_0$.*

Proof. Note that $u_1(p,q^*) \leq u_1(p^*,q^*)$ iff $u(p^*,q^*) = -u_1(p^*,q^*) \leq -u_1(p,q^*) = u(p,q^*)$. Similarly, we have $u_2(p^*,q) \leq u_2(p^*,q^*)$ iff $u(p^*,q) = u_2(p^*,q) - f(p^*) \leq u_2(p^*,q^*) - f(p^*) = u(p^*,q^*)$. □

Thus to solve an almost zero-sum game, it suffices to find a NE $\langle p^*, q^*\rangle$ of its associated zero-sum game. Moreover, Player 1's utility $u_1(p^*,q^*) = -v_{\mathcal{G}_0}$ is just the opposite of the value of $\mathcal{G}_0$ that is given by the minimax strategy $\min_{p\in\Delta_X}\max_{q\in\Delta_Y} u(p,q)$. Consequently, Player 1's utility $u_1(p^*,q^*)$ is always the same in any NE $\langle p^*, q^*\rangle$. Thus Player 1 is indifferent on which equilibrial strategy to play. Nevertheless, the Player 2's utility may differ in different NEs because $u_2(p^*,q^*) = v_{\mathcal{G}_0} + f(p^*)$, i.e., Player 1 can influence Player 2's utility.

Example 6. Consider the almost zero-sum game $\mathcal{G}$ from Example 3. Its associated zero-sum game is $\mathcal{G}_0 = \langle\{l_1,l_2,l_3\},\{l_1,l_2\},u\rangle$ where $u(l_i,l_j) = c(l_i) + u_2(l_i,l_j)$. The game has several NEs. At first, Player 1 plays either l_1 with probability $1/2$ or l_2 with probability $1/2$. Player 2 plays l_1 with probability 0.52 or l_2 with probability 0.48. The expected utility for Player 2 in this case is 50. The expected Player 1's utility is -53. In the remaining equilibria, Player 1 plays always l_3 and Player 2's strategy is arbitrary. The expected Player 1's utility is again -53 but Player 2's utility is 0 now. So Player 1 is indifferent which NE to choose but her choice influences Player 2's utility. Still, note that the optimal utility of Player 1's is well defined, so we can compute an optimal strategy for Player 1.

In practical applications, we often look for an approximation of NE. We call a pair of mixed strategies $\langle p', q'\rangle$ an ϵ-NE if both players can improve their utilities at most by ϵ when playing p', q'.

Definition 7. Let $G = \langle X, Y, u_1, u_2\rangle$ be a game and $\epsilon > 0$. A pair of mixed strategies $\langle p', q'\rangle \in \Delta$ is called an **ϵ-Nash equilibrium** (ϵ-NE) if for all $\langle p, q\rangle \in \Delta$ we have

$$u_1(p,q') \leq u_1(p',q') + \epsilon, \quad u_2(p',q) \leq u_2(p',q') + \epsilon. \quad (2)$$

The advantage of ϵ-NE is that it is guaranteed that there exists a NE with "small" supports; see (Lipton, Markakis, and Mehta 2003). Even if the supports of NE might have size of X or Y, there is an ϵ-NE whose size of supports is bounded by $\frac{12\ln n}{\epsilon^2}(u_1^{max} - u_1^{min})(u_2^{max} - u_2^{min})$ where $n = \max\{|X|,|Y|\}$, $u_1^{max} = \max_{x\in X, y\in Y} u_1(x,y)$, $u_1^{min} = \min_{x\in X, y\in Y} u_1(x,y)$ and analogously for u_2^{max} and u_2^{min}.

Double Oracle

If we want to compute a NE of a zero-sum game $\mathcal{G} = \langle X, Y, u\rangle$ and we know its utility function u on $X \times Y$, we can find it by linear programming in polynomial time in the size of the representation of u; see e.g. (Shoham and Leyton-Brown 2009, Section 4.1). However, in this paper we deal with situations when X or Y might be very large or it might be difficult to compute the utility function for all possible pure strategies.

Algorithm 1 Double Oracle Algorithm

Input: Zero-sum game $G = \langle X, Y, u\rangle$, nonempty finite subsets $X_1 \subseteq X, Y_1 \subseteq Y$, and $\epsilon \geq 0$

Let $i := 0$
repeat
  $i := i + 1$
  Find NE (p_i^*, q_i^*) of subgame $\langle X_i, Y_i, u\rangle$
  Find some $x_{i+1} \in \mathrm{br}(q_i^*)$ and $y_{i+1} \in \mathrm{br}(p_i^*)$
  Let $X_{i+1} := X_i \cup \{x_{i+1}\}$ and $Y_{i+1} := Y_i \cup \{y_{i+1}\}$
  Let $\underline{v}_i := u(x_{i+1}, q_i^*)$ and $\overline{v}_i := u(p_i^*, y_{i+1})$
until $\overline{v}_i - \underline{v}_i \leq \epsilon$

Output: ϵ-NE (p_i^*, q_i^*) of G

To overcome this difficulty, we recall the Double Oracle Algorithm (DO) introduced in (McMahan, Gordon, and Blum 2003) that can compute a NE or ϵ-NE if it is stopped before the final iteration. It iteratively computes a NE of a subgame without evaluating the utility function in all points. When $X' \subseteq X$ and $Y' \subseteq Y$ are nonempty subsets, we define the **subgame** $G' = \langle X', Y', u\rangle$ of $\mathcal{G}$ by restriction of u to $X' \times Y'$ (denoted by the same letter).

The subgames in particular iterations of DO are constructed from best responses. Given a mixed strategy $q \in \Delta_Y$, the **best response set** for Player 1 is defined as $\mathrm{br}(q) = \{x \in X \mid u(x,q) = \min_{x'\in X} u(x',q)\}$. Analogously, for $p \in \Delta_X$ the best response set for Player 2 is: $\mathrm{br}(p) = \{y \in Y \mid u(p,y) = \max_{y'\in Y} u(p,y')\}$.

The pseudocode of the DO algorithm is listed in Algorithm 1. The algorithm starts with the sets X_1 and Y_1 of initial pure strategies. Typically, X_1 and Y_1 are singletons. Next, these sets are iteratively enlarged by best responses and the resulting subgame is solved by an LP solver. The crucial observation regarding the convergence of DO is the fact that a best-response for Player 1 gives a lower bound on the game value $\underline{v}_i$ whereas a best response for Player 2 gives an upper bound on the game value $\overline{v}_i$. Consequently, if these bounds are ϵ-close in an iteration i, the NE of the subgame $\langle X_i, Y_i, u\rangle$ is an ϵ-NE of $\mathcal{G}$. Further, note that DO is not a deterministic algorithm as the best responses are not unique.

Cost-Adversarial Planning Games

Now we are ready to formally define cost-adversarial planning games. Let Π be a planning task inducing an LTS Θ_Π with a cost function c. We define a game over Π as an almost zero-sum game where Player 1's pure strategies are plans in $\mathcal{P}(\Pi)$ and Player 2's pure strategies are cost functions from a given finite set of cost functions $\mathcal{C}$. For a plan $\pi \in \mathcal{P}(\Pi)$ and a cost function $g \in \mathcal{C}$, Player 2's utility $u_2(\pi, g)$ is just the cost of π w.r.t. g. Player 1's utility $u_1(\pi, g)$ is the opposite of the cost of π w.r.t. the cost function $c + g$. The cost function c defines base action costs and g represents the adversary modification of the action costs.

Definition 8. Let Π be a planning task and $\Theta_\Pi = \langle \mathcal{S}, \mathcal{L}, \mathcal{T}, s_I, S_\star \rangle$ its underlying LTS with a cost function c. Further, let $\mathcal{C} \subseteq (\mathbb{R}_0^+)^{\mathcal{L}}$ be a finite set of cost functions. A **cost adversarial planning game** (CAPG) is an almost zero-sum game $\mathcal{G} = \langle \mathcal{P}(\Pi), \mathcal{C}, u_1, u_2 \rangle$ where $u_1(\pi, g) = -c(\pi) - g(\pi)$ and $u_2(\pi, g) = g(\pi)$ for $\pi \in \mathcal{P}(\Pi)$ and $g \in \mathcal{C}$.

We call Player 1 in $\mathcal{G}$ **planning player** (shortly **P-player**) and Player 2 is called **cost player** (shortly **C-player**).

Note that the set of plans $\mathcal{P}(\Pi)$ might be in fact infinite, whenever the underlying LTS contains cycles. However, only finitely many of them are actually relevant. To see that note that the cost of a plan depends only on the multiset of its actions. Thus the relevant plans are those with minimal multisets of actions. There can be at most finitely many minimal multisets as follows from Dickson's lemma (Dickson 1913). Consequently, there can only be finitely many relevant plans because for each multiset of actions, there is only a finite number of plans corresponding to this multiset.

On the other hand, the number of relevant plans can be exponential in the size of the planning task Π. To see that, let us consider an $n \times n$-grid of locations from the visitall domain. Suppose we look for the shortest paths starting in the bottom-left corner leading to the top-right corner. In this case, the number of shortest plans is exponential in n because in each of the first n moves we can go either up or right. This explains the necessity of applying DO to solve CAPGs instead of solving them directly by precomputing a set of relevant plans and using an LP solver.

CAPGs can be solved by solving its associated zero-sum game $\mathcal{G}_0 = \langle \mathcal{P}(\Pi), \mathcal{C}, u \rangle$ where $u(\pi, g) = -u_1(\pi, g) = c(\pi) + g(\pi)$. Thus $\mathcal{G}_0$ can be solved by DO. To do so, we need to be able to compute a best response of any player to a mixed strategy of her opponent (Line 5 in Algorithm 1). First, consider a plan $\pi = \langle l_1, \dots, l_n \rangle$ and a mixed strategy $q \in \Delta_\mathcal{C}$. The utility $u(\pi, q)$ is the expected cost of π w.r.t. q, i.e., $u(\pi, q) = \sum_{g \in \mathrm{spt}(q)} q(g)(c(\pi) + g(\pi))$. Thus $u(\pi, q)$ is the cost of π w.r.t. the cost function assigning to $l \in \mathcal{L}$ the cost $c(l) + \sum_{g \in \mathrm{spt}(q)} q(g) g(l)$. Consequently, to find a best response of the planning player to q, we must find a plan π such that $u(\pi, q) = \min_{\pi' \in \mathcal{P}(\Pi)} u(\pi', q)$. Such plan π is nothing else than an optimal plan w.r.t. the above cost function $c + \sum_{g \in \mathrm{spt}(q)} q(g) g$ that can be found by any optimal planner.

To compute a best response of the cost player to a mixed strategy $p \in \Delta_{\mathcal{P}(\Theta)}$, we must find a cost function g such that $u(p, g) = \max_{g' \in \mathcal{C}} u(p, g')$. We have $u(p, g') = \sum_{\pi \in \mathrm{spt}(p)} p(\pi)(c(\pi) + g'(\pi))$. Thus it suffices to compute $g = \arg\max_{g' \in \mathcal{C}} \sum_{\pi \in \mathrm{spt}(p)} p(\pi) g'(\pi)$. To compute the above cost function g, we simply iterate through all the costs functions from $\mathcal{C}$.

Even though the set $\mathcal{C}$ is an arbitrary finite set of cost functions, some of its elements could be redundant because they can be dominated by other mixed strategies. Note that the elements of $\Delta_\mathcal{C}$ can be identified with the convex combinations of elements from $\mathcal{C}$, i.e., $\Delta_\mathcal{C}$ is the convex hull of $\mathcal{C}$. Thus we can restrict w.l.o.g. the set $\mathcal{C}$ to the extremal points of $\Delta_\mathcal{C}$. Moreover, as the cost player strives to maximize her utility, it suffices to consider only the extremal points which are maximal w.r.t. the pointwise order.

Case Studies

Cost-adversarial planning games provide quite a flexible framework to model antagonistic situations. In this section, we discuss particular applications of our framework. We modelled all underlying planning tasks in PDDL (McDermott et al. 1998). For our purposes, we extended the PDDL format a bit to be able to specify the cost player's pure strategies. In general, to capture CAPGs in PDDL, one has to encode several cost functions instead of a single one. To simplify the model, we decided that the C-player modifies only the costs of action instances of a single action schema. Consequently, our extended PDDL specifies only multiple action costs for the instances of that action schema.

Patrolling Security Games

The first application is inspired by security games (Tambe 2012). Security games are usually modelled as Stackelberg games. A 2-player Stackelberg game is specified by the same data as a 2-player normal-form game. The difference is in what players know about their opponent's mixed strategy. In the normal-form game, the players have no knowledge. On the other hand, in Stackelberg games, one player is a leader and the other one a follower. The leader has to announce her mixed strategy in advance. The follower chooses her strategy afterwards. A solution for a Stackelberg game is a leader's mixed strategy maximizing her utility provided that the follower always plays her best response; see (Conitzer and Sandholm 2006).

Thus each CAPG (and in fact each 2-player normal-form game) defines also a 2-player Stackelberg game. As each CAPG is an almost zero-sum game, we can relate its solution to the solution of the corresponding Stackelberg game. More precisely, the P-player's mixed strategy from NE is the solution for the Stackelberg game provided that P-player is the leader. This follows because the P-player's equilibrial strategy is the minimax strategy. Thus she can announce her strategy publicly without providing her opponent with an advantage.

Now we describe our first case study, based on patrolling security games (Yang et al. 2014; Shieh et al. 2012). Suppose there is a national park attacked by poachers. For simplicity we assume that there is a single poacher who regularly lays down a snare somewhere in the park. Locations with higher density of animals are more attractive for the poacher. On the other hand, we have a ranger who patrols in the park every day looking for the snare. However, the park is too large for the ranger to inspect each location in the park within the day. Our task is to find a probability distribution over a collection of circular paths of limited length starting and finishing at the ranger's base so that he minimises the expected costs for not discovering the snare and the travelled distance.

Formally, we model the park as a graph whose vertices represent particular locations in the park and edges are the road connections between them. Each road connection has its length. One of the vertices is the ranger's base. An example can be seen in Figure 2. The poacher's pure strategies

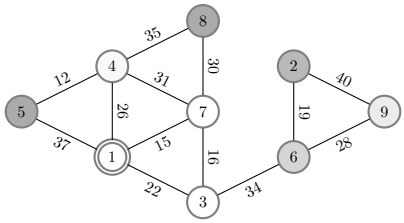
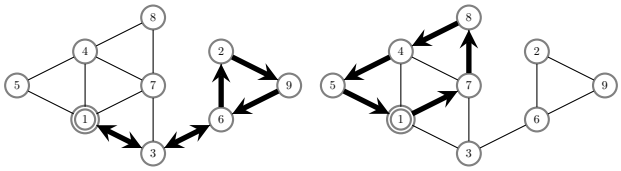
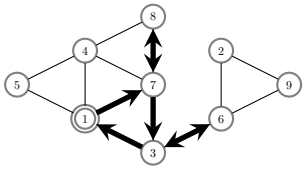


Figure 2: The graph representing the park. The double circled vertex denotes ranger's base. The equilibrial strategy for the poacher is depicted with the gray nodes. The respective probabilities of the locations 8, 5, 2, 6, 9, 4 are 0.287, 0.283, 0.276, 0.131, 0.016, 0.005.

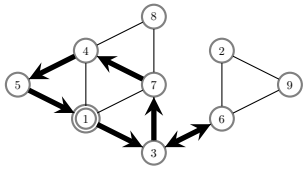


(a) Probability 0.416

(b) Probability 0.25



(c) Probability 0.166



(d) Probability 0.166

Figure 3: The equilibrial strategy for the P-player for the second scenario.

are vertices where he can put the snare. If the ranger visits that location during his patrol, he eliminates the snare so that the poacher's utility u_2 is 0. On the other hand, if the ranger misses the location, the poacher's utility is proportional to the density of animals in that location. The ranger is able to do at most k many moves during his patrol. Thus the ranger's pure strategies are paths starting and finishing in the ranger's base of length at most k. Ranger's cost (i.e., $-u_1$) is the travelled distance if he eliminates the snare. If he misses the snare, his cost is increased by the poacher's utility.

Example 9. Consider the graph in Figure 2. It represents locations in the park and their road connections with their distances. The ranger is able to make at most seven moves between the locations during a patrol. If the poacher traps an animal, his utility is 10000 and 0 otherwise. The ranger's cost is the sum of travelled distance plus the poacher's utility. We will discuss solutions to this game in two scenarios.

Firstly, we assume that the poacher always traps an animal obtaining the value of 10000 if his snare is not eliminated by the ranger. As the value 10000 is much greater than the distances in the graph, the optimal mixed strategy for the poacher is to put the snare into the most distant locations from the base (i.e., locations 8, 9). The precise probabilities on the locations 9, 8, 2, 5, 4 are respectively 0.5, 0.492, 0.003, 0.002, 0.001. On the other hand, the ranger is uniformly choosing among the paths (a) and (b) in Figure 3. Note that the paths consist of at most seven moves and cover altogether all the locations. The value of the associated zero-sum game (i.e., the ranger's cost) is 5164. The poacher's utility is 5000. Thus the ranger saves every second animal on average.

Secondly, we consider a more realistic scenario, whose equilibrial strategies are shown in Figures 2 and 3. Suppose that the poacher, if his snare is not eliminated, traps an animal at 30% of cases in all locations except the location 6 where he is successful in 70% of cases. So the poacher's utility, provided that the snare is not eliminated, in the location 6 is 7000 and 3000 in the remaining locations. Now the solution is not symmetric as before. The location 6 is attractive for the poacher so it pays off for him to risk putting the snare into a closer location. At the same time, the poacher should consider the locations 8 and 5 with higher probabilities 0.287 and 0.283. By this, the poacher forces the ranger occasionally to visit the left side of the graph while leaving the attractive location 6 unvisited. On the other hand, the ranger tends to visit the location 6 often to balance its attractiveness for the poacher. The value of the associated zero-sum game is 1926. The poacher's utility is 1633. Thus the poacher traps an animal roughly in 16% of cases.

We model this security game as a CAPG. The underlying planning task is formalized in PDDL. The graph of locations is encoded in the same way it is done in the IPC domain transport together with the `drive` action allowing the ranger to move in the graph. The objective is to find a route that finds as many locations where the poachers may have put some snare as possible, so we have a soft goal for each of those locations whose value depends on the poachers utility. We compile these soft goals away (Keyder and Geffner 2009), by forcing the plan to have two phases. The first one is the patrolling phase in which the ranger is moving through the graph by applying at most k `drive` actions. If he visits a location loc, unary predicate `visited`(loc) is set to true. Once he is back in the base, the planner can use an action `finish-patrol` switching the plan to the second phase. In this phase, the planner has to iterate through all locations in an encoded linear order. In each iteration, to get to the next location, the planner has to apply either `check-visited-location` or `check-unvisited-location`. The first action can be used only if the location was visited by the ranger and its cost is zero. The second one can only be applied if the location is not visited and its cost is determined by the poacher's mixed strategy, i.e., it is the product of the probability of catching an animal at that location and the probability of putting the snare into that location.

Finally, we discuss differences between our formulation of the patrolling game and those coming from the literature on security games (Yang et al. 2014; Shieh et al. 2012). They use a compact representation for the ranger's mixed strategies. Instead of a probability distribution over paths, they use a probability distribution over the locations expressing the probability that a given location will be covered by the ranger. After solving the game, the actual patrol paths are

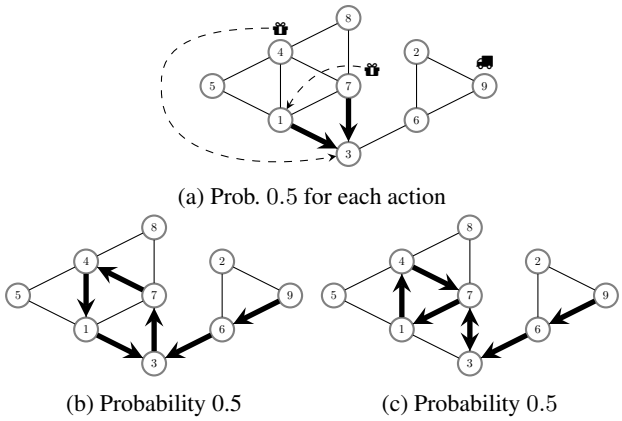
then generated by sampling from the set of all possible paths satisfying given constraints. As this set is very large, various pruning techniques are applied. On the other hand, our approach works directly with paths (and in general with plans). Furthermore, as our solution uses domain-independent planning modelling the underlying planning task in PDDL, we can easily encode other activities the ranger has to do during the patrol. As we use the DO algorithm, we do not have to use the compact representation and sampling because the optimal planner in each iteration of DO picks up most "relevant" plans. Further, we assume a perfectly rational poacher whereas the model of (Yang et al. 2014; Shieh et al. 2012) limited human rationality using the quantal response model (McKelvey and Palfrey 1995) and the data collected with human subjects to set the model parameters.

IPC Domains

Every IPC planning task can be turned into a CAPG if we specify a set of pure strategies $\mathcal{C}$ for the C-player. In some cases the solution of the resulting CAPG can be trivial. This happens for instance if the P-player must apply an obligatory action within her plan and the C-player may increase its cost. Consider for instance the BlocksWorld domain. Suppose the P-player has to build a tower having a block b on a block a. If the C-player can increase the cost of the action stack b on a, the P-player cannot avoid it. In such situations, the game has a pure NE, i.e., the players do not randomize at all.

To create more interesting CAPGs, one needs domains having sufficient variability in plans. In other words, the P-player must have several possibilities for how to achieve her goal to avoid the C-player's traps. Such a domain is, for instance, the transport domain as there are typically several paths to follow in order to achieve the goal. One way to define the C-player's pure strategies for a planning task in the transport domain is to let the C-player choose a set of the `drive` actions whose cost is increased. This approach has a natural interpretation that the C-player lurks along a limited number of roads to cause further expenses for the P-player if she takes one of these roads. The P-player needs to deliver all the packages to their destinations not knowing where the C-player is. The extra expenses need not be the same everywhere. They can for example reflect how suitable the road is for the attack or what load is being carried by the truck.

Example 10. Suppose we have a planning task from the transport domain whose map of locations is the same as in Figure 2. We have a single truck located at the location 9. There are two packages to be delivered. The first in the location 7 should be delivered to the location 1 and the second one from the location 4 to the location 3; see Figure 4(a). The C-player can choose and attack any `drive` action provided the truck is loaded. Thus it makes no sense to lurk along the edge between the locations 3 and 6. If the P-player applies the chosen drive action, she must pay 10000 penalty. The optimal strategy for the C-player attacks two `drive` actions $1 \to 3$ and $7 \to 3$ each with the probability 0.5; see Figure 4(a). This is reasonable as the P-player must deliver a package to the location 3. Moreover, the number of possibilities how to avoid the C-player's trap is the smallest one for the location 3 because the nodes corresponding to other locations with packages (i.e. 7, 4, 1) have degree 4; see Figure 4(a). On the other hand, the P-player has to randomize uniformly between two plans depicted in Figure 4(b–c).



(a) Prob. 0.5 for each action

(b) Probability 0.5

(c) Probability 0.5

Figure 4: The equilibrial strategies in Example 10 (a) for the C-player, (b–c) for the P-player.

Experiments

We evaluate the DO algorithm in several domains. First, in the patrol domain we introduced in the previous section. Particular instances of the patrol domain were generated based on the underlying maps of locations of the 30 instances of the 2008 optimal-track version of the IPC transport domain. For each map, we considered three variants limiting the number of moves the ranger can take at most to 6, 12, and 18, respectively.

Further, we created variants of several domains from the IPC, namely transport, data-network and visitall. For the transport domain, we considered two variants of the C-player's pure strategies. In the first one (we call it transport), the C-player can increase by a penalty the cost of a single `drive` action determined by two locations x, y and a truck t. So the penalty is applied only if the P-player applies exactly that `drive` action. If another truck t' drives from x to y, the P-player pays no penalty. The second variant (called transport-road) of the transport domain allows the C-player to increase simultaneously the cost of all `drive` actions from x to y no matter which truck applies it. The first variant clearly allows larger flexibility for the P-player to avoid the C-player's trap. In data network, the C-player can choose a server s and a script sc and increase simultaneously the cost of all actions processing the script sc on the server s no matter which data are processed. In visitall, we allow the C-player to choose among particular `move` actions where a penalty is applied. In all domains, the penalties increasing the base costs were generated randomly from the interval $[1000, 10000]$ for each pure strategy.

The core of DO was implemented in Python[1] and experimentally evaluated on a cluster of computing nodes with Intel Xeon Scalable Gold 6146 processors. For each task, we

[1] https://github.com/geoborek/CAPGs

domain	ipdb										lmcut						ms					
	NE	*cov*	*gmt*	*maxt*	*ml*	*tl*	*avgit*	*mit*	*avgP*	*avgC*	*NE*	*cov*	*gmt*	*maxt*	*ml*	*tl*	*NE*	*cov*	*gmt*	*maxt*	*ml*	*tl*
patrol	**85**	**90**	0.7	3.3	4	1	18.2	60	5.4	7.1	71	**90**	1.6	53.6	0	19	81	**90**	1.3	15.8	3	6
transport	**13**	**14**	0.7	7.2	17	0	11.8	21	5.2	7.2	9	11	1.3	982.4	0	21	11	11	0.8	20.5	19	0
transport-road	**13**	**14**	1.2	74.1	17	0	5.3	10	2.3	3.2	10	11	1.7	452.2	0	20	11	11	1.1	78.6	19	0
data-network	**3**	**12**	0.8	33.3	16	1	14.7	21	7.7	8.3	**3**	**12**	1.3	288.5	0	17	**3**	10	4.5	14.7	16	1
visitall[11]	**11**	**16**	0.5	13.7	9	0	17.5	34	9.1	13.3	9	10	0.5	4.2	0	11	9	9	0.9	5.3	11	0

Table 1: Overall results for all tested optimal planners. *NE:* the number of solved tasks; *cov:* the number of tasks where at least one plan was produced; *gmt:* the geometric mean of the best-response computation times for the commonly solved tasks; *maxt:* the maximum best-response computation time for the commonly solved tasks; *ml:* the number of tasks that failed due to the memory limit; *tl:* the number of tasks that failed due to the time limit; *avgit:* the average number of iterations for the tasks solved by ipdb; *mit:* the maximum number of iterations for the tasks solved by ipdb; *avgP:* the average size of the P-player's support for the tasks solved by ipdb; *avgC:* the average size of the C-player's support for the tasks solved by ipdb.

set a 30-minute time limit on the whole computation process and a 4 GB memory limit for the optimal planner. A vast majority of computational resources are consumed by the optimal planner. Our implementation of DO can be used with any optimal planner to compute the P-player's best response. In the experiments, we use Fast Downward (Helmert 2006) using A* with three different admissible heuristics: pattern databases (ipdb) (Haslum et al. 2007), LM-Cut (lmcut) (Helmert and Domshlak 2009) and merge-and-shrink (ms) with bisimulation based shrinking (Helmert et al. 2014) and the merge strategy SCC-DFP (Sievers, Wehrle, and Helmert 2016).

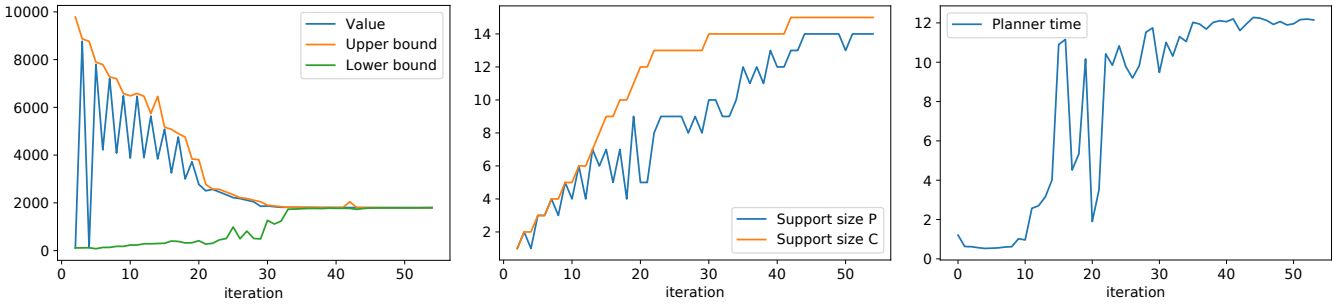
The overall results with all considered optimal planners are shown in Table 1. One can see that iPDB performed the best w.r.t. the "usual" coverage as shown in the column *cov,* meaning that the optimal planner was able to solve the first planning task in the initialization of DO, which corresponds to the original cost function disregarding the C-player. This translates into finding also the NE in most cases. Interestingly, the difference between *NE* and *cov* suggests that in most cases, if the underlying planner is able to find an optimal plan, the DO algorithm will terminate with a satisfactory NE too. Further, Table 1 shows the average, maximum numbers of iterations and the average size of the P-player's and the C-player's support. As these numbers are similar for all the considered planners, we present them only for (ipdb).

For the best performing optimal planner (ipdb), we present data on a typical DO execution for a single planning task in the patrol domain. In this task, the ranger was allowed to make up to 18 moves during his patrols. The map of locations having 21 locations is the same as in the task `p07.pddl` in the IPC-2008 version of the transport domain. The convergence of DO is shown in Figure 5. In the end, the NE is found after 54 iterations and the resulting mixed strategies have 14 plans and 15 cost functions, respectively. However, after each iteration the DO algorithm produces a candidate solution. The lower bounds computed by the planner in DO (i.e., the cost of the optimal plan found at each iteration) are small as long as there are good P-player's plans covering the C-player's mixed strategies considered so far. Once the C-player finds sufficiently many best responses w.r.t. previously found plans, the upper and lower bounds start to converge; see Figure 5(a–b).

Even though the optimal planner solves the exact same planning task only w.r.t. a different cost function, the times needed to compute an optimal plan increase as DO gets closer to the optimal strategies; see Figure 5(c). Similar behaviour can be observed in other tasks as well. The reason is that mixed strategies for the C-player induce more complex and diverse cost functions[2], which is known that they can negatively impact search performance (Wilt and Ruml 2011; Fan, Müller, and Holte 2017). Figure 6(a) shows a comparison between the considered planners in the best-response computation time for a single task (`problem06-half.pddl`) from the IPC-2011 visitall. The DO convergence for this task is depicted in Figure 6(b) showing how the difference between the upper and lower bounds decreases during the DO execution for all considered planners. Note that the actual curves are different due to non-uniqueness of best responses, namely finding different optimal plans may impact subsequent iterations of the algorithm.

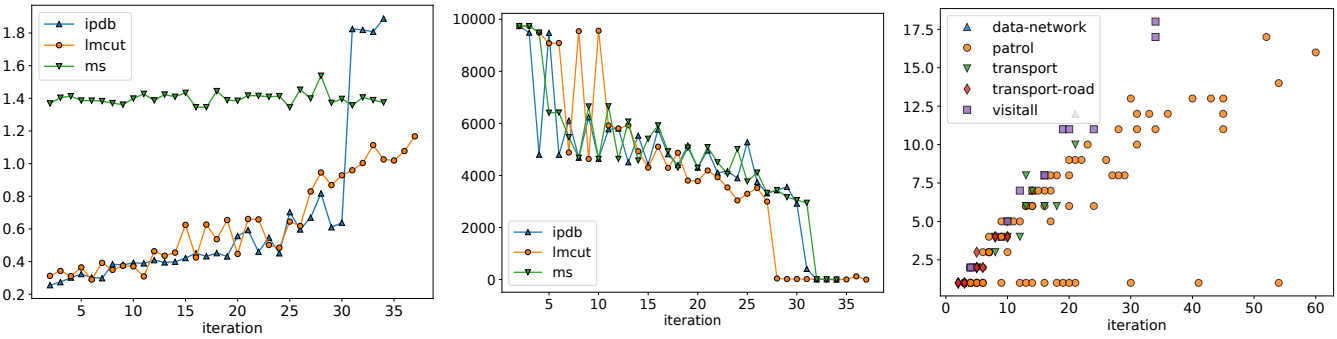
The final graph in Figure 6(c) displays a correlation between the number of DO iterations and the size of the P-Player's equilibrial support for all successfully computed tasks. It demonstrates that DO is a relatively efficient algorithm regarding its iterations. In each iteration at least one new pure strategy (best response) is computed and added to the subgame. It follows that many of the generated pure strategies are actually needed in the support of NE. Note that there are several degenerated cases with tens of iterations and support of size 1. These are the tasks from the patrol domain with a smaller map and a larger number of allowed moves. It takes several iterations to discover that there is a single plan covering the whole map giving the C-player no chance to trap an animal. The sizes of supports for the C-player behave similarly.

[2]Note that, even when the simple strategies only increase some action's cost by a constant, mixed strategies are linear combinations of those, potentially resulting on very diverse cost functions.



(a) DO convergence (b) Size of supports (c) Best-response computation time in sec.

Figure 5: The iterations of DO for a single planning task in the patrol domain. (a) The lower bound of the game value computed by the optimal planner; the upper bound of the game value computed by searching the best C-player's cost function; the value of the DO subgame (b) The size of supports of NEs for the DO subgame. (c) The time the optimal planner needed to compute the P-player's best response.



(a) Best response computation time in sec. (b) ϵ-convergence (c) Sizes of P-player's equilibrial supports

Figure 6: (a) The best response computation time for the planners ipdb, lmcut and ms on the problem `problem06-half.pddl` in the 2011 version of the IPC domain visitall. (b) The difference between the upper and lower bound for the planners ipdb, lmcut and ms on the same problem `problem06-half.pddl`. (c) The correlation between the number of iterations and the size of the P-player's equilibrial support for all tasks where a NE was found.

Related Work

Most of the literature on adversarial planning focus on multi-agent planning where several agents are planning in the same environment. To formalize agents interactions and conflicts of their plans is a non-trivial task. Bowling, Jensen, and Veloso (2003) proposed quite a complex definition of an equilibrium for multi-agent planning. A Nash equilibrium for multi-agent planning was considered in (Larbi, Konieczny, and Marquis 2007). Brafman et al. (2009 introduce coalition games into multiagent setting. Stackelberg game where a leader executes a plan followed by a plan of the follower were considered in (Speicher et al. 2018; Torralba et al. 2021). In this paper, we focus on a conceptually much easier setting where the adversary agent directly modifies the action costs. Despite its simplicity, it still formalizes a reasonable class of problems relevant for practical applications, e.g. in security games.

The DO algorithm was proposed in (McMahan, Gordon, and Blum 2003) to solve a certain class of zero-sum planning games over MDPs. Although these games substantially overlap with CAPGs, McMahan, Gordon, and Blum (2003) do not use any domain-independent formalism to model the planning component of their games. The DO algorithm in combination with domain-independent planners was also applied in (Rytíř, Chrpa, and Bošanský 2019; Chrpa, Rytíř, and Horčík 2020) focusing on resource collection games.

Conclusion

We introduced cost-adversarial planning games and showed which problems can be modelled by these games. In particular, we illustrated how to model patrolling security games within this formalism. Further, we showed how to solve them using DO together with the tools from optimal classical planning and experimentally evaluated our solution method.

As a byproduct, it turns out that the runtime of optimal planners could be substantially influenced by the chosen cost function. This suggests that CAPGs might serve as a reasonable benchmark to test optimal planners w.r.t. several cost functions, in particular given that many IPC domains used in optimal planning have unit costs.

Acknowledgements

This research was funded by AFOSR award FA9550-18-1-0097 and by the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

References

Bowling, M. H.; Jensen, R. M.; and Veloso, M. M. 2003. A Formalization of Equilibria for Multiagent Planning. In *Proc. IJCAI'03*, 1460–1462.

Brafman, R. I.; Domshlak, C.; Engel, Y.; and Tennenholtz, M. 2009. Planning Games. In *Proc. IJCAI'09*, 73–78.

Chrpa, L.; Rytíř, P.; and Horčík, R. 2020. Planning Against Adversary in Zero-Sum Games: Heuristics for Selecting and Ordering Critical Actions. In *Proc. SOCS'20*, 20–28.

Conitzer, V.; and Sandholm, T. 2006. Computing the optimal strategy to commit to. In Feigenbaum, J.; Chuang, J. C.; and Pennock, D. M., eds., *Proceedings 7th ACM Conference on Electronic Commerce (EC-2006), Ann Arbor, Michigan, USA, June 11-15, 2006*, 82–90. ACM.

Dickson, L. E. 1913. Finiteness of the Odd Perfect and Primitive Abundant Numbers with n Distinct Prime Factors. *American Journal of Mathematics*, 35: 413.

Dritsoula, L.; Loiseau, P.; and Musacchio, J. 2017. A Game-Theoretic Analysis of Adversarial Classification. *IEEE Trans. Inf. Forensics Secur.*, 12(12): 3094–3109.

Fan, G.; Müller, M.; and Holte, R. 2017. The Two-Edged Nature of Diverse Action Costs. In *Proc. ICAPS'17*, 98–106.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-Independent Construction of Pattern Database Heuristics for Cost-Optimal Planning. In *Proc. AAAI'07*, 1007–1012.

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

Helmert, M.; and Domshlak, C. 2009. Landmarks, Critical Paths and Abstractions: What's the Difference Anyway? In *Proc. ICAPS'09*, 162–169.

Helmert, M.; Haslum, P.; Hoffmann, J.; and Nissim, R. 2014. Merge & Shrink Abstraction: A Method for Generating Lower Bounds in Factored State Spaces. *Journal of the Association for Computing Machinery*, 61(3): 16.1–16.63.

Keyder, E.; and Geffner, H. 2009. Soft Goals Can Be Compiled Away. *Journal of Artificial Intelligence Research*, 36: 547–556.

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.

Lipton, R. J.; Markakis, E.; and Mehta, A. 2003. Playing large games using simple strategies. In Menascé, D. A.; and Nisan, N., eds., *Proceedings 4th ACM Conference on Electronic Commerce (EC-2003), San Diego, California, USA, June 9-12, 2003*, 36–41. ACM.

McDermott, D.; et al. 1998. *The PDDL Planning Domain Definition Language*. The AIPS-98 Planning Competition Comitee.

McKelvey, R. D.; and Palfrey, T. R. 1995. Quantal Response Equilibria for Normal Form Games. *Games and Economic Behavior*, 10(1): 6–38.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the Presence of Cost Functions Controlled by an Adversary. In Fawcett, T.; and Mishra, N., eds., *Machine Learning, Proceedings of the Twentieth International Conference (ICML 2003), August 21-24, 2003, Washington, DC, USA*, 536–543. AAAI Press.

Nash, J. 1951. Non-cooperative games. *Annals of Mathematics*, 54(2): 286–295.

Rytíř, P.; Chrpa, L.; and Bošanský, B. 2019. Using Classical Planning in Adversarial Problems. In *31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4–6, 2019*, 1335–1340. IEEE.

Shieh, E.; An, B.; Yang, R.; Tambe, M.; Baldwin, C.; DiRenzo, J.; Maule, B.; and Meyer, G. 2012. PROTECT: a deployed game theoretic system to protect the ports of the United States. In van der Hoek, W.; Padgham, L.; Conitzer, V.; and Winikoff, M., eds., *International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2012, Valencia, Spain, June 4-8, 2012 (3 Volumes)*, 13–20.

Shoham, Y.; and Leyton-Brown, K. 2009. *Multiagent Systems - Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press. ISBN 978-0-521-89943-7.

Sievers, S.; Wehrle, M.; and Helmert, M. 2016. An Analysis of Merge Strategies for Merge-and-Shrink Heuristics. In *Proc. ICAPS'16*, 294–298.

Speicher, P.; Steinmetz, M.; Backes, M.; Hoffmann, J.; and Künnemann, R. 2018. Stackelberg Planning: Towards Effective Leader-Follower State Space Search. In *Proc. AAAI'18*, 6286–6293.

Tambe, M. 2012. *Security and Game Theory - Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press. ISBN 978-1-10-709642-4.

Torralba, Á.; Speicher, P.; Künnemann, R.; Steinmetz, M.; and Hoffmann, J. 2021. Faster Stackelberg Planning via Symbolic Search and Information Sharing. In *Proc. AAAI'21*, 11998–12006.

von Neumann, J. 1928. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100: 295–320.

Wilt, C. M.; and Ruml, W. 2011. Cost-Based Heuristic Search Is Sensitive to the Ratio of Operator Costs. In *Proc. SOCS'11*.

Yang, R.; Ford, B. J.; Tambe, M.; and Lemieux, A. 2014. Adaptive resource allocation for wildlife protection against illegal poachers. In Bazzan, A. L. C.; Huhns, M. N.; Lomuscio, A.; and Scerri, P., eds., *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS '14, Paris, France, May 5-9, 2014*, 453–460. IFAAMAS/ACM.