Effective Planning in Resource-Competition Problems by Task Decomposition

Lukáš Chrpa, Pavel Rytíř, Andrii Nyporko, Rostislav Horčík, Stefan Edelkamp

Faculty of Electrical Engineering, Czech Technical University in Prague
{lukas.chrpa, pavel.rytir, andrii.nyporko, rostislav.horcik, stefan.edelkamp}@fel.cvut.cz

Abstract

Effective planning while competing for limited resources is crucial in many real-world applications such as on-demand transport companies competing for passengers. Planning techniques therefore have to take into account possible actions of an adversarial agent. Such a challenge that can be tackled by leveraging game-theoretical methods such as Double Oracle.

This paper aims at the scalability issues arising from combining planning techniques with Double Oracle. In particular, we propose an abstraction-based heuristic for deciding how resources will be collected (e.g. which car goes for which passenger and in which order) and we propose a method for decomposing planning tasks into smaller ones (e.g. generate plans for each car separately). Our empirical evaluation shows that our proposed approach considerably improves scalability compared to the state-of-the-art techniques.

Introduction

Presence of adversarial agents in an environment poses a challenge for planning techniques as plans have to take into account their possible actions (Applegate, Elsaesser, and Sanborn 1990). Succinct symbolic representations of state sets helped generating optimistic and strong cyclic adversarial plans (Jensen, Veloso, and Bowling 2001; Kissmann and Edelkamp 2009), however, at the cost of exploring most if not all alternatives like in minimax or FOND planning (Cimatti et al. 2003). Monte-Carlo Tree Search (MCTS) techniques have been used in zero-sum games (Lelis 2020), general game playing (Björnsson and Finnsson 2009), or the well known Starcraft real-time strategy game (Justesen and Risi 2017). Deep Reinforcement Learning (DRL) has shown impressive performance in numerous adversarial domains such as (again) Starcraft (Vinyals et al. 2019) or Go (Silver et al. 2018). MCTS and DRL work "online", i.e., they iteratively select and apply the most promising action until they reach the terminal state.

In contrast to MCTS and DRL, planning techniques generate plans upfront and these plans are then executed as they are, which might be necessary in cases in which plans cannot be easily amended (e.g. due to lack of communication between units). For example, *Plan Interdiction Games* leverage planning to reason about attackers and defenders in computer networks (Vorobeychik and Pritchard 2020). From the domain-independent standpoint, planning has been used in *Stackelberg games* for generating follower's plans against the adversarial leader (Speicher et al. 2018; Torralba et al. 2021), or for *counter-planning*, where an agent tries to invalidate landmarks required by the adversary, so the adversary cannot achieve its goal (Pozanco et al. 2018). Recently, a subclass of *normal-form games* has been tackled from the domain-independent planning perspective (Rytíř, Chrpa, and Bošanský 2019; Chrpa, Rytíř, and Horčík 2020). In contrast to Stackelberg games, in normal-form games, players do not necessarily follow the leader-follower scheme and can be in "equal" positions while selecting their strategies they will play. In contrast to counter-planning, normal-form games aim at achieving agent's goals rather than preventing the adversary from achieving its goals, which is often (but not necessarily) a "byproduct".

Resource-Competition Problems, which are the target of this paper, are a subclass of normal-form games where agents compete for limited resources. For example, suppose a scenario in which two on-demand transport (or Taxi) companies compete for passengers. When one company picks up a passenger, the other company no longer can serve that passenger. Similarly, two agents might compete for resources in a strategy game such that when an agent collects a resource, the opponent can no longer collect the resource for itself. Successful collection of (some) resources is usually crucial for achieving the goals of the agent (e.g. serve as many passengers as possible). That means that plans of the agent have to be optimised for maximising the chance of collecting important resources against the adversarial (or competing) agent.

Rather than a single plan, the agent should generate a mixed strategy, i.e., a set of plans with probabilities for being selected and executed, as it is harder to exploit (LaValle 2006). To generate mixed strategies, we can leverage the game-theoretic Double-Oracle algorithm (McMahan, Gordon, and Blum 2003). Rytíř, Chrpa, and Bošanský (2019) incorporated domain-independent planning into Double Oracle, however, the results have shown a lack of scalability (even for suboptimal settings). We have identified two main reasons: (i) lack of guidance on means by which resources

are collected, and (ii) the centralised form of planning. Chrpa, Rytíř, and Horčík (2020) aimed at addressing (i) by introducing a delete-relaxation heuristic for estimating when resources can be collected and by what means, and leveraging it as a guidance during (centralised) planning. In contrast to Chrpa, Rytíř, and Horčík, we propose an abstraction-based heuristic, which is computationally cheaper, and fits into a more general theoretical framework. In this paper, we also address (ii) by specifying under which conditions we can decompose the problem into smaller subproblems that can be solved individually. Problem decomposition can be straightforwardly done for problems in which the agent controls a fleet of units with limited joint interaction. Then, we can plan for each unit individually (e.g. for each car) and these plans can be easily combined into a plan for the original problem. Our empirical evaluation shows that our proposed approach considerably improves scalability compared to the state-of-the-art techniques. (Rytíř, Chrpa, and Bošanský 2019; Chrpa, Rytíř, and Horčík 2020).

Preliminaries

This section introduces the terminology we use in this paper.

Automated Planning

In our setting, we consider classical planning with action duration and discrete timelines. Note that our model is simpler than PDDL 2.1 (Fox and Long 2003).

Let V be a set of ***variables*** where each variable $v \in V$ is associated with its domain $D(v)$. An ***assignment*** of a variable $v \in V$ is a pair (v, val), where its value $val \in D(v)$. Hereinafter, an assignment of a variable is also denoted as a ***fact***. A (partial) ***variable assignment*** p over V is a set of assignments of individual variables from V, where $vars(p)$ is a set of all variables in p and $p[v]$ represents a value of v in p. For a (partial) variable assignment p over V and a subset of variables $V' \subseteq V$, we define a variable assignment q over V' being a ***projection*** of p into V', denoted as $[p|V']$, as $q[v'] = p[v']$ for each $v' \in V'$ ($q[v']$ is undefined iff $p[v']$ is undefined). A ***state*** is a complete variable assignment (over V). To accommodate the notion of time, we denote that an assignment f or a (partial) variable assignment p holds in time t as $f(t)$ or $p(t)$ respectively. An ***action*** is a triple $a = (dur(a), pre(a), eff(a))$, where $dur(a)$ is a positive integer representing the duration of a's application, $pre(a)$ is a partial variable assignment representing the a's precondition and $eff(a)$ is a partial variable assignment representing the a's effects. The set of variables of a is denoted $vars(a) = vars(pre(a)) \cup vars(eff(a))$.

We say that actions a_i and a_j are ***independent*** iff $vars(a_i) \cap vars(a_j) = \emptyset$, otherwise we say that a_i and a_j ***interfere*** with each other. We say that an action a is ***applicable*** in a state s and time t if and only if $s(t) \models pre(a)$. The ***application interval*** of an action a in time t is $(t, t+dur(a))$.

A ***planning task*** is a quadruple $\mathcal{P} = (V, A, I, G)$, where V is a set of variables, A a set of actions, I a complete variable assignment representing the ***initial state*** and G a partial variable assignment representing the ***goal***. Let $\pi = \{(a_1, t_1), \ldots, (a_n, t_n)\}$ be a set of couples (action,time) such that for every a_i, a_j $(i \neq j)$ such that a_i and a_j interfere the application interval of a_i in t_i is disjoint with the application interval of a_j in t_j. The ***state trajectory*** of π with respect to a state I is a sequence of states $s(0) = I, s(1), \ldots$ such that for $t > 0$, $s(t)[v] = eff(a_i)[v]$ iff $t = t_i + dur(a_i)$ and $v \in vars(eff)(a_i))$, or $s(t)[v] = s(t-1)[v]$ otherwise. We say that π is a ***plan*** for $\mathcal{P}$ iff for its state trajectory w.r.t I it is the case that a_i is applicable in $s(t_i)$ for each $1 \leq i \leq n$ and $s(\max_{i=1}^{n}(t_i + dur(a_i))) \models G$.

Another variant of planning task definition considers, rather than a single (hard) goal, a set of ***soft goals*** (each goal is a set of variable assignments) such that failing to achieve a goal is penalised. Therefore, for a planning task $\mathcal{P} = (V, A, I, G)$, $G = \{G_1, \ldots, G_n\}$, where each G_i is associated with a cost M_i $(1 \leq i \leq n)$ such that for a plan π it is the case that its cost is $\sum_{i \in \{i \mid G_i \text{ not achieved}\}} M_i$.

Normal-Form Games

A ***normal-form game*** Γ is a tuple (N, Ω, u), where N is the number of players, $\Omega = \Omega_1 \times \cdots \times \Omega_N$, where $\Omega_1, \ldots, \Omega_N$ are finite sets of pure strategies of players $1, \ldots, N$ and $u = (u_1, \ldots, u_N)$ is an N-tuple of utility functions that assign a real-valued utility of player i for each outcome of the game defined by a strategy profile – an N-tuple of pure strategies (one for each player); $u_i : \Omega \to \mathbb{R}$. We say that a normal-form game is a ***zero-sum game*** if $\sum_{i=1}^{N} u_i(\omega) = 0$ for each $\omega \in \Omega$. From now, we focus only on 2-player games, i.e., $N = 2$.

A ***mixed strategy*** for a player i is a probability distribution σ_i over the set of player's pure strategies Ω_i. A pair of mixed strategies $\sigma = (\sigma_1, \sigma_2)$ is called a ***mixed-strategy profile***. We extend the definition of utility functions so that for a given mixed-strategy profile σ the value $u_i(\sigma)$ is the expected utility of player i. We say that a mixed strategy of one player σ_i is the ***best response*** to the strategy of the opponent σ_{-i} (denoted as $\sigma_i = br(\sigma_{-i})$) when $u_i(\sigma_i, \sigma_{-i}) \geq u_i(\sigma'_i, \sigma_{-i})$ for all mixed strategies σ'_i over Ω_i. We say that a mixed-strategy profile $\sigma = (\sigma_1, \sigma_2)$ is in ***Nash equilibrium (NE)*** if for each mixed-strategy profile $\sigma' = (\sigma'_1, \sigma'_2)$ it is the case that $u_1(\sigma_1, \sigma_2) \geq u_1(\sigma'_1, \sigma_2)$ and $u_2(\sigma_1, \sigma_2) \geq u_2(\sigma_1, \sigma'_2)$. A mixed-strategy profile in NE can be computed via Linear Programming.

Double Oracle

One way for tackling normal-form games is to incrementally build the game using the ***Double-Oracle*** algorithm (McMahan, Gordon, and Blum 2003) as summarised in Algorithm 1. The algorithm starts with a restricted game, where each player's NE mixed strategy is composed from a subset of pure strategies, usually containing only a single pure strategy (Lines 2–3). Then, iteratively each player computes the (best) response expanding the restricted game by (the best) pure strategy (Lines 5–6) and the NE mixed-strategy profile for the expanded restricted game is determined (Line 7). The algorithm terminates when neither of the players can add a (best) response strategy that improves the expected outcome from the restricted game (Line 8).

If best responses are computed in each iteration, the NE

Algorithm 1: Double Oracle Algorithm (for 2 Players)

Require: A 2-player normal-form game $((\Omega_1, \Omega_2), (u_1, u_2))$
Ensure: A mixed-strategy profile σ
1: Let $X' \subseteq \Omega_1$ and $Y' \subseteq \Omega_2$ (X and Y are arbitrary nonempty subsets of Ω_1 and Ω_2)
2: $\sigma^{X',Y'} = \mathrm{NE}((X', Y'), (u_1, u_2))$
3: **repeat**
4: $X \leftarrow X', Y \leftarrow Y', \sigma^{X,Y} \leftarrow \sigma^{X',Y'}$
5: $x = \arg\max_{x \in \Omega_1 \setminus X} u_1(x, \sigma_2^{X,Y})$
6: $y = \arg\max_{y \in \Omega_2 \setminus Y} u_1(\sigma_1^{X,Y}, y)$
7: $\sigma^{X',Y'} = \mathrm{NE}((X \cup \{x\}, Y \cup \{y\}), (u_1, u_2))$
8: **until** $u_1(\sigma_1^{X',Y'}, \sigma_2^{X,Y}) \leq u_1(\sigma^{X,Y})$ and $u_2(\sigma_1^{X,Y}, \sigma_2^{X',Y'}) \leq u_2(\sigma^{X,Y})$
9: **return** $\sigma^{X',Y'}$

of the restricted game matches the one in the original game, since the best responses are computed over the unrestricted set of all pure strategies (McMahan, Gordon, and Blum 2003). The algorithm returns an optimal strategy, but is not monotone (in the upper and lower bounds on the game value in each iteration), and might have to consider, in the worst case, all pure strategies during its computation.

In the planning setting, pure strategies are not known in advance. The (best) response computation accounts for generating an (optimal) plan against the current mixed strategy of the other player (Rytíř, Chrpa, and Bošanský 2019). Note that a plan is optimal if the expected cost of failing the goals is minimal (we elaborate on this aspect later in the paper).

Case Studies

We present two domains as our case studies.

Resource Hunting Domain

We consider a two-player game introduced by Rytíř, Chrpa, and Bošanský (2019), called *Resource Hunting*, where each player controls its fleet of unmanned aerial vehicles (UAVs) that tries to collect as many resources as possible. Each UAV can *move* from one location to another. Each UAV can carry at most two sensors. For each resource to be collected, one or two (different) sensors are required. One or two UAVs can *collect* an available resource if the UAV(s) are at the same location as the resource and carry the required sensors.

Taxi Domain

We consider an on-demand transport scenario in which there are two taxi companies competing for passengers who require to be transported from one location to another. When one company picks up a passenger, she/he can no longer be transported by the other company. The goal of each taxi company is to maximise its rewards by transporting passengers, at the expense of the competing taxi company. Each taxi company operates a fleet of cars. Each car can carry at most one passenger at time. The car can move between two connected locations by the *drive* action, can *load* a passenger into itself if both are at the same location, and can *unload* the passenger if being in his/her destination location.

Resource-Competition Planning Task

In adversarial settings, agents might interfere with each other while executing their plans. Such conflicts are usually inevitable as "winning" the conflict might be essential for achieving a certain (soft) goal. Picking up a passenger before the other agent does is a good example of "winning" the conflict.

In general, we can define a planning task for 2-player normal form games. Our definition is partially inspired by the MA-STRIPS formalism (Brafman and Domshlak 2008) used in Multi-agent planning. In contrast to MA-STRIPS, we consider soft goals for each agent and simple durative actions.

Definition 1. *Let $\mathcal{NP} = (V, A^1, A^2, I, G^1, G^2)$ be a* **2-Player Normal-form Game (2PNG) Planning Task**, *where V is a set of variables, A^1 and A^2 such that $A^1 \cap A^2 = \emptyset$ are sets of (durative) actions for the first and second agent respectively, I is an initial state and G^1 and G^2 are sets of soft goals for the first and second agent respectively.*

To address the 2PNG planning task, each agent might generate its own plan by solving an underlying planning task, where each agent uses only its actions to achieve its own goals. Plans of both agents are executed simultaneously and we will assume that plans of both agents start their execution at the same time. Even though plans of individual agents are fully applicable on their own, when both agents' plans are applied simultaneously, conflicts can arise (the agents compete against each other).

In our execution model, in which plans of both agents are applied simultaneously, actions which are inapplicable in their scheduled timestamp and actions which interfere with an already running action (of the other agent) in their scheduled timestamp are skipped and their effects thus do not take place. If interfering actions are scheduled at the same time, then one action is randomly selected by the "coin toss" (i.e., with an equal chance) to be applied while the other become inapplicable. After plans of both agents are applied, the cost of the plan for each agent is determined as the sum of costs of soft goals the agent failed to achieve. The utility value for each player is computed by subtracting the cost of the plan from the sum of the costs of all player's soft goals.

Facts, i.e., variable assignments, that are required by actions of one agent (or they are part of agent's goals) can be deleted by actions of the competing agent. We call such facts *conflicting*. Specific conflicting facts that are initially true but cannot be reachieved by either agent are called *critical facts* (the notion is adopted from Chrpa, Rytíř, and Horčík (2020)). Critical facts, in other words, can only be consumed and hence account for *limited resources* (e.g. waiting passengers).

Definition 2. *Let $\mathcal{NP} = (V, A^1, A^2, I, G^1, G^2)$ be a 2PNG planning task. We say that (v, val), where $v \in V$ and $val \in D(v)$, is a* **conflicting fact** *iff $G^i[v] = val$ or there exists $a \in$*

$A^i : \mathit{pre}(a)[v] = val$ *and there exists* $a' \in A^j : \mathit{eff}(a')[v] = val' \wedge val \neq val'$ *with* $1 \leq i, j \leq 2$ *and* $i \neq j$.

We say that a conflicting fact (v, val) *is a* **critical fact** *iff* $I[v] = val$ *and for each* $a \in A^1 \cup A^2 : \mathit{eff}(a)[v] \neq val$.

Resource Competition planning tasks, as defined below, are a subclass of 2PNG planning tasks in which all conflicting facts are critical facts. Conflicts between agents are hence limited to critical facts, that is, which agent manages to consume a particular critical fact earlier.

Definition 3. *Let* $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ *be a 2PNG planning task. We say that* $\mathcal{RP}$ *is a* **Resource Competition (RC) Planning Task** *iff each conflicting fact is a critical fact.*

Towards Solving RC Planning Tasks

RC planning tasks can be addressed by leveraging the game-theoretical Double Oracle together with (domain-independent) plan generation as initially proposed by Rytíř, Chrpa, and Bošanský (2019). Essentially, selecting the (best) pure strategy to respond the adversary current mixed strategy (and vice versa), i.e., Lines 5 and 6 of Algorithm 1, is associated with a plan generation episode that aims at finding a plan maximising the profit of the agent against the current mixed strategy of the adversary. We refer to such plans as *response plans* or *best response plans* if optimal. Note that in this setting pure strategies are plans and they are not known in advance (i.e., we do not know Ω_1 and Ω_2 up front).

How good the agent's plan is depends, in a nutshell, how it is successful in "conflicts" over critical facts with the adversary. In the Taxi example, the agent's goal is to deliver a passenger into her destination. In order to do so, the agent has to pick the passenger up at some point. However, the adversary might have the same goal and wants to pick up the passenger as well. The passenger who is waiting in her location of origin can therefore be understood as such a "conflict". To win the "conflict", the agent has to pick up the passenger before the adversary does.

In more general terms, the agent might need to apply certain *critical actions* requiring specific *critical facts* to be true (e.g. picking up a waiting passenger into a car). The adversary, on the other hand, can apply *adversary actions* that delete these critical facts making them no longer achievable (e.g. picking up the waiting passenger first). Hereinafter, we present the terminology from the perspective of agent 1, i.e., agent 2 is considered as the adversary. Note that we adopted the notions of critical and adversary actions from Chrpa, Rytíř, and Horčík (2020).

Definition 4. *Let* $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ *be a RC planning task and* (v, val) *be a critical fact. We say that* $\mathcal{A}^c(v, val) = \{a^c \in A^1 \mid \mathit{pre}(a^c)[v] = val\}$ *is a set of* **critical actions** *over* (v, val). *We also define a set of* **adversary actions** *over* (v, val) *as* $\mathcal{A}^a(v, val) = \{a^a \in A^2 \mid \mathit{eff}(a^a)[v] = val', val \neq val'\}$.

Conceptually, the agent needs to apply its critical actions before the adversary applies its adversary actions (as indicated in the above example). Therefore, adversary actions set deadlines for agent's critical actions. Note that the definition of a critical fact restricts on situations, where the fact is present in the initial state and when deleted it cannot be reachieved. Hence deadlines set by adversary actions are "strict", that is, missing the deadline means that critical actions over the affected critical fact will no longer be applicable.

Definition 5. *Let* $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ *be a RC planning task and* $\pi' = \{(a'_1, t'_1), \ldots, (a'_m, t'_m)\}$ *be a plan of the adversary. Let* F^c *be the set of critical facts and* $\mathcal{A}^c$ *be the set of all critical actions in* $\mathcal{RP}$ *(*$\mathcal{A}^c = \bigcup_{f \in F^c} \mathcal{A}^c(f)$*). Then, for each* $a^c \in \mathcal{A}^c$, *we determine its* **deadline** *w.r.t* π' *as* $dl(a^c, \pi') = \min\{t \mid (a', t) \in \pi', a' \in \mathcal{A}^a(f), a^c \in \mathcal{A}^c(f), f \in F^c\}$.

As indicated above, deadlines set by adversary actions determine whether corresponding critical actions can be successfully applied. Assuming that the adversary follows a mixed strategy (as in Double Oracle), that is, it selects a plan from its set of plans according to a given probability, meeting or missing the deadline is determined by the probability that the adversary selects a plan with later or earlier occurrence of the corresponding adversary actions. Note that our execution model gives a 50% chance of meeting the deadline if the critical and adversary actions are scheduled to be applied at the same timestamp.

Definition 6. *Let* $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ *be a RC planning task and* $\mathcal{A}^c$ *be the set of all critical actions. Let* $\sigma = \{(\pi'_1, p'_1), \ldots, (\pi'_n, p'_n)\}$ *be a mixed strategy of the adversary. For each* $a^c \in \mathcal{A}^c$, *we define a function* succ *representing the probability of* a^c *being successfully applied in time* t *w.r.t* σ *as follows:*

$$succ(a^c, t, \sigma) = \sum \{p' \mid t < dl(a^c, \pi'), (\pi', p') \in \sigma\} + 0.5 \cdot \sum \{p' \mid t = dl(a^c, \pi'), (\pi', p') \in \sigma\}$$

The $succ$ function can be leveraged for plan generation such that probability of successful application of critical actions is propagated to actions depending on them (achieving precondition atoms for them). Consequently, we can determine the probability of achieving particular (soft) goals and thus the utility or the cost of the plan (hereinafter denoted as $cost(\pi, \sigma)$ for agent's plan π and adversary's mixed strategy σ). An agent's plan with the minimum cost (or the maximum utility) with respect to the adversary mixed strategy σ is the agent's *best response plan* against σ (Line 5 of Algorithm 1) and is a part of agent's *best response* to σ (Line 7 of Algorithm 1).

Note that for a subclass of RC planning tasks in which at most one critical action is needed to achieve a particular soft goal, the probability of failure of critical actions can be directly reflected in their costs and hence cost-optimal planning can be leveraged for generating best response plans. For details, see (Rytíř, Chrpa, and Bošanský 2019).

Critical Action Selection

The quality of response plans depends on what critical actions and in which order they are applied in these plans. For example, in the Taxi domain, the agent has to decide which car of its fleet will serve what passenger and in which order. Such a decision, in consequence, affects the time when

each of the passengers will be picked up and whether the deadlines (for picking up the passengers) will be met or not.

Selecting and partially ordering critical actions prior response plan generation can speed up the planning process as the planner only needs to "fill the gaps" between the critical actions and goals (Chrpa, Rytíř, and Horčík 2020). Here, we formally conceptualise the idea by introducing the *Critical Action Selection (CAS)* structure. We have to take into consideration that interfering critical actions have to be ordered (as they cannot be applied simultaneously) and that a *sound* CAS structure allows a plan in which only the selected critical actions occur and they occur in the specified order.

Definition 7. *Let $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ be a RC planning task and $\mathcal{A}^c$ be the set of all critical actions. We say that a couple $(A^c, \prec)$ is a* **Critical Action Selection (CAS)** *for $\mathcal{RP}$ if $A^c \subseteq \mathcal{A}^c$, $\prec$ is a partial order relation over A^c such that for each $a_i, a_j \in A^c$ ($a_i \not\equiv a_j$) that interfere, it is the case that either $a_i \prec a_j$ or $a_j \prec a_i$. We also say that a plan π^1 for a planning task $\mathcal{P} = (V, A^1, I, G^1)$* **follows** *CAS $(A^c, \prec)$ if and only if π^1 contains all critical actions from A^c but none from $\mathcal{A}^c \setminus A^c$ and for each $a, a' \in A^c$ such that $a \prec a'$ it is the case that a' is applied later than a. If such π^1 exists, we say that $(A^c, \prec)$ is* **sound**.

It can be observed that critical actions that are *mutually exclusive*, that is, at most one of the actions can be present in a plan, cannot be both in a sound CAS. For instance, critical actions that consume the same critical fact (e.g. different cars picking up the same passenger) are mutually exclusive.

Estimating the Cost of CAS by Abstraction

Since a CAS forms a skeleton of a response plan, it is important to find a "good" CAS in order to be able to generate a good quality response plan. To estimate the cost of a response plan, we need to estimate the cost of the CAS the plan follows. To do so, we need to estimate application times of the critical actions in the CAS. In contrast to Chrpa, Rytíř, and Horčík (2020), who exploit a delete-relaxation heuristic for estimating application times of actions, we present a method based on abstraction. The main advantage of the abstraction-based approach is that we need to take into consideration only "distances" between critical actions (i.e., how long after the former action finishes it might take before the latter action becomes applicable). On top of that, the "distances" have to be computed only once and reused whenever needed.

To abstract the state space, we leverage the notion of Domain Transition Graph (Jonsson and Bäckström 1998), which is an atomic projection to a single variable, that we adapt for our temporal settings.

Definition 8. *Let $\mathcal{P} = (V, A, I, G)$ be a planning task. For a variable $v \in V$, we define a* **Temporal Domain Transition Graph** *(T-DTG) as a weighted labelled directed graph $G_v = (D(v), T_v)$, where $D(v)$ is a set of vertices, T_v is a set of weighted labelled edges. For all $x, y \in D(v)$ and $a \in A$, $(x, a, y)^w \in T_v$ iff $\mathit{eff}(a)[v] = y$ and either $\mathit{pre}(a)[v] = x$ or $v \notin vars(\mathit{pre}(a))$ with $w = 0$ iff $v \notin vars(\mathit{pre}(a))$, or $w = dur(a)$ otherwise.*

For each variable $v \in V$, we define $d_v : D(v) \times D(v) \to \mathbb{N}_0^+$ such that $d_v(x, y)$ represents the value of the shortest path from x to y in the T-DTG G_v, measured by the sum of weights of the edges in the path.

Furthermore (considering $\mathcal{P} = (V, A, I, G)$), we define ha, $dist$ and $time$ functions. Let $ha : A \times V \to \bigcup_{v \in V} D(v)$ be a function which returns the value of the variable $v \in vars(a)$ that will hold after a finishes its application:

$$ha(a, v) = \mathit{eff}(a)[v] \text{ iff } v \in vars(\mathit{eff}(a))$$
$$ha(a, v) = \mathit{pre}(a)[v] \text{ iff } v \in (vars(\mathit{pre}(a)) \setminus vars(\mathit{eff}(a)))$$

Inspired by the concept of h^m heuristic (Haslum 2009), we define a function $dist : A \cup \{\bot\} \times A \to \mathbb{N}_0^+$ representing an estimated distance between actions, that is, after how long since one action finishes the other action can start (note that for independent actions the distance is 0), or an estimated distance between an initial state and an action, that is, after how long the action can start:

$$dist(\bot, a) = \max\{d_v(I[v], \mathit{pre}(a)[v]) \mid v \in vars(\mathit{pre}(a))\}$$
$$dist(a_i, a_j) = \max(\{0\} \cup \{d_v(ha(a_i, v), \mathit{pre}(a_j)[v]) \mid v \in vars(a_i) \cap vars(\mathit{pre}(a_j))\})$$

Together with the notion of distance between two actions, we can estimate when each critical action in a given CAS can be applied. For this purpose, we define a function $time^{(A^c, \prec)} : A^c \to \mathbb{N}_0^+$ over CAS $(A^c, \prec)$ as follows:

$$time^{(A^c, \prec)}(a) = dist(\bot, a) \text{ iff } \nexists a' \in A^c : a' \prec a$$
$$time^{(A^c, \prec)}(a) = \max\{time^{(A^c, \prec)}(a') + dur(a') + dist(a', a) \mid a' \in A^c, a' \prec a\}$$

The claim of the following proposition follows from an observation that d_v provides a lower bound on the time needed to change one value of v to another and since $dist$ considers the most "expensive" value change among the relevant variables, $time$ thus provides a lower bound of time of critical action applicability.

Proposition 9. *Let $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ be a RC planning task and $(A^c, \prec)$ be a CAS for $\mathcal{RP}$. For each plan π^1 for a planning task $\mathcal{P} = (V, A^1, I, G^1)$ that follows $(A^c, \prec)$, it is the case that for each $a \in A^c$ it holds that $time^{(A^c, \prec)}(a) \leq t$ where $(a, t) \in \pi^1$.*

With the time estimates for critical actions from a given CAS $(A^c, \prec)$ we can estimate their chance for being successfully applied with respect to a given adversary's mixed strategy σ as in Definition 6. For estimating the probability of achievement for each of the soft goals, we leverage the notion of a *disjunctive action landmark* (Hoffmann, Porteous, and Sebastia 2004), a set of actions such that every plan must contain at least one of the actions. We denote a disjunctive action landmark as *minimal* if none of its proper subsets is a disjunctive action landmark on its own. Also, we consider only disjunctive action landmarks in which all actions are critical actions. Let $\mathcal{L}_i^c$ denote a set of minimal disjunctive action landmarks for a goal G_i that contain only critical

Algorithm 2: Optimising CAS by simulated annealing

Require: CAS X, Adversary strategy σ, Initial temperature t, Constant K and Temperature decrease ϵ.
Ensure: Optimised CAS X.

while $t > 0$ **do**
  $X' \leftarrow$ SAMPLENEXTCAS(X)
  $\Delta \leftarrow cost(X', \sigma) - cost(X, \sigma)$
  if $\Delta < 0$ **then**
    $X \leftarrow X'$
  else
    $X \leftarrow X'$ with probability $\exp(-\Delta/(t * K))$
  end if
  $t \leftarrow (t - \epsilon)$
end while
return X

actions. Then, the probability of achieving G_i, is estimated as follows:

$$ach(G_i, (A^c, \prec), \sigma) = \prod_{L_i^c \in \mathcal{L}_i^c} \max(\{0\} \cup \{succ(a, time^{(A^c,\prec)}(a), \sigma) \mid a \in L_i^c\})$$

The success rate for each landmark is a maximum of the probabilities of its critical actions being successfully applied in $(A^c, \prec)$, or 0 if none of its critical actions is in $(A^c, \prec)$. If sets of actions in $\mathcal{L}_i^c$ are not pairwise disjoint (i.e., some disjunctive action landmarks overlap), we can modify the expression by replacing "$\prod$" by "$\min$".

It can be observed that the ach function overestimates the probability of achieving the given goal as it optimistically assumes that failure of some critical action does not affect applicability of subsequent critical actions (so they might become inapplicable despite meeting their deadlines).

Let $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ be a RC planning task, where $G^1 = \{G_1, \ldots, G_n\}$ is a set of soft goals, each G_i is associated with a cost of failure M_i, and $(A^c, \prec)$ be a CAS for $\mathcal{RP}$. Then, the cost of $(A^c, \prec)$ with respect to a given adversary's mixed strategy σ is calculated as follows:

$$cost((A^c, \prec), \sigma) = \sum_{i=1}^{n} M_i(1 - ach(G_i, (A^c, \prec), \sigma))$$

From Proposition 9, we can derive that the $succ$ function overestimates the probability of success of critical actions as the estimation of their application time is optimistic. Together with the observation that ach overestimates the probability of achieving the given goal, we derive the claim of the following proposition.

Proposition 10. *Let $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ be a RC planning task, $(A^c, \prec)$ be a CAS for $\mathcal{RP}$ and σ be adversary's mixed strategy. For each plan π^1 for a planning task $\mathcal{P} = (V, A^1, I, G^1)$ that follows $(A^c, \prec)$, it is the case that $cost((A^c, \prec), \sigma) \leq cost(\pi^1, \sigma)$.*

Generating CAS

Given the RC planning task and adversary's mixed strategy, we initially determine the set of all critical actions (see Def. 4). Then, we identify *mutex groups* of critical actions, that is, critical actions in a mutex group are pairwise mutually exclusive. Mutual exclusivity of two actions is identified by checking whether one action "permanently" deletes a precondition for the other one and the other way round (note that this is an underapproximation as the problem in general is intractable). For discovering disjunctive action landmarks, we used the back-chaining method (Hoffmann, Porteous, and Sebastia 2004) (which is again an underapproximation).

Initially, we randomly select a critical action being part of some disjunctive action landmark from each mutex group and randomly order those actions that have to be ordered (see Def. 7). Then, to find a good quality CAS (with as small cost as possible) we use a local search algorithm, depicted in Algorithm 2), leveraging simulated annealing (Kirkpatrick, Gelatt, and Vecchi 1983) in a similar fashion as Chrpa, Rytíř, and Horčík (2020). The SampleNextCAS function modifies CAS X by either swapping two randomly chosen critical actions ordered in X or replacing a randomly chosen critical action by another critical action that belongs to the same mutex group (each modification occurs with uniform probability). Note that modifications causing the value of $time$ for some action to be infinity are discarded.

Response Planning by Task Decomposition

As mentioned before, a CAS provides guidance for planners in form of what critical actions and in which order they have to be applied. Centralised planning even with the support of a CAS might still not scale very well.

We might however take advantage of structure of a subclass of RC planning tasks. If it is the case that the agent in RC planning tasks controls multiple *units* (e.g. a fleet of cars, or UAVs) that have only limited interaction, it might be possible under the circumstances we will specify later to plan for each unit individually as the CAS provides an essential information what unit does what critical action (and in which order). To give an example, we might observe that in the Taxi domain we might plan for each car individually if we know which passengers a particular car will serve (and in which order). Individual cars do not interfere or interact with each other as a sound CAS prevents situations in which more than one car try to pick up the same passenger. Hence we can observe that the union of plans for individual cars forms a response plan. In the Resource Hunting domain, the only difference is that UAVs might need to cooperate while collecting resources. Here, individual UAVs interact with each other only while performing critical actions. Hence, plans for individual UAVs might share some critical actions whose application times need to be synchronised, which is always possible since there is not a cyclic dependency among critical actions (otherwise CAS would not be sound).

Formally speaking, let $\mathcal{RP} = (V, A^1, A^2, I, G^1, G^2)$ be a RC planning task, $(A^c, \prec)$ be a CAS for $\mathcal{RP}$ and $\mathcal{A}^c$ be the set of all critical actions. Without loss of generality we assume that $\bigcup_{a \in A^1} vars(\textit{pre}(a)) \subseteq \bigcup_{a \in A^1} vars(\textit{eff}(a))$[1].

[1]Note that variables that cannot be modified by effects of any action can be compiled away such that actions requiring a value

We can divide agent's actions into subsets if the following conditions holds:

$$(A^1 \setminus \mathcal{A}^c) \cup A^c = A_1 \cup \cdots \cup A_k, \text{ where}$$
$$\forall i \neq j : (A_i \cap A_j \subseteq A^c) \text{ and } \forall a' \in A_i, \forall a'' \in A_j \setminus A_i : vars(\textit{eff}(a')) \cap vars(\textit{pre}(a'')) = \emptyset$$

It can be seen that critical actions not selected in the CAS are not part of any action set. The sets might overlap but only in the selected critical actions (e.g. two or more units collect a resource) as they have to be present in corresponding individual plans (to follow the CAS). Also, actions from one set cannot modify a variable for an action in another set (unless it is a joint critical action). This condition ensures that actions in individual plans will not interfere apart of joint critical actions.

For a subclass of RC planning tasks in which the agent controls a fleet of units we can divide agent's actions into sets corresponding to particular units and verify whether the above conditions hold. The information about which action controls which unit can be obtained from the task description (in PDDL).

Individual-unit planning tasks following the above decomposition of the set of actions can be constructed as follows. Let $\mathcal{P}_i = (V_i, A'_i, I_i, G_i)$ be such an individual-unit planning task over the set of actions A_i. The set of variables V_i is determined as $V_i = \bigcup_{a \in A_i} vars(\textit{eff}(a))$, i.e., V_i contains only variables that actions from A_i modify. Actions from A_i are modified such that their preconditions are projected to V_i, i.e., $A'_i = \{a' = (\textit{dur}(a), [\textit{pre}(a)|V_i], \textit{eff}(a)) \mid a \in A_i\}$. The initial state is projected to V_i, i.e., $I_i = [I|V_i]$. The soft goals from G^1 for which some of the minimal disjunctive action landmarks (MDAL) contain actions from A_i are considered, i.e., $G_i = \{G' \mid G' \in G, L \text{ being a MDAL for } G', L \cap A_i \neq \emptyset\}$.

The ordering constraints between critical actions specified in the CAS can be enforced during the (individual unit) plan generation by introducing special variables representing whether a corresponding critical action has been applied. These special variables are initially false and each critical action sets its special variable to be true. Also, each critical action requires the special variables of each of its direct predecessors to be true. Note that for sake of clarity the ordering constraint enforcement is not considered in the above representation of the individual-unit planning task.

After all individual-unit plans are generated, joint critical actions might be scheduled on different times in different plans. For each critical action that has to be "deconflicted" we determine its new application time as the maximum of the application times across all plans the action is present. Consequently, the actions dependent on the rescheduled action have to be rescheduled too (shifted by the same amount of time). The critical actions are deconflicted according to $\prec$ of the CAS.

It can be observed that the union of all individual plans (after "deconflicting") forms a plan that follows the CAS if it is sound and hence the plan is a response plan. Since the method for generating CAS uses some underapproximations, in some cases, it might result in generating unsound CAS (see Section "Generating CAS"). In such a case, we might need to resort to the complete approach for generating response plans (see Section "Towards Solving RC Planning Tasks").

We would like to note that for more complicated interactions between units we might use multi-agent planning (Torreño et al. 2018) although it has been shown that in case of "tight" interactions between the units centralised planning is a better option (Brafman and Domshlak 2008).

Experimental Evaluation

The aim of our experimental evaluation is to demonstrate that our approach combining the abstraction-based heuristic for generating CAS and decentralised planning (DO_ADP) scales better than the state of the art while maintaining reasonable quality of generated mixed strategies. In particular we compare DO_ADP against the plain combination of planning and Double Oracle (DO) (Rytíř, Chrpa, and Bošanský 2019), the delete-relaxation heuristic for generating CAS as a guidance for centralised planning (DO_RCP) (Chrpa, Rytíř, and Horčík 2020) and to directly compare the abstraction-based and relaxation-based heuristics, we combine the delete-relaxation heuristic for generating CAS of Chrpa, Rytíř, and Horčík with our decentralised planning approach (DO_RDP).

For the evaluation, we used the two domains, we have described earlier. Namely, we considered the Resource Hunting (RH) domain (Rytíř, Chrpa, and Bošanský 2019) and the Taxi domain.

We specified the domains and problem instances in PDDL 2.1 (Fox and Long 2003) and used Temporal Fast Downward (Eyerich, Mattmüller, and Röger 2009) to produce grounded state-variable representation and to compute T-DTGs. We used CPT4 (Vidal 2011), which is an optimal temporal planner, for generating "single-unit" plans. For centralised planning we encoded (best) response planning problem as Chrpa, Rytíř, and Horčík (2020) did and used the same cost-optimal planner – the potential heuristic (Pommerening et al. 2015) optimised by the diversification method proposed by Seipp, Pommerening, and Helmert (2015). The parameters for Algorithm 2 were set as $t = 150$, $K = 2.1$ and $\epsilon = 0.005$. The values of the parameters were determined experimentally on a small number of problems. We have observed that a too small initial temperature (t) leads to worse quality results while a too large initial temperature leads to higher runtime without improving the quality. A similar observation can be made for a step (ϵ). Setting it too large negatively affects solution quality while setting it too small leads to larger runtime with a negligible effect on quality. The experiments were run on a laptop with 8-core Intel Core i9 2.3GHz CPU with 16GB RAM[2].

The running times for the considered methods are displayed in Figure 1, where we present the results for problems in which the number of resources is twice as much

of the variable other than initial can be removed as they will never become applicable.

[2]Source code and experimental data can be found at: https://gitlab.com/FRASProject/socs22-distributed-adversarial-planning

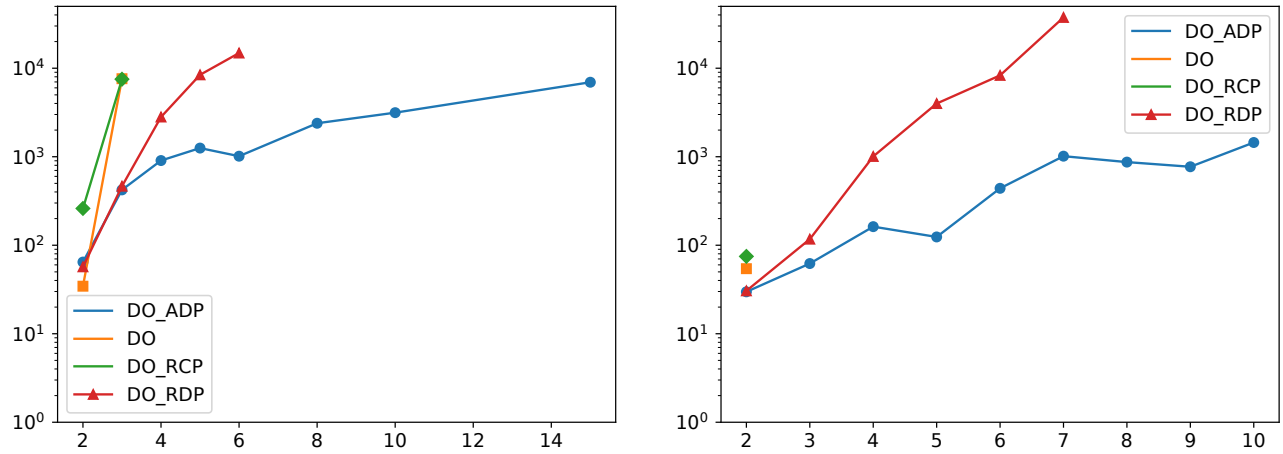

Figure 1: Running times of the considered methods for Resource Hunting problems (left) and Taxi problems (right). The number of resources is twice as much as the number of units.

as the number of units. Missing data points point out problems in which a particular method failed to find a solution either because we ran out of memory or the timeout was reached (1800s for a planning episode). It can be seen that centralised planning approaches (DO an DO_RCP) do not scale well as they could solve only very small problems. DO_RDP which combines delete-relaxation-based heuristic for generating CAS (Chrpa, Rytíř, and Horčík 2020) with our decentralised planning approach scaled better (up to 6 units in RH and 5 units in Taxi). The use of our abstraction-based heuristic for generating CAS further improved the scalability. Such a result highlights the fact that the delete-relaxation-based heuristic has to analyse the whole problem to compute time estimates while the abstraction-based heuristic needs to analyse only relevant part of the problem and needs to compute the $dist$ function for each pair of actions at most once.

In a nutshell, DO_ADP consists of three components: preprocessing (grounding, computing T-DTGs, landmarks etc.), generating CAS and planning. In problems with up to 9 units, more than 80% of runtime is taken by the CAS component. For larger problems, the runtime share of the planning and preprocessing components grows up to 70%. We have observed that the main bottleneck preventing DO_ADP to scale more is the grounding part of the preprocessing. For instance, it did not finish in several hours for RH problems with more than 20 units. We also conducted additional experiments that showed that DO_ADP can scale up to 20 units and 20 resources in RH and 14 units and 14 resources in Taxi.

To determine quality of mixed strategies generated by DO_ADP we have used two measures – exploitability (the difference of the best response plan against the strategy and the equilibrium value) and "almost symmetrical" scenarios (where the expected utility value is close to 0.5 of the game value). Exploitability could be calculated only on very small problems (up to 3 or 2 units for RH or Taxi, respectively). The average exploitability of the strategies found by DO_ADP was 0.13 and DO_RCP was 0.15 (smaller is better). In almost symmetrical problems, the normalised utility value of mixed strategies found by DO_ADP is 0.52 ± 0.05. Note that the standard deviation was slightly higher in TAXI (0.06) than in RH (0.03). Hence, the results show that DO_ADP generates mixed strategies of reasonable quality (close to equilibrium).

Conclusion

Resource-competition planning tasks represent a subclass of adversarial planning tasks in which agents compete for limited resources. From the game-theoretical standpoint, RC planning tasks fall under the umbrella of normal-form games that can be solved by the Double-Oracle algorithm. The (best) response in the context of RC planning tasks accounts for generating a plan optimised against a mixed strategy of the adversary. Besides formalising the concept in more general way than presented in literature (Rytíř, Chrpa, and Bošanský 2019; Chrpa, Rytíř, and Horčík 2020), we proposed an abstraction-based heuristic for estimating when resources can be collected and by what means which serves as a guidance for plan generation. Also, we specified under what conditions the task can be decomposed, which can be straightforwardly done for RC planning tasks in which the agent controls a fleet of units that can interact with each other only in a limited fashion (in joint critical actions). Hence, a plan can be generated for each individual unit and then the individual-unit plans are combined. The results show that both our contributions play important role in improving scalability of the approach while maintaining a reasonable quality of generated mixed strategies.

Although our concepts are presented for the sake of clarity for 2 agents (players), we believe that they can be straightforwardly extended to n agents (players) that compete against each other. The extension would involve adapting the Double Oracle algorithm, so the best responses are generated for each agent, as well as our concepts of adversary actions (where all the other agents' actions have to be considered) and the definition of the $succ$ function (see Definition 6), which would have to consider mixed strategies of

all the other agents.

In future, we plan to investigate how the presented concepts can be effectively used in online planning (e.g. alongside MCTS), in which the agent would iteratively plan for the "nearest" critical actions.

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

Applegate, C.; Elsaesser, C.; and Sanborn, J. C. 1990. An architecture for adversarial planning. *IEEE Trans. Systems, Man, and Cybernetics*, 20(1): 186–194.

Björnsson, Y.; and Finnsson, H. 2009. CadiaPlayer: A Simulation-Based General Game Player. *IEEE Trans. Comput. Intell. AI Games*, 1(1): 4–15.

Brafman, R. I.; and Domshlak, C. 2008. From One to Many: Planning for Loosely Coupled Multi-Agent Systems. In *The 18th International Conference on Automated Planning and Scheduling, ICAPS 2008*, 28–35.

Chrpa, L.; Rytíř, P.; and Horčík, R. 2020. Planning Against Adversary in Zero-Sum Games: Heuristics for Selecting and Ordering Critical Actions. In *The 13th International Symposium on Combinatorial Search, SOCS 2020*, 20–28.

Cimatti, A.; Pistore, M.; Roveri, M.; and Traverso, P. 2003. Weak, strong, and strong cyclic planning via symbolic model checking. *Artif. Intell.*, 147(1-2): 35–84.

Eyerich, P.; Mattmüller, R.; and Röger, G. 2009. Using the Context-enhanced Additive Heuristic for Temporal and Numeric Planning. In *The 19th International Conference on Automated Planning and Scheduling, ICAPS 2009*.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.*, 20: 61–124.

Haslum, P. 2009. $h^m(P) = h^1(P^m)$: Alternative Characterisations of the Generalisation From $h^{\max}$ To h^m. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009*.

Hoffmann, J.; Porteous, J.; and Sebastia, L. 2004. Ordered Landmarks in Planning. *J. Artif. Intell. Res.*, 22: 215–278.

Jensen, R.; Veloso, M.; and Bowling, M. 2001. OBDD-based optimistic and strong cyclic adversarial planning. In *ECP*, 265–276.

Jonsson, P.; and Bäckström, C. 1998. State-Variable Planning Under Structural Restrictions: Algorithms and Complexity. *Artif. Intell.*, 100(1-2): 125–176.

Justesen, N.; and Risi, S. 2017. Continual online evolutionary planning for in-game build order adaptation in StarCraft. In *The Genetic and Evolutionary Computation Conference, GECCO 2017*, 187–194.

Kirkpatrick, S.; Gelatt, C. D.; and Vecchi, M. P. 1983. Optimization by Simulated Annealing. *Science*, 220(4598): 671–680.

Kissmann, P.; and Edelkamp, S. 2009. Solving Fully-Observable Non-deterministic Planning Problems via Translation into a General Game. In *KI*, volume 5803, 1–8. Springer.

LaValle, S. M. 2006. *Planning Algorithms*. Cambridge University Press. ISBN 9780511546877.

Lelis, L. H. S. 2020. Planning Algorithms for Zero-Sum Games with Exponential Action Spaces: A Unifying Perspective. In *The 29th International Joint Conference on Artificial Intelligence (IJCAI)*, 4892–4898.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the Presence of Cost Functions Controlled by an Adversary. In *ICML*, 536–543.

Pommerening, F.; Helmert, M.; Röger, G.; and Seipp, J. 2015. From Non-Negative to General Operator Cost Partitioning. In *Proc. AAAI'15*, 3335–3341.

Pozanco, A.; E-Martín, Y.; Fernández, S.; and Borrajo, D. 2018. Counterplanning using Goal Recognition and Landmarks. In *The 27th International Joint Conference on Artificial Intelligence, IJCAI 2018*, 4808–4814.

Rytíř, P.; Chrpa, L.; and Bošanský, B. 2019. Using Classical Planning in Adversarial Problems. In *the 31st IEEE International Conference on Tools with Artificial Intelligence*, 1327–1332.

Seipp, J.; Pommerening, F.; and Helmert, M. 2015. New Optimization Functions for Potential Heuristics. In *Proc. ICAPS'15*, 193–201.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2018. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419): 1140–1144.

Speicher, P.; Steinmetz, M.; Backes, M.; Hoffmann, J.; and Künnemann, R. 2018. Stackelberg planning: Towards effective leader-follower state space search. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Torralba, Á.; Speicher, P.; Künnemann, R.; Steinmetz, M.; and Hoffmann, J. 2021. Faster Stackelberg Planning via Symbolic Search and Information Sharing. In *the 35th AAAI Conference on Artificial Intelligence*, 11998–12006.

Torreño, A.; Onaindia, E.; Komenda, A.; and Stolba, M. 2018. Cooperative Multi-Agent Planning: A Survey. *ACM Comput. Surv.*, 50(6): 84:1–84:32.

Vidal, V. 2011. CPT4: An Optimal Temporal Planner Lost in a Planning Competition without Optimal Temporal Track. In *Booklet of the 7th International Planning Competition (IPC-2011)*, 25–28.

Vinyals, O.; Babuschkin, I.; Czarnecki, W.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.; Powell, R.; Ewalds, T.; Georgiev, P.; Oh, J.; Horgan, D.; Kroiss, M.; Danihelka, I.; Huang, A.; Sifre, L.; Cai, T.; Agapiou, J.; Jaderberg, M.; and Silver, D. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575.

Vorobeychik, Y.; and Pritchard, M. 2020. Plan Interdiction Games. In *Adaptive Autonomous Secure Cyber Systems*, 159–182. Springer.