Multiagent Systems (BE4M36MAS)

Solving Extensive-Form Games and Other Representations of Dynamic Games

Branislav Bošanský

Artificial Intelligence Center,
Department of Computer Science,
Faculty of Electrical Engineering,
Czech Technical University in Prague

branislav.bosansky@agents.fel.cvut.cz

November 5, 2019

Previously ... on multi-agent systems.

1. Solving Extensive-Form Games
2. Sequence-Form Representation

Sequence Form Linear Program (SQF)

We are now ready to state the linear program:

$$\max_{r_1, v} v(root) \tag{1}$$

$$\text{s.t.} \quad r_1(\emptyset) = 1 \tag{2}$$

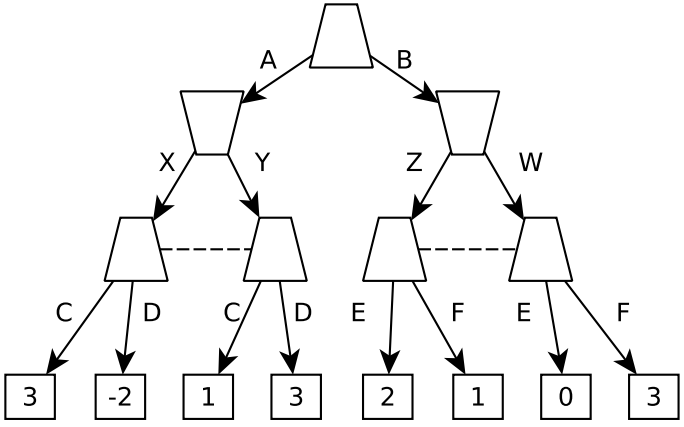
$$0 \leq r_1(\sigma_1) \leq 1 \qquad \forall \sigma_1 \in \Sigma_1 \tag{3}$$

$$\sum_{a \in \mathcal{A}(I_1)} r_1(\sigma_1 a) = r_1(\sigma_1) \quad \forall I_1 \in \mathcal{I}_1, \sigma_1 = \mathsf{seq}_1(I_1) \tag{4}$$

$$\sum_{I' \in \mathcal{I}_2 : \sigma_2 a = \mathsf{seq}_2(I')} v(I') + \sum_{\sigma_1 \in \Sigma_1} g(\sigma_1, \sigma_2 a) r_1(\sigma_1) \geq v(I) \qquad \forall I \in \mathcal{I}_2, \sigma_2 = \mathsf{seq}_2(I), \forall a \in \mathcal{A}(I) \tag{5}$$

- $\mathsf{seq}_i(I)$ is a sequence of player i to information set,
- $I \in \mathcal{I}_i$, v_I is an expected utility in an information set,
- $\mathsf{inf}_i(\sigma_i)$ is an information set, where the last action of σ_i has been executed,
- $\sigma_i a$ denotes an extension of a sequence σ_i with action a

Sequence Form LP - Example



$$\max_{r_1, v} v(\mathsf{inf}_2(X)) + v(\mathsf{inf}_2(Z)) \tag{6}$$

$$r_1(\emptyset) = 1; r_1(A) + r_1(B) = r_1(\emptyset) \tag{7}$$

$$r_1(AC) + r_1(AD) = r_1(A), \tag{8}$$

$$r_1(BE) + r_1(BF) = r_1(B) \tag{9}$$

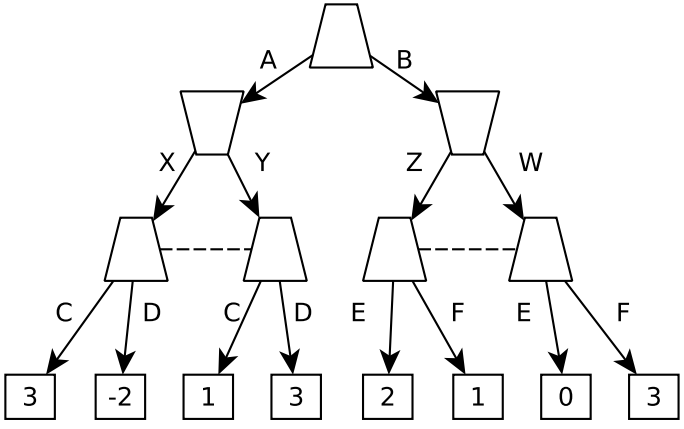
$$v(\mathsf{inf}_2(X)) \leq 0 + g(AC, X) r_1(AC) + g(AD, X) r_1(AD) \tag{10}$$

$$v(\mathsf{inf}_2(Y)) \leq 0 + g(AC, Y) r_1(AC) + g(AD, Y) r_1(AD) \tag{11}$$

$$v(\mathsf{inf}_2(Z)) \leq 0 + g(BE, Z) r_1(BE) + g(BF, Z) r_1(BF) \tag{12}$$

$$v(\mathsf{inf}_2(W)) \leq 0 + g(BE, W) r_1(BE) + g(BF, W) r_1(BF) \tag{13}$$

Sequence Form LP - Example



$$\min_{r_2, v} v(\mathsf{inf}_1(A)) \tag{14}$$

$$r_2(\emptyset) = 1; r_2(X) + r_2(Y) = r_2(\emptyset) \tag{15}$$

$$r_2(Z) + r_2(W) = r_2(\emptyset) \tag{16}$$

$$v(\mathsf{inf}_1(A)) \geq v(\mathsf{inf}_1(AC)),\ v(\mathsf{inf}_1(B)) \geq v(\mathsf{inf}_1(BE)) \tag{17}$$

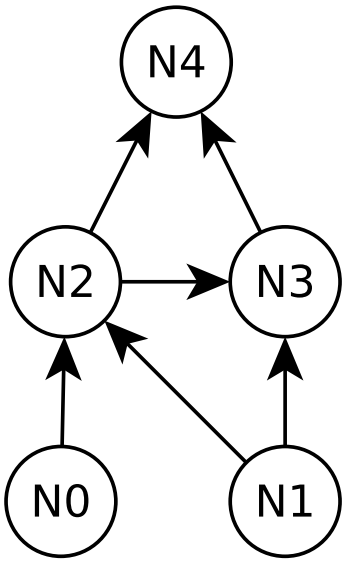
$$v(\mathsf{inf}_1(AC)) \geq g(AC, X)r_2(X) + g(AC, Y)r_2(Y) \tag{18}$$

$$v(\mathsf{inf}_1(AD)) \geq g(AD, X)r_2(X) + g(AD, Y)r_2(Y) \tag{19}$$

$$v(\mathsf{inf}_1(BE)) \geq g(BE, Z)r_2(Z) + g(BE, W)r_2(W) \tag{20}$$

$$v(\mathsf{inf}_1(BF)) \geq g(BF, Z)r_2(Z) + g(BF, W)r_2(W) \tag{21}$$

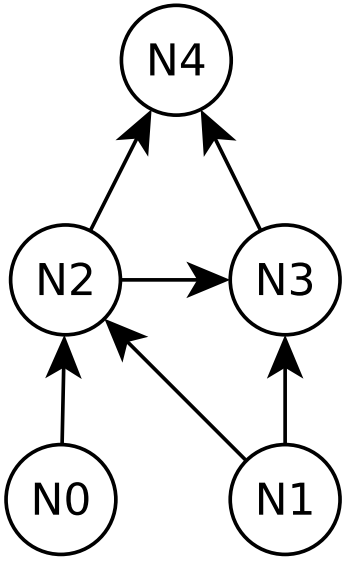
Simple Network Security Scenario – Flip-It Game



Flip-it Game in a network

- players aim to gain control over the hosts in the network
- the defender initially controls all hosts
- both players choose which node to attack/protect simultaneously (in case of a tie, the control of the node does not change)
- players only observe the result of their last move
- there are different rewards/costs for each node

Simple Network Security Scenario – Flip-It Game



SQF for Flip-it Game in a network

Depth	Size (# Nodes)	Time [s]	LP Time [s]
3	15,685	1	1
4	495,205	23	8
5	16,715,941	–	–

Advantages/Disadvantages of SQF

(+) the fastest exact algorithm (if the LP fits into memory)

(+) quite easy to implement

(−) scales poorly due to memory limitations

(−) very difficult to make it domain-specific

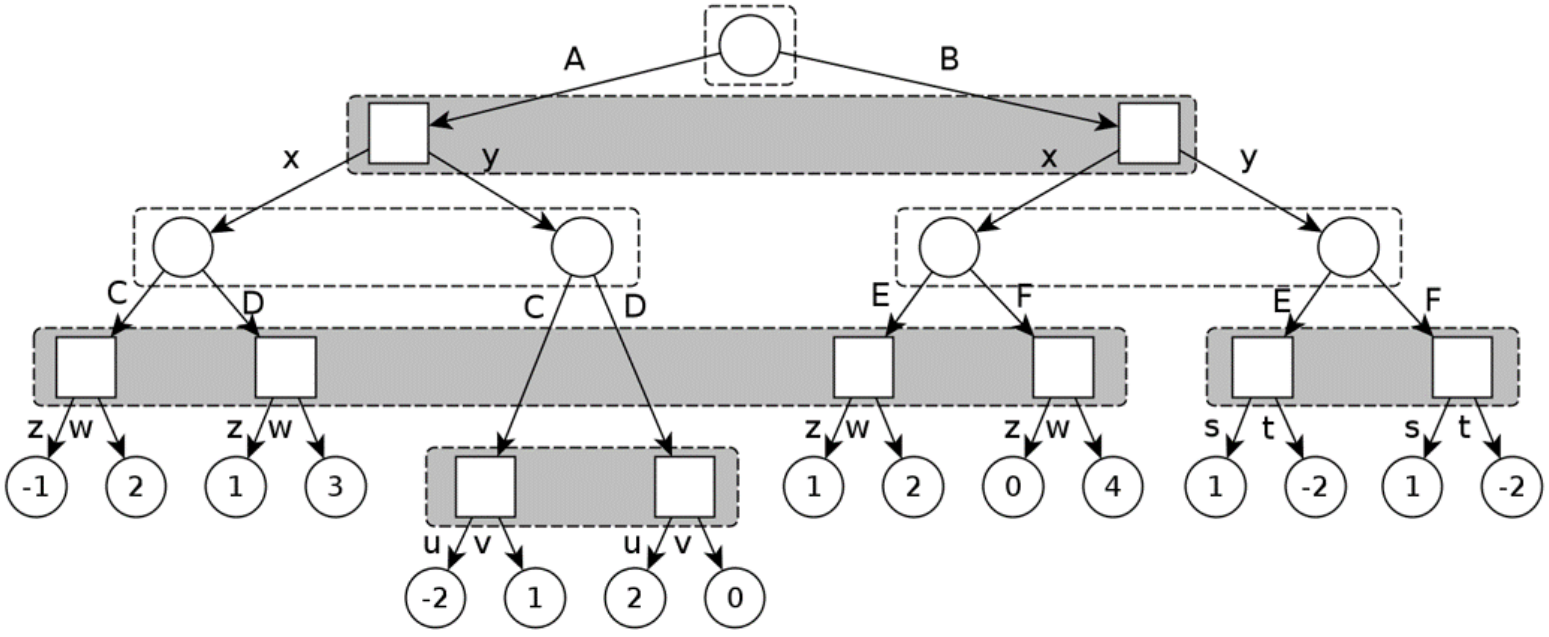
Incremental Strategy Generation

Large linear programs can be solved by an incremental construction of the LP. In game theory, the method has been known as *double-oracle algorithm*. There are 4 steps that repeat until convergence [Bosansky et al., 2014]:

1. **create a restricted game** – a simplified game where the players are allowed to choose only from a limited set of sequences of actions,
2. **solve the restricted game** – formalize the restricted game as a sequence-form LP and solve it,
3. **compute the best response** – each player computes a best response in the **original game** to the strategy from the **restricted game**,
4. **expand the restricted game** – if the best responses strictly improve the expected value, they are added as possible actions into the restricted game.

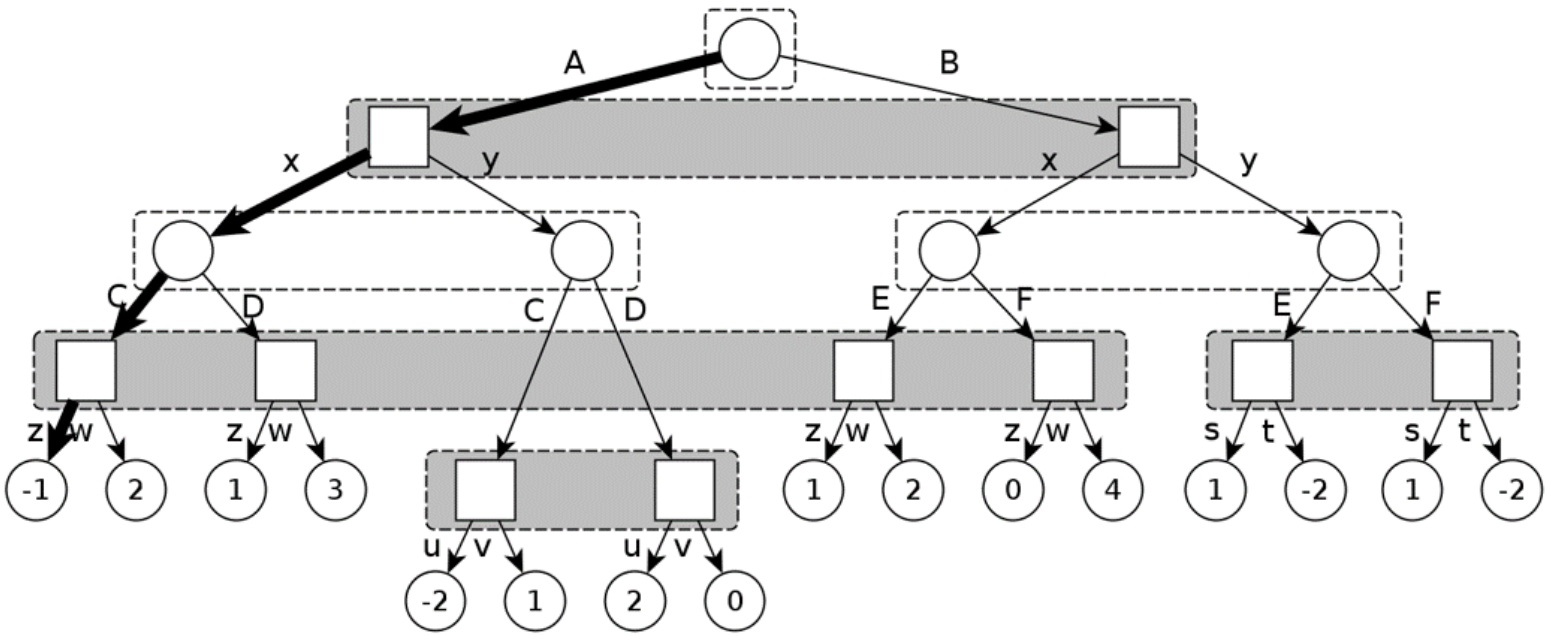
Double Oracle Algorithm for EFGs

The original game. Sequences that form the restricted game will be highlighted.



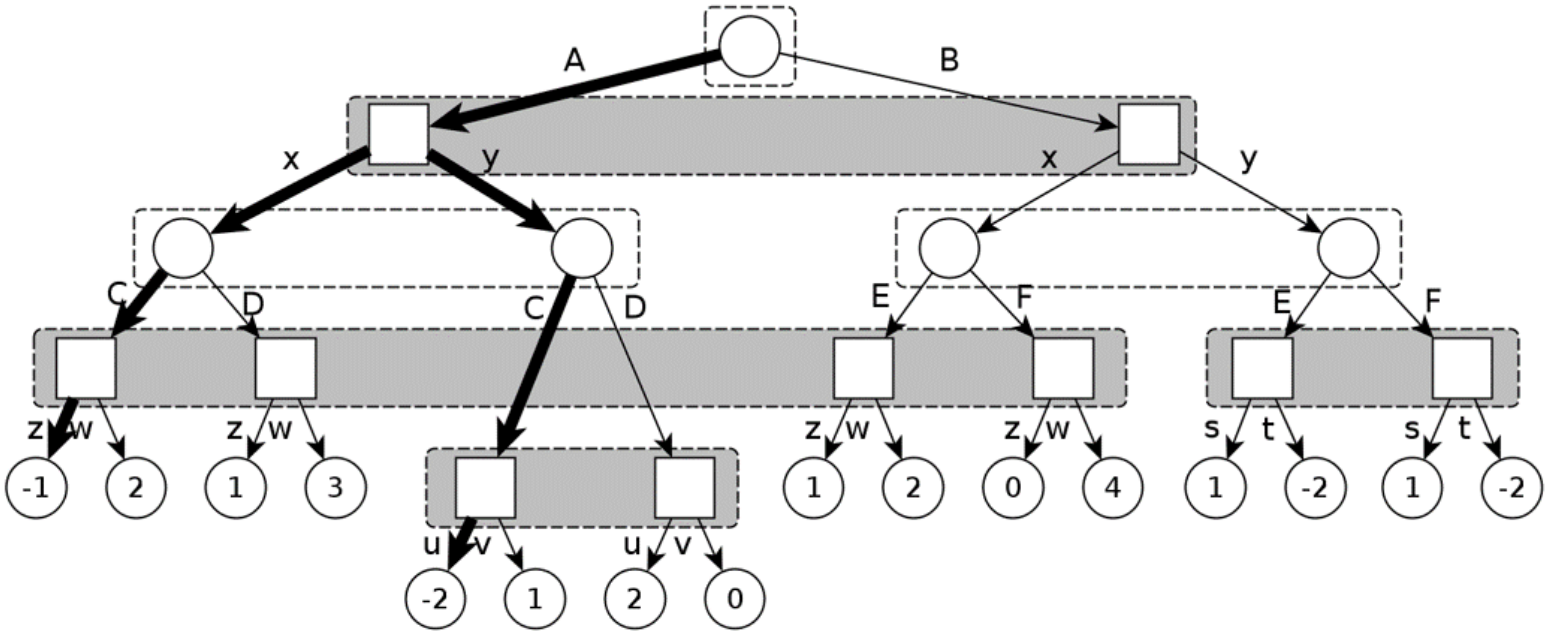
Double Oracle Algorithm for EFGs

Sequences AC and xz are added to the restricted game (as default sequences of actions).



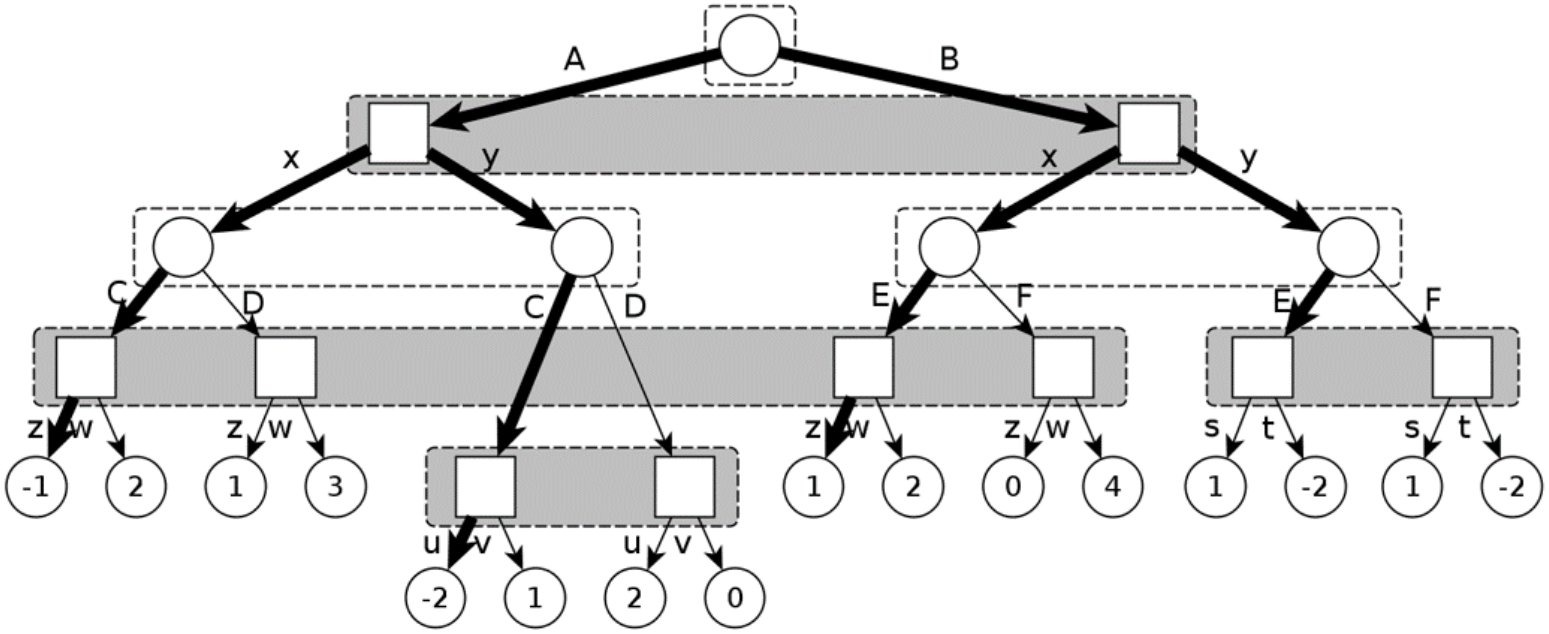
Double Oracle Algorithm for EFGs

Sequence yu is added to the restricted game as a best response of the minimizing player.



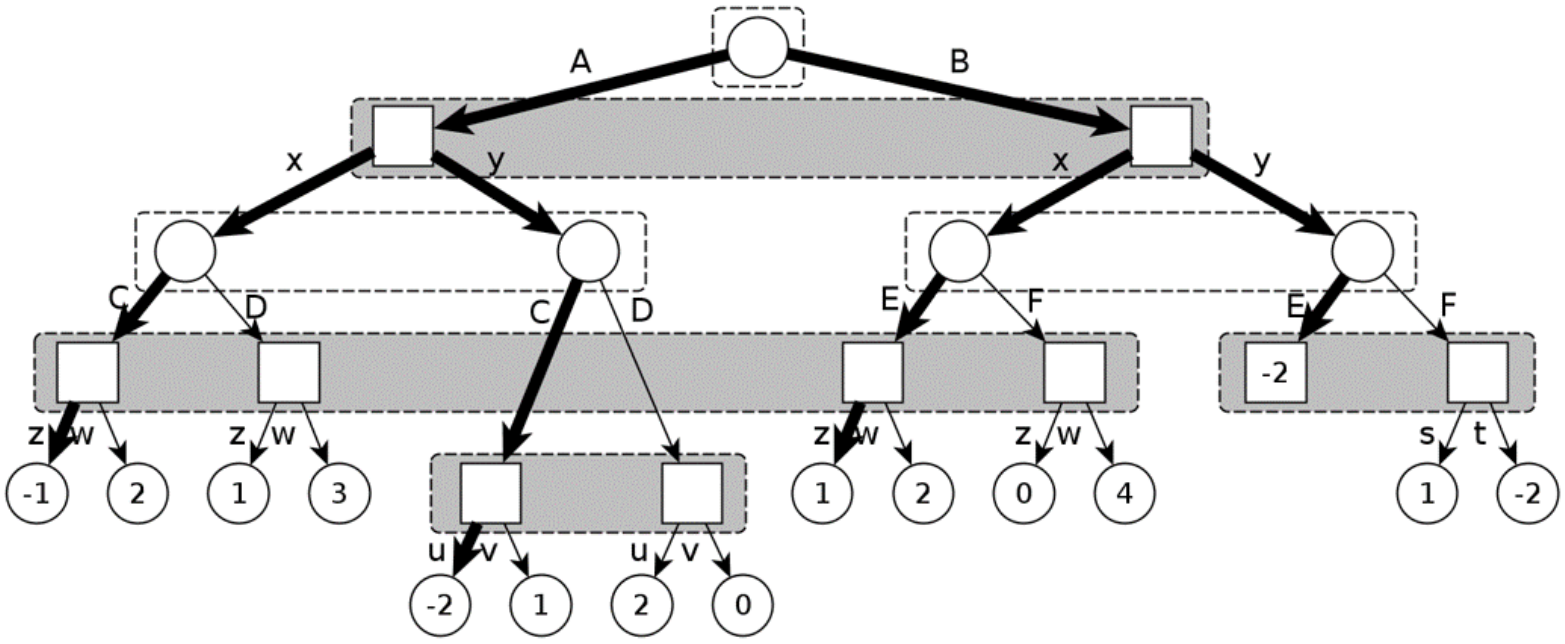
Double Oracle Algorithm for EFGs

Sequence BE is added to the restricted game as a best response of the maximizing player.



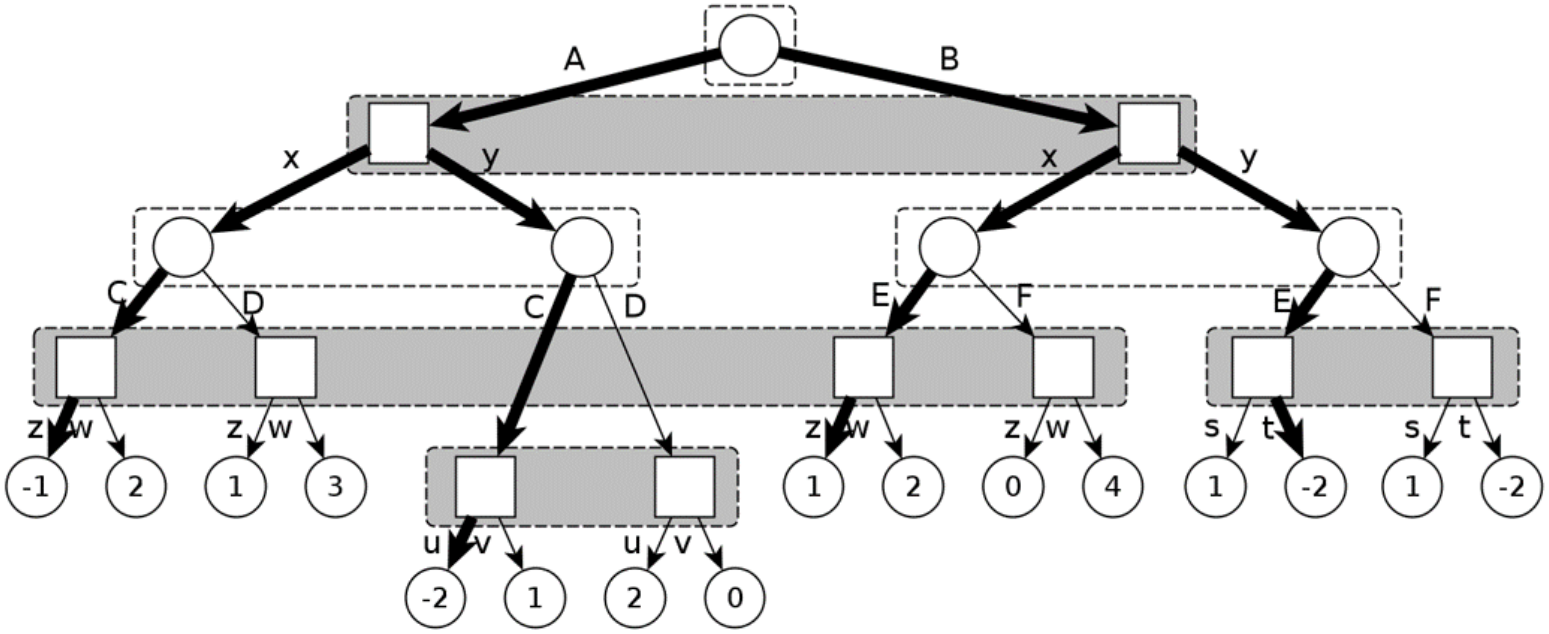
Double Oracle Algorithm for EFGs

There is no action defined for the node with history ByE. The algorithm turns that node into a temporary leaf and assigns a temporary utility value for that leaf.



Double Oracle Algorithm for EFGs

The algorithm turns the temporary leaf into a node when an action s or t is added into the restricted game.



Characteristics of DOEFG

Generalization of the double oracle principle to structured strategy spaces (such as sequences/realization plans).

Creating a valid restricted game is more complicated than adding a single strategy (one may need to create temporary leaves).

DOEFG converges in at most linear number of iterations in the size of the game tree (compared to the exponential number of iterations when using strategies).

Simple Network Security Scenario – Flip-It Game

DOEFG for Flip-it Game in a network

Depth	# Nodes	SQF [s]	SQF LP [s]	DOEFG [s]
3	15,685	1	1	1
4	495,205	23	8	9
5	16,715,941	–	–	508

Advantages/Disadvantages of DOEFG

(+) can solve much larger domains compared to SQF

(+) in a domain-independent way, the algorithm identifies necessary strategies to consider in a large EFG

(+) best-response algorithms can be significantly improved for specific domains/problems

(−) not that easy to implement

(−) the sequence-form linear program of the restricted game can be a bottleneck

Simple Network Security Scenario – Flip-It Game

DOEFG with ordered moves for BR algorithm for Flip-it Game in a network

Depth	# Nodes	SQF [s]	SQF LP [s]	DOEFG [s]	DOEFG ordered [s]
3	15,685	1	1	1	1
4	495,205	23	8	9	5
5	16,715,941	–	–	508	168

For depth 6 (size $\approx 4 \times 10^9$ nodes), DOEFG with ordered moves for BR reached error 0.1 in 2 hours.

Approximate Algorithms for Extensive-Form Games

Algorithms based on *Counterfactual Regret Minimization*

Approximate Algorithms for Extensive-Form Games

Instead of computing the optimal strategy directly, one can employ learning algorithms and learn the strategy via repeated (simulated, or self-) play.

The algorithm minimizes so called *regret* and these algorithms are also known as *no-regret learning* algorithms.

Main idea:

- in each iteration, traverse through the game tree and adapt the strategy in each information set according to the learning rule
- this learning rule minimizes the (counterfactual) regret
- the algorithm minimizes the overall regret in the game
- the average strategy converges to the optimal strategy

Regret and Counterfactual Regret

Player i's regret for *not playing* an action a_i' against opponent's action a_{-i}

$$u_i(a_i', a_{-i}) - u_i(a_i, a_{-i})$$

In extensive-form games we need to evaluate the value for each action in an information set *(counterfactual value)*

$$v_i(s, I) = \sum_{z \in \mathcal{Z}_I} \pi^s_{-i}(z[I]) \pi^s_i(z|z[I]) u_i(z),$$

where

- $\mathcal{Z}_I$ are leafs reachable from information set I
- $z[I]$ is the history prefix of z in I
- $\pi^s_i(h)$ is the probability of player i reaching node h following strategy s

Regret and Counterfactual Regret

Counterfactual value for one deviation in information set I; strategy s is altered in information set I by playing action $a : v_i(s_{I \to a}, I)$

at a time step t, the algorithm computes *counterfactual regret* for current strategy

$$r_i^t(I, a) = v_i(s_{I \to a}, I) - v_i(s_I, I)$$

the algorithm calculates the *cumulative regret*

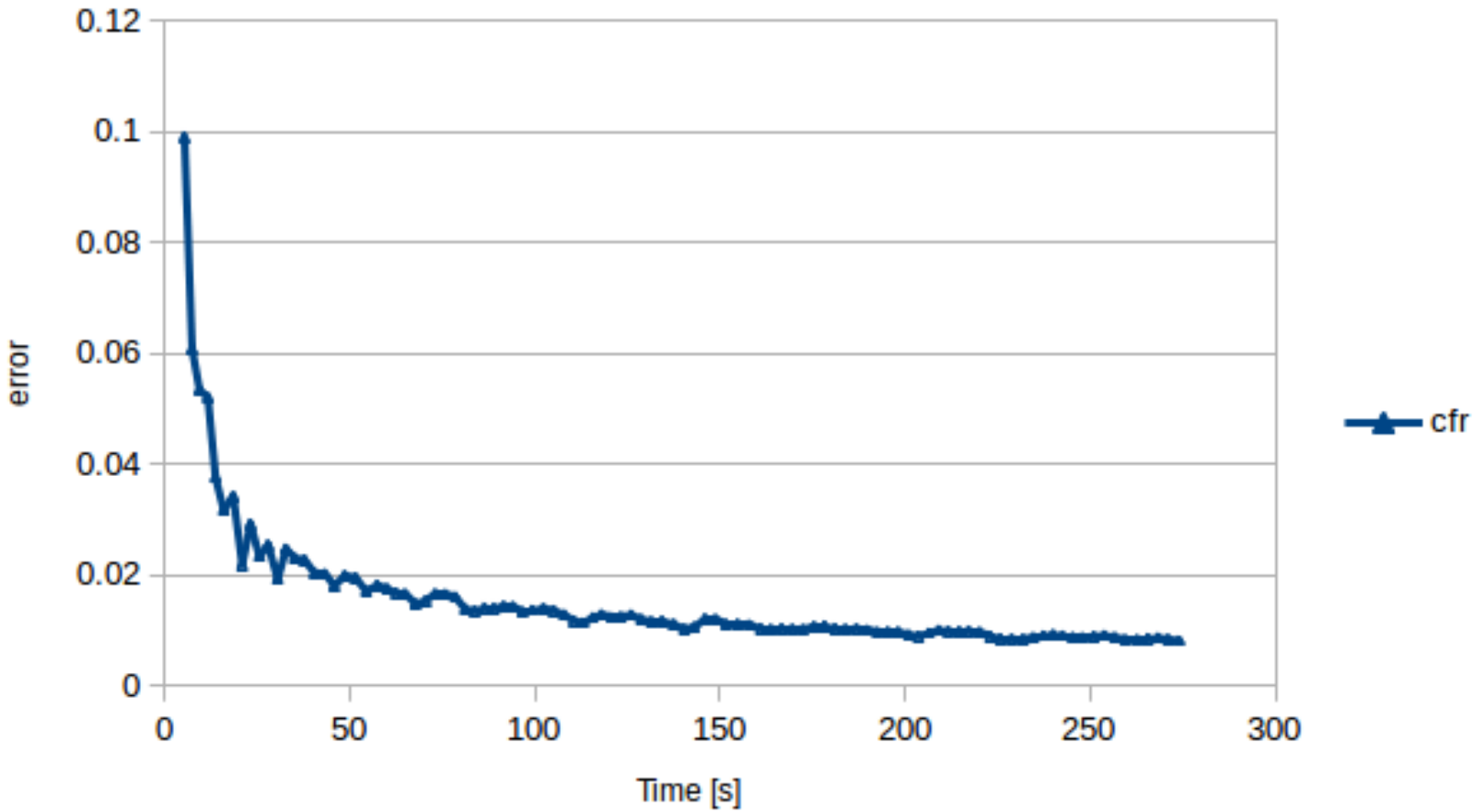
$$R_i^T = \sum_{t=1}^{T} r_i^t(I, a), \qquad R_i^{T,+}(I, a) = \max\{R_i^T(I, a), 0\}$$

strategy for the next iteration is selected using *regret matching*

$$s_i^{t+1}(I, a) = \begin{cases} \dfrac{R_i^{T,+}(I,a)}{\sum_{a' \in \mathcal{A}(I)} R_i^{T,+}(I,a')} & \text{if the denominator is positive} \\ \dfrac{1}{|\mathcal{A}(I)|} & \text{otherwise} \end{cases}$$

Simple Network Security Scenario – Flip-It Game

CFR for Flip-it Game in a network[1]



[1] With the game tree pre-built in memory (took 1088s).

Extensions of Counterfactual Regret Minimization

There are **many** variants of the vanilla CFR algorithm:

- **MCCFR** – CFR updates are not performed in the complete game, but using outcome sampling (faster iterations) [Lanctot, 2013, Brown and Sandholm, 2016]
- **CFR-BR** – the second player performs a best-response (BR) update instead of a CFR update (ideal for games where a domain-specific BR algorithm is available) [Johanson et al., 2011]
- **CFR-D** – decomposition of CFR updates by subgames (helpful if the game is too large to keep all information sets in memory) [Burch et al., 2014]
- **CFR+** – main modification of the baseline CFR algorithm that significantly improves convergence [Tammelin, O. 2014]

Extensions of Counterfactual Regret Minimization (CFR+)

CFR+ differs from CFR in three aspects:

- only positive regrets are kept in cumulative regrets R_i^T
- players are alternating in the updates
- in the computation of the average strategy, first d iterations are ignored, later iterations are more important compared to first iterations

Sometimes, even the current strategy reaches low exploitability.

Extensions of Counterfactual Regret Minimization (CFR+)

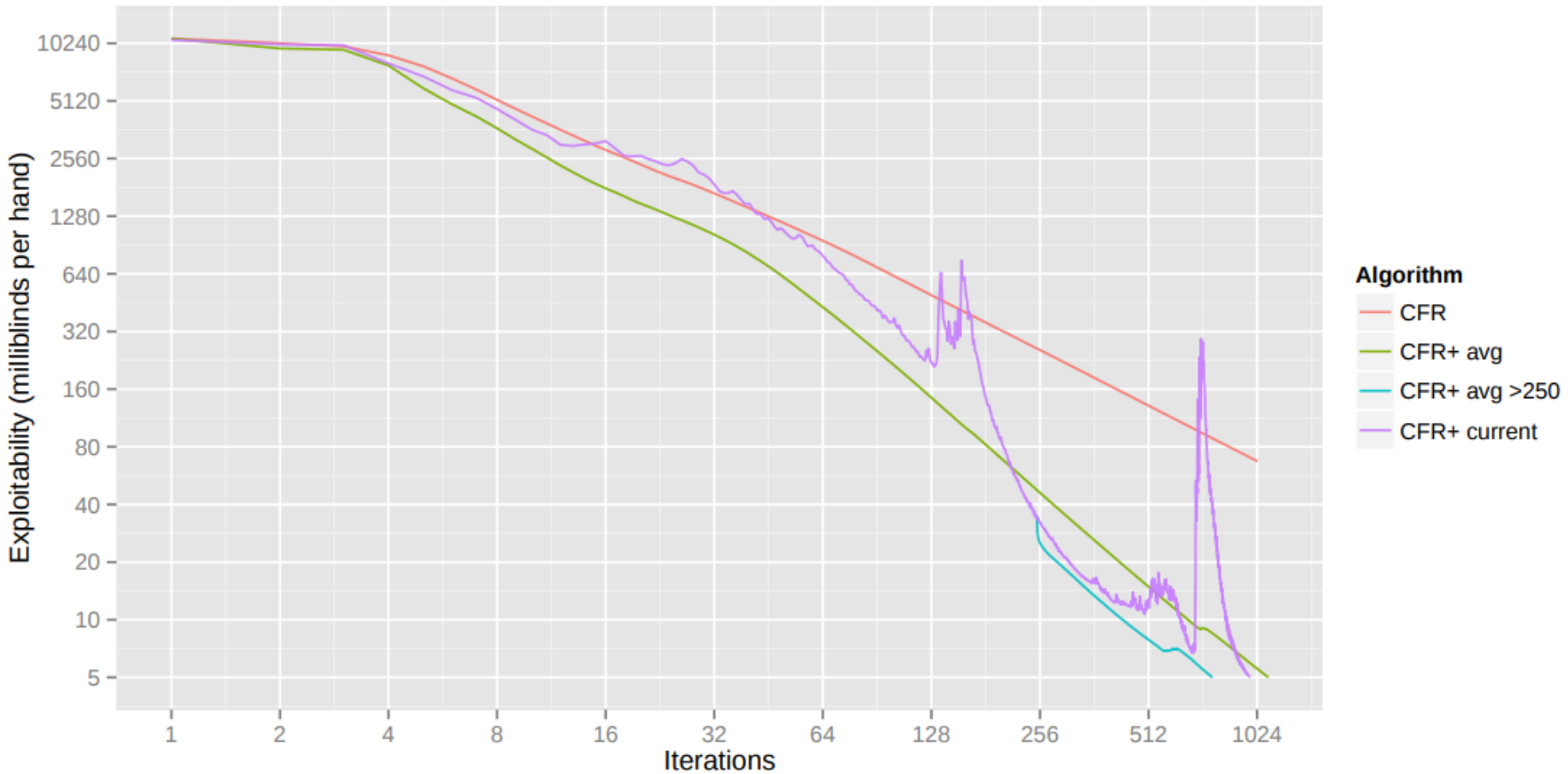


Figure 2: No Limit Texas Hold'em flop subgame

[2]Figure from [Tammelin, O. 2014].

Advantages/Disadvantages of CFR

(+) practical optimization algorithm

(+) easy to implement [Lanctot, 2013, p.22]

(+) memory requirements can be reduced with domain-specific implementation (or CFR-D)

(−) CFR converges very slowly if a close approximation is required (CFR+ is better)

(−) performance in other domains than poker is largely unknown (in some cases slower than DOEFG)

Continual Resolving and Deepstack

Is there no hope for a provably algorithm that behaves similarly to perfect information games?

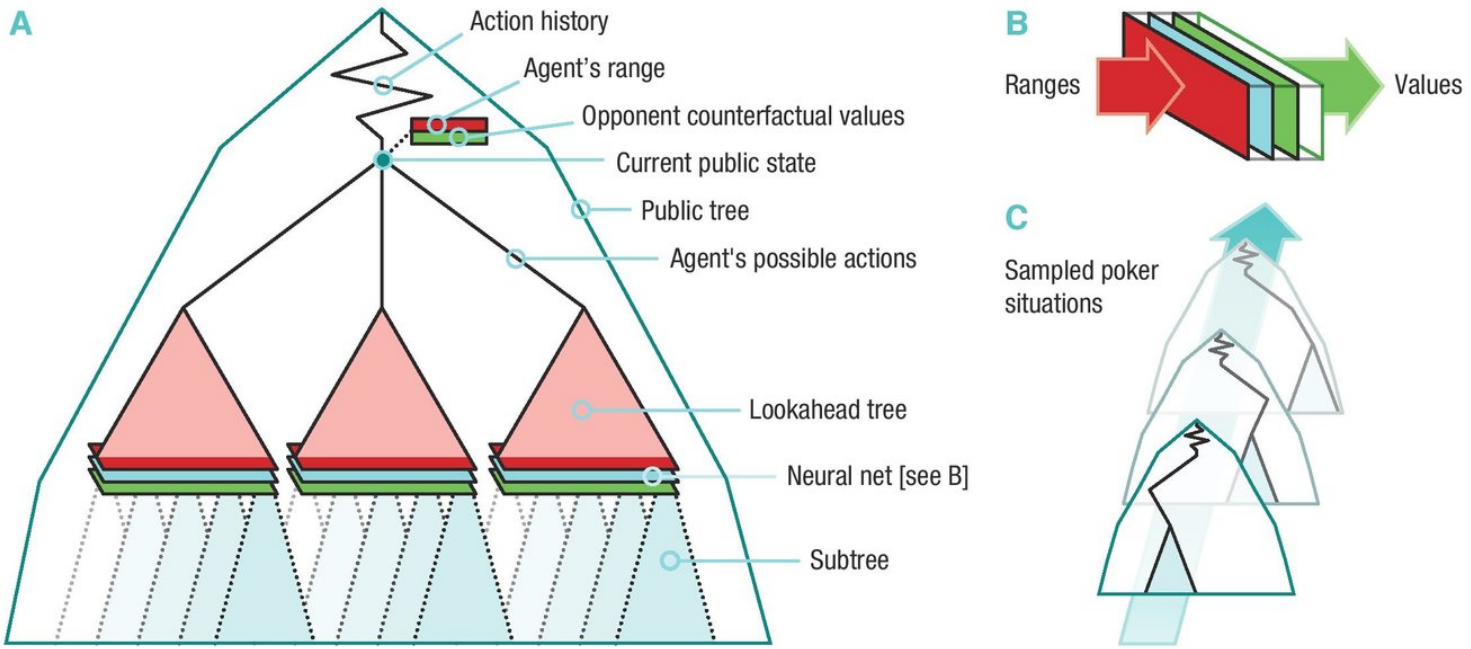
Recently, new methods that allow limited-lookahead algorithm for imperfect information games for poker
[Moravcik et al., 2017, Brown and Sandholm, 2017].

Key properties:

- Use (a more complex) heuristic function to evaluate positions at the end of the depth-limited game tree
- Solve an EFG with a limited lookahead (e.g., using CFR or other algorithm)
- Use a specific gadget construction when advancing to next turn of the game.

One cannot assign a heuristic value just to a state (as in perfect information games), but to all states players consider possible.

Continual Resolving and Deepstack



[3] Picture from [Moravcik et al., 2017].

Generalization of Continual Resolving

Adaptation of continual resolving technique to other (security) domains is not straightforward:

- the actions are generally not observable (the defender does not know which host the attacker infected)
- the size of information sets (in number of possible states) increases exponentially with number of turns in the game
  - the size of the information sets is changing for the heuristic/neural network
  - the size of the information sets becomes impractical for large horizon
- the number of turns can be very large (e.g., Advanced Persistent Threats (APTs))

Other Representations

Other representations for dynamic / sequential games.

Extensive-Form Games

Let's assume that we want to play some normal-form game twice.
For example, rock-paper-scissors:

	$\mathbf{R}$	$\mathbf{P}$	$\mathbf{S}$
$\mathbf{R}$	$(0, 0)$	$(-1, 1)$	$(1, -1)$
$\mathbf{P}$	$(1, -1)$	$(0, 0)$	$(-1, 1)$
$\mathbf{S}$	$(-1, 1)$	$(1, -1)$	$(0, 0)$

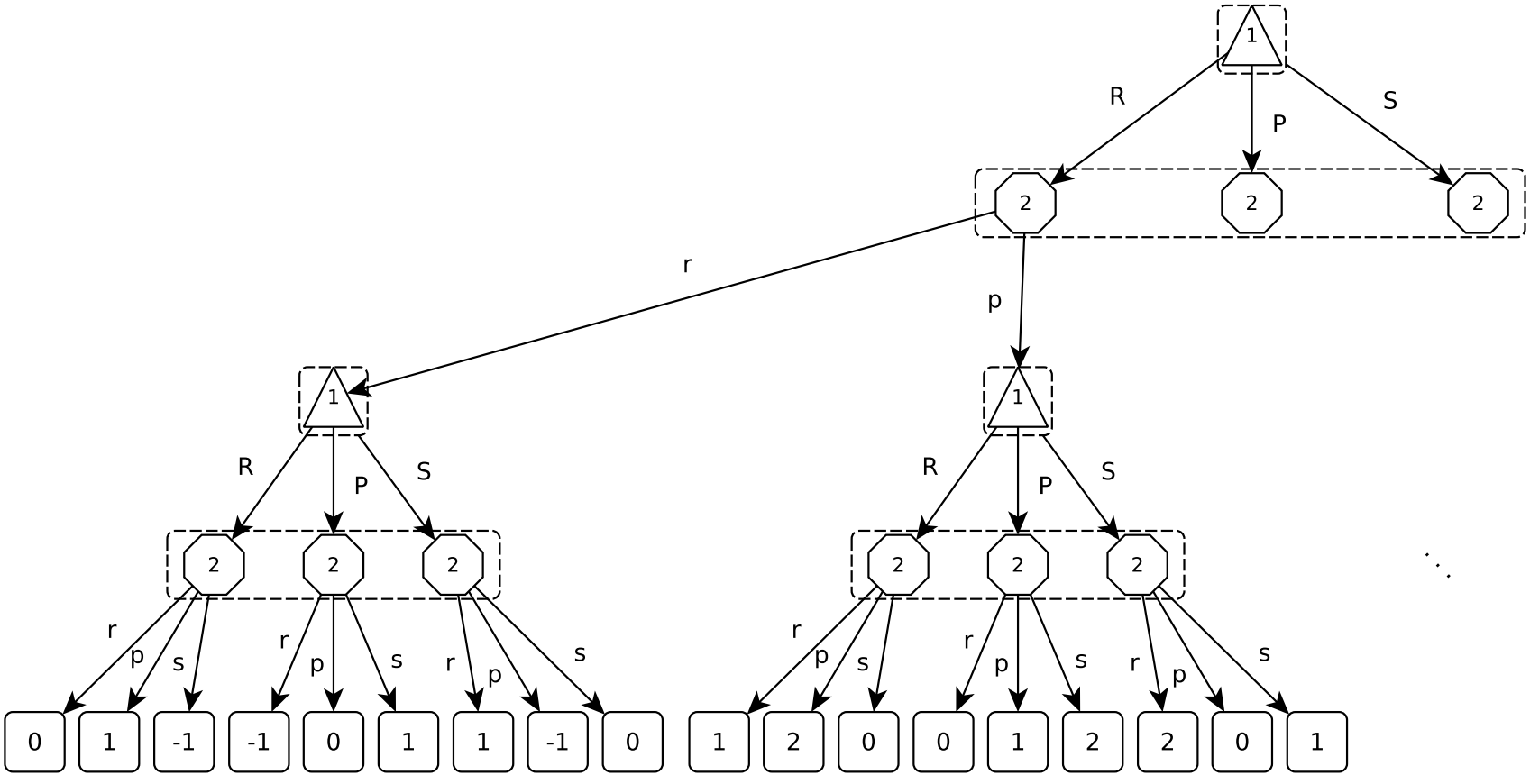
Question

How can we model such games?

We can model the game as an extensive-form game.

Pros: we already know how to solve such a game.
Cons: it is unnecessarily large.

RPS Played Twice as an Extensive-Form Games



We can use a model specific for repeated games.

Finitely Repeated Games

Definition

In repeated games we assume that a normal-form game, termed the *stage game*, is played repeatedly. If the number of repetitions (or rounds) is finite, we talk about *finitely repeated games*.

Question

How can we solve finitely repeated games?

We can use backward induction.

Why does this work if we have an extensive-form game with imperfect information?

Infinitely Repeated Games

Definition

Assume that a *stage game* is played repeatedly. If the number of repetitions (or rounds) is infinite, we talk about *infinitely repeated games*.

We cannot use extensive-form games as a underlying model. There are no leafs to assign utility values to. We need to define other utility measures:

Definition

Given an infinite sequence of payoffs $r_i^{(1)}, r_i^{(2)}, \ldots$ for player i, the *average reward* of i is

$$\lim_{k \to \infty} \frac{\sum_{j=1}^{k} r_i^{(j)}}{k}$$

Infinitely Repeated Games

Definition

Given an infinite sequence of payoffs $r_i^{(1)}, r_i^{(2)}, \ldots$ for player i, and a discount factor β with $0 \leq \beta \leq 1$, the *future discounted reward* is

$$\sum_{j=1}^{\infty} \beta^j r_i^{(j)}$$

Why do we use discount factor?

- a player cares more about immediate rewards
- a repeated game can terminate after each round with probability $1 - \beta$

Strategies in Repeated Games

How can we represent the strategies in infinitely repeated games? (the game tree is infinite)

- *a stationary strategy* – a randomized strategy that is played in each stage game

Is this enough? Consider a repeated prisoners dilemma – what is the most famous strategy in repeated prisoners dilemma?

Tit-for-tat: the player starts by cooperating and thereafter chooses in round $j+1$ the action chosen by the other player in round j.

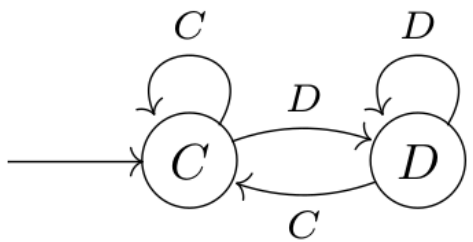
We can have more complex strategies consisting of states/machines.

Strategies in Repeated Games

Definition

Given a game $G = (N, A, u)$ that will be played repeatedly, an automaton M_i for player i is a four-tuple $(Q_i, q_i^0, \delta_i, f_i)$, where:

- Q_i is a set of states;
- q_i^0 is the start state;
- δ_i defines a transition function mapping the current state and an action profile to a new state, $\delta_i : Q_i \times A \to Q_i$
- f_i is a strategy function associating with every state an action for player i, $f_i : Q_i \to A_i$.



A strategy for Tit-for-Tat

Strategies in Repeated Games

Definition

A payoff profile $r = (r_1, r_2, \ldots, r_n)$ is *enforceable* if $\forall i \in \mathcal{N}$, $r_i \geq v_i$.

where v_i is a minmax value for player i

$$v_i = \min_{s_{-i} \in \mathcal{S}_{-i}} \max_{s_i \in \mathcal{S}_i} u_i(s_{-i}, s_i)$$

Definition

A payoff profile $r = (r_1, r_2, \ldots, r_n)$ is *feasible* if there exist rational, nonnegative values α_a such that for all i, we can express r_i as $\sum_{a \in \mathcal{A}} \alpha_a u_i(a)$, with $\sum_{a \in \mathcal{A}} \alpha_a = 1$.

Nash Strategies in Repeated Games

Theorem (Folk Theorem)

Consider any n-player normal-form game G and any payoff profile $r = (r_1, r_2, \ldots, r_n)$.

1. *If r is the payoff profile for any Nash equilibrium s of the infinitely repeated G with average rewards, then for each player i, r_i is enforceable.*
2. *If r is both feasible and enforceable, then r is the payoff profile for some Nash equilibrium of the infinitely repeated G with average rewards.*

Stochastic Games

Let's generalize the repeated games. We do not have to play the same normal-form game repeatedly. We can play different normal-form games (possibly for infinitely long time).

Definition (Stochastic game)

A *stochastic game* is a tuple $(Q, \mathcal{N}, \mathcal{A}, \mathcal{P}, \mathcal{R})$, where:

- Q is a finite set of games
- $\mathcal{N}$ is a finite set of players
- $\mathcal{A}$ is a finite set of actions, $\mathcal{A}_i$ are actions available to player i
- $\mathcal{P}$ is a transition function $\mathcal{P} : Q \times \mathcal{A} \times Q :\to [0, 1]$, where $\mathcal{P}(q, a, q')$ is a probability of reaching game q' after a joint action a is played in game q
- $\mathcal{R}$ is a set of reward functions $r_i : Q \times \mathcal{A} \to \mathbb{R}$

Stochastic Games

Similarly to repeated games we can have several different rewards (or objectives):

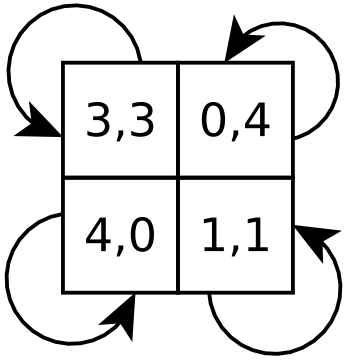
- discounted
- average
- reachability/safety

In reachability objectives a player wants to visit certain games infinitely often.

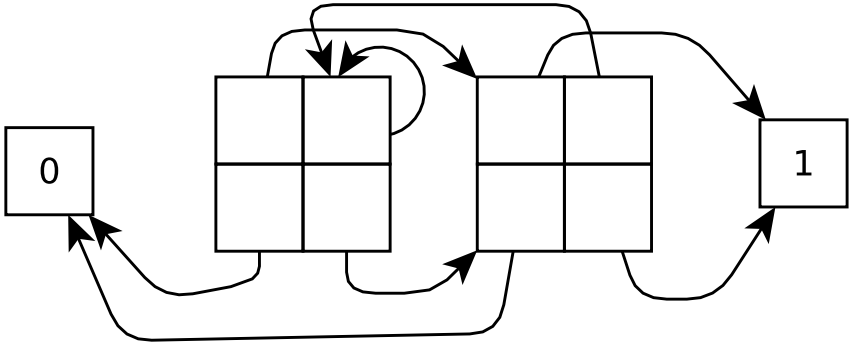
Related to reaching some target state (for example attacking a target) in a game without a pre-determined horizon.

Stochastic Games - Examples

Repeated prisoners dilemma:



Dante's purgatory:



Equilibria in Stochastic Games

Definition (History)

Let $h_t = (q_0, a_0, q_1, a_1, \ldots, a_{t1}, q_t)$ denote a history of t stages of a stochastic game, and let H_t be the set of all possible histories of this length.

Definition (Behavioral strategy)

A behavioral strategy $s_i(h_t, a_{i_j})$ returns the probability of playing action a_{i_j} for history h_t.

Definition (Markov strategy)

A Markov strategy s_i is a behavioral strategy in which $s_i(h_t, a_{i_j}) = s_i(h'_t, a_{i_j})$ if $q_t = q'_t$, where q_t and q'_t are the final games of h_t and h'_t, respectively.

Equilibria in Stochastic Games

Definition

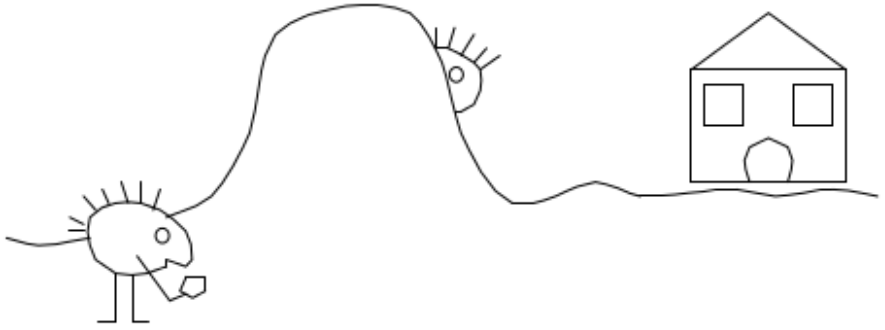
A strategy profile is called a *Markov perfect equilibrium* if it consists of only Markov strategies, and is a Nash equilibrium.

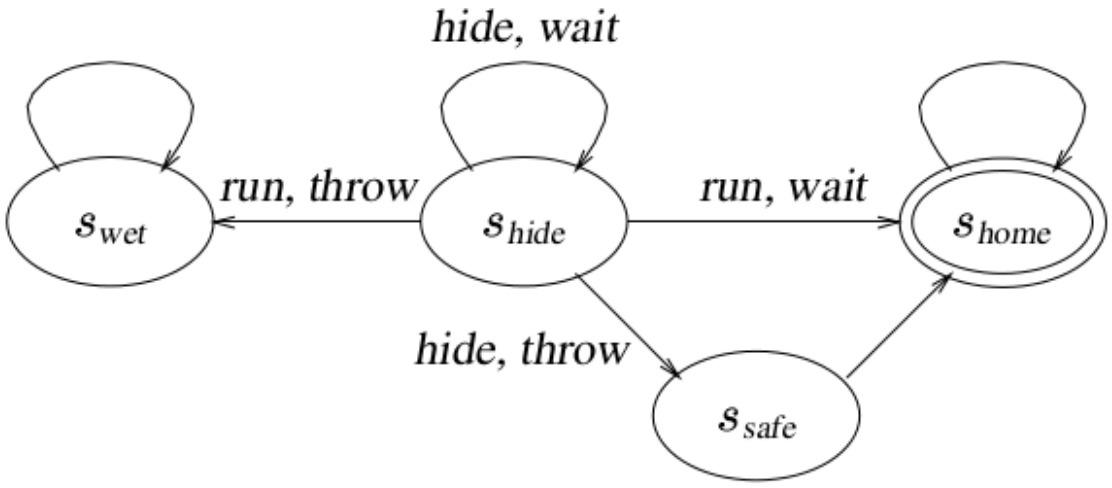
Theorem

Every n-player, general-sum, discounted-reward stochastic game has a Markov perfect equilibrium.

Equilibria in Stochastic Games

For other rewards, Markov perfect equilibrium does not have to exist.





Approximating Optimal Strategies in Stochastic Games

Standard algorithms from Markov Decision Processes, value and strategy iteration, translate to stochastic games.

Algorithm 1. Value Iteration

1: $t := 0$
2: $\tilde{v}^0 := (0, \ldots, 0, 1)$ *// the vector* $\tilde{v}^0$ *is indexed* $0, 1, \ldots, N, N+1$
3: **while true do**
4: $\quad t := t + 1$
5: $\quad \tilde{v}^t_0 := 0$
6: $\quad \tilde{v}^t_{N+1} := 1$
7: $\quad$ **for** $i \in \{1, 2, \ldots, N\}$ **do**
8: $\quad\quad \tilde{v}^t_i := \mathbf{val}(A_i(\tilde{v}^{t-1}))$

Approximating Optimal Strategies in Stochastic Games

Algorithm 2. Strategy Iteration

1: $t := 1$
2: $x^1 :=$ the strategy for Player I playing uniformly at each position
3: **while true do**
4: $\quad y^t :=$ an optimal *best reply* by Player II to x^t
5: $\quad$ **for** $i \in \{0, 1, 2, \ldots, N, N+1\}$ **do**
6: $\quad\quad v_i^t := \mu_i(x^t, y^t)$
7: $\quad t := t + 1$
8: $\quad$ **for** $i \in \{1, 2, \ldots, N\}$ **do**
9: $\quad\quad$ **if** $\mathrm{val}(A_i(v^{t-1})) > v_i^{t-1}$ **then**
10: $\quad\quad\quad x_i^t := \mathrm{maximin}(A_i(v^{t-1}))$
11: $\quad\quad$ **else**
12: $\quad\quad\quad x_i^t := x_i^{t-1}$

Invitation - Algorithmic Game Theory (XEP36AGT)

XEP36AGT – Algoritmická teorie her

This is a CW page for the course Algorithmic Game Theory (XEP36AGT).

Lectures

Date	Topic	Lecturer
18.2.2019	Introduction, Overview	Bosansky
25.2.2019	— Cancelled (Invited lecture by Yufei Han - Life of PI: Towards Trustworthy Machine) —	
4.3.2019	Nash, Fixed Point, Main Complexity Classes	Bosansky
11.3.2019	Computing and Approximating a Nash Equilibrium (Lemke Howson, MILP)	Bosansky
18.3.2019	Computing a Stackelberg Equilibrium	Bosansky
25.3.2019	Computing and Approximation of a Correlated Equilibrium	Bosansky
1.4.2019	Repeated and Stochastic Games	Bosansky
8.4.2019	Online Learning and Multiarmed Bandit Problems	Lisy
15.4.2019	Learning in Normal-Form Games, Fictitious Play	Lisy
22.4.2019	— Cancelled (Easter) —	
29.4.2019	Regret Matching, Counterfactual Regret Minimization	Lisy
6.5.2019	Continual Resolving in Extensive-Form Games (DeepStack)	Lisy
13.5.2019	Continuous Games and Their Equilibria. Separable Games.	Kroupa
20.5.2019	Polynomial Games. Reduction to an SDP Problem.	Kroupa

References I

[Bosansky et al., 2014] Bosansky, B., Kiekintveld, C., Lisy, V., and Pechoucek, M. (2014).

An Exact Double-Oracle Algorithm for Zero-Sum Extensive-Form Games with Imperfect Information.

Journal of Artificial Intelligence Research, 2014.

[Bowling et al., 2015] M. Bowling, N. Burch, M. Johanson, O. Tammelin.

Heads-up limit holdem poker is solved.

Science 347 (6218) (2015) 145–149.

[Brown and Sandholm, 2016] Brown, N. and Sandholm, T. (2016).

Strategy-Based Warm Starting for Regret Minimization in Games.

In *Proceedings of AAAI Conference on Artificial Intelligence*.

[Brown and Sandholm, 2017] Brown, N. and Sandholm, T. (2017).

Safe and Nested Subgame Solving for Imperfect-Information Games

In *Proceedings of 31st Conference on Neural Information Processing Systems (NIPS 2017)*.

References II

[Burch et al., 2014] Burch, N., Johanson, M., and Bowling, M. (2014).

Solving Imperfect Information Games Using Decomposition.

In *Proceedings of AAAI Conference on Artificial Intelligence*.

[Hoda et al., 2010] S. Hoda, A. Gilpin, J. Peña, T. Sandholm, (2010)

Smoothing Techniques for Computing Nash Equilibria of Sequential Games.

Mathematics of Operations Research 35 (2) (2010) 494–512.

[Johanson et al., 2011] Johanson, M., Bowling, M., Waugh, K., and Zinkevich, M. (2011).

Accelerating best response calculation in large extensive games.

In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI)*, pages 258–265.

[Koller and Megiddo, 1992] D. Koller, N. Megiddo. (1992)

The Complexity of Two-person Zero-sum Games in Extensive Form,

Games and Economic Behavior 4:528–552.

References III

[Kroer et al., 2018] Kroer, C., Waugh, K., Klnc-Karzan, F., Sandholm, T. (2018).

Faster algorithms for extensive-form game solving via improved smoothing functions.

Mathematical Programming, 1–33.

[Lanctot, 2013] Lanctot, M. (2013).

Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision Making in Large Extensive-Form Games.

PhD thesis, University of Alberta.

[Moravcik et al., 2017] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, M. Bowling, Deepstack: Expert-level Artificial Intelligence in Heads-up No-limit Poker, Science.

[Shoham and Leyton-Brown, 2009] Shoham, Y. and Leyton-Brown, K. (2009).

Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations.

Cambridge University Press.

References IV

[Tammelin, O. 2014] O. Tammelin, (2014)

CFR+,

CoRR, abs/1407.5042.

[Zinkevich et al., 2008] M. Zinkevich, M. Johanson, M. H. Bowling, C. Piccione. (2007)

Regret minimization in games with incomplete information.

In *Advances in Neural Information Processing Systems*, pp. 1729–1736.