Multiagent Systems (BE4M36MAS)

Solving Extensive-Form 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 13, 2018

Previously ... on multi-agent systems.

1. Extensive-Form Games
2. Transformations between representations

Solving Imperfect Information Extensive-Form Games

Backward induction does not work, there is a dependence between the information sets.

The algorithms (typically) need to consider the game as a whole:

- We can solve an EFG as a normal-form game.
- We can use so-called *sequence form* to formulate a linear program that has a linear size in the size of the game.

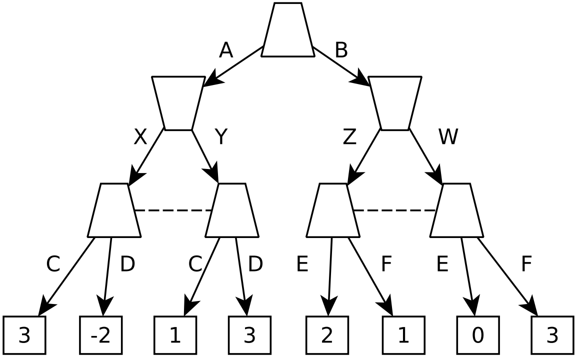
State-of-the-art algorithms:

- Double Oracle for Extensive-Form Games (DOEFG) [Bosansky et al., 2014]
- Counterfactual Regret Minimization (CFR) [Zinkevich et al., 2008, Tammelin, O. 2014]
- Excessive Gap Technique (EGT) [Hoda et al., 2010, Kroer et al., 2018]

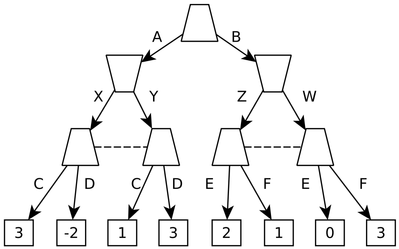
LP Algorithms for Extensive-Form Games

Algorithms based on *linear programming*

Imperfect Information EFG



Induced Normal-Form Game

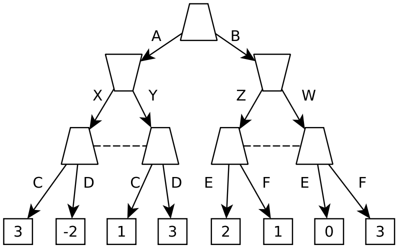


	XZ	XW	YZ	YW
ACE	3	3	1	1
ACF	3	3	1	1
ADE	-2	-2	3	3
ADF	-2	-2	3	3
BCE	2	0	2	0
BCF	1	3	1	3
BDE	2	0	2	0
BDF	1	3	1	3

Normal form representation is too verbose. The same leaf is stated multiple times in the table.

We can avoid it by using sequences.

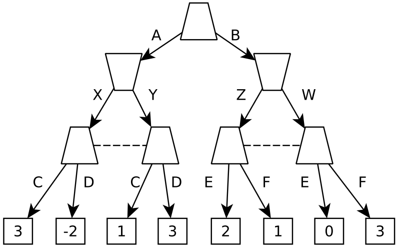
Sequences in Extensive-Form Games



Definition

An ordered list of actions of player i executed from the root of the game tree to some node $h \in \mathcal{H}$ is called a *sequence* σ_i. Set of all possible sequences of player i is denoted Σ_i.

Sequences in Extensive-Form Games

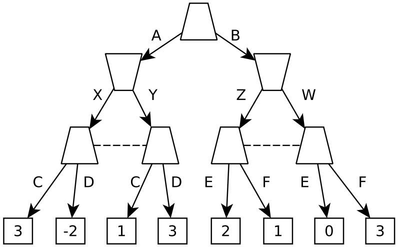


$\triangle(\Sigma_1)$	$\bigtriangledown(\Sigma_2)$
$\emptyset$	$\emptyset$
A	X
B	Y
AC	Z
AD	W
BE	
BF	

Definition

An ordered list of actions of player i executed from the root of the game tree to some node $h \in \mathcal{H}$ is called a *sequence* σ_i. Set of all possible sequences of player i is denoted Σ_i.

Extended Utility Function



$\triangle(\Sigma_1)$	$\bigtriangledown(\Sigma_2)$
$\emptyset$	$\emptyset$
A	X
B	Y
AC	Z
AD	W
BE	
BF	

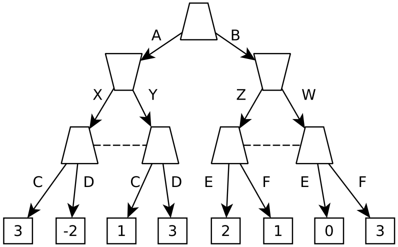
We need to extend the utility function to operate over sequences:

$$g : \Sigma_1 \times \Sigma_2 \to \mathbb{R},$$

where $g(\sigma_1, \sigma_2) =$

- $u(z)$ iff z corresponds to a leaf (terminal history) represented by sequences σ_1 and σ_2
- 0 otherwise

Extended Utility Function

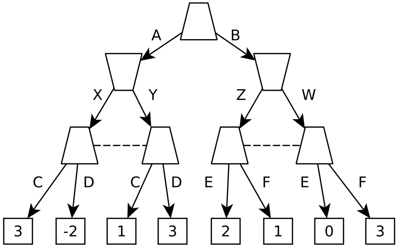


$\triangle(\Sigma_1)$	$\bigtriangledown(\Sigma_2)$
$\emptyset$	$\emptyset$
A	X
B	Y
AC	Z
AD	W
BE	
BF	

In games with chance a combination of sequences can lead to multiple nodes/leafs. $g(\sigma_1, \sigma_2) =$

- $\sum_{z \in \mathcal{Z}'} \mathcal{C}(z)u(z)$ iff $\mathcal{Z}'$ is a set of leafs that correspond to history represented by sequences σ_1 and σ_2, and $\mathcal{C}(z)$ represents the probability of leaf z being reached due to chance
- 0 otherwise

Extended Utility Function

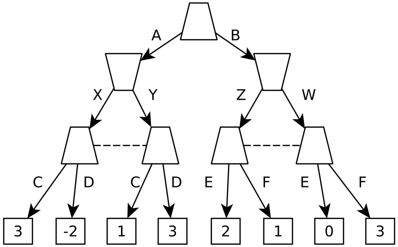


$\triangle(\Sigma_1)$	$\triangledown(\Sigma_2)$
$\emptyset$	$\emptyset$
A	X
B	Y
AC	Z
AD	W
BE	
BF	

Examples:

- $g(\emptyset, W) = 0$
- $g(AC, W) = 0$
- $g(BF, W) = 3$
- $g(A, X) = 0$
- $\ldots$

Realization Plans



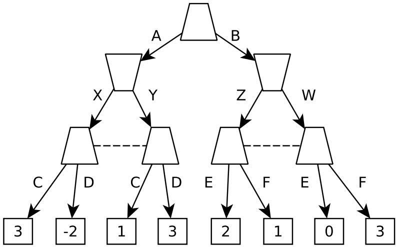
$\triangle(\Sigma_1)$	$\bigtriangledown(\Sigma_2)$
$\emptyset$	$\emptyset$
A	X
B	Y
AC	Z
AD	W
BE	
BF	

We need to express a mixed strategy using sequences. We need to be prepared for all situations.

Let's assume that the opponent (player 2) will play everything and assign a probability that certain sequence σ_1 will be played.

A *realization plan* $(r_i(\sigma_i))$ is a probability that sequence σ_i will be played assuming player $-i$ plays such actions that allow actions from σ_i to be executed.

Realization Plans



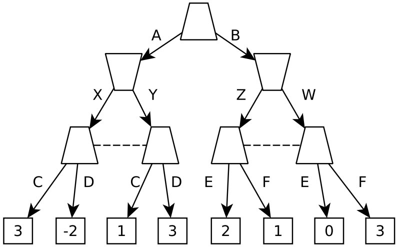
$\triangle(\Sigma_1)$	$\bigtriangledown(\Sigma_2)$
$\emptyset$	$\emptyset$
A	X
B	Y
AC	Z
AD	W
BE	
BF	

Examples:

- $r_1(\emptyset) = 1$
- $r_1(A) + r_1(B) = r_1(\emptyset)$
- $r_1(AC) + r_1(AD) = r_1(A)$
- $r_1(BE) + r_1(BF) = r_1(B)$

- $r_2(\emptyset) = 1$
- $r_2(X) + r_2(Y) = r_2(\emptyset)$
- $r_2(Z) + r_2(W) = r_2(\emptyset)$

Best Response



$\triangle(\Sigma_1)$	$\bigtriangledown(\Sigma_2)$
$\emptyset$	$\emptyset$
A	X
B	Y
AC	Z
AD	W
BE	
BF	

- We now have almost everything – a strategy representation and an extended utility function.
- We will have a maximization objective and need a best response for the minimizing player.
- A player selects the best action (the one that minimizes the expected utility) in each information set.
- An expected utility after playing an action in an information set corresponds to a sum of (1) utility values of leafs and (2) information sets that are immediately reached.

Sequence Form Linear Program (SQF)

We are now ready to state the linear program:

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

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

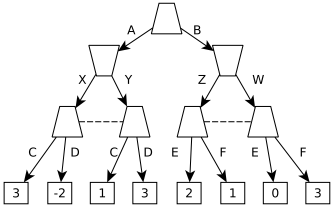
$$0 \leq r_1(\sigma_1) \leq 1 \qquad \forall \sigma_1 \in \Sigma_1 \tag{3}$$

$$\sum_{a \in \mathcal{A}(I_1)} r_1(\sigma_1 a) = r_1(\sigma_1) \quad \forall \sigma_1 \in \Sigma_1, \forall I_1 \in \mathsf{inf}_1(\sigma_1) \tag{4}$$

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

- $\mathsf{seq}_i(I)$ 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



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

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

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

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

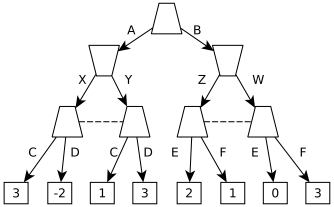
$$v(\mathsf{inf}_2(X)) \leq 0 + g(AC, X)r_1(AC) + g(AD, X)r_1(AD) \tag{10}$$

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

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

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

Sequence Form LP - Example



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

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

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

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

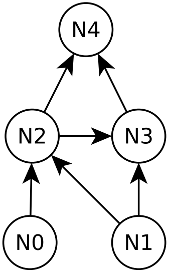
$$v(\mathsf{inf}_1(AC)) \geq g(AC, X) r_2(X) + g(AC, Y) r_2(Y) \tag{18}$$

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

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

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

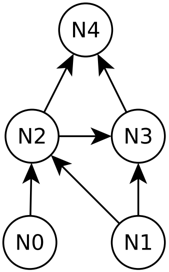
Simple Network Security Scenario – Flip-It Game



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

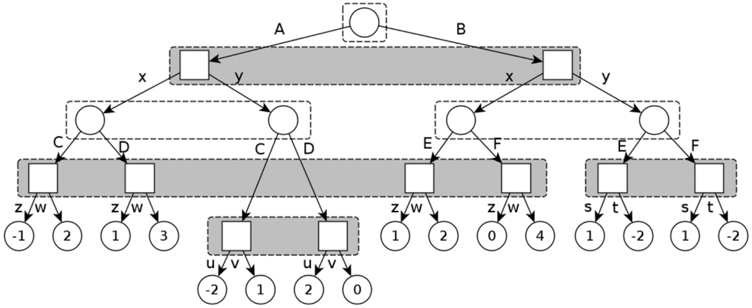
Incremental Strategy Generation

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.

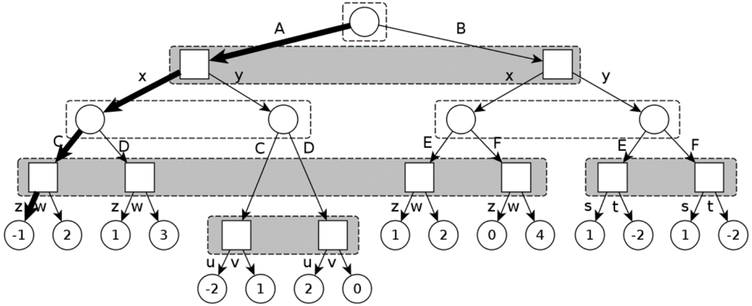
Double Oracle Algorithm for EFGs

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



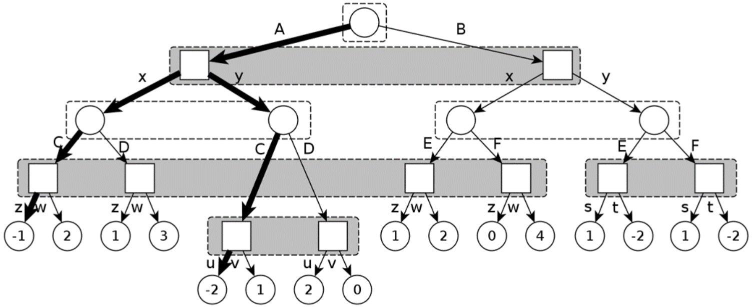
Double Oracle Algorithm for EFGs

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



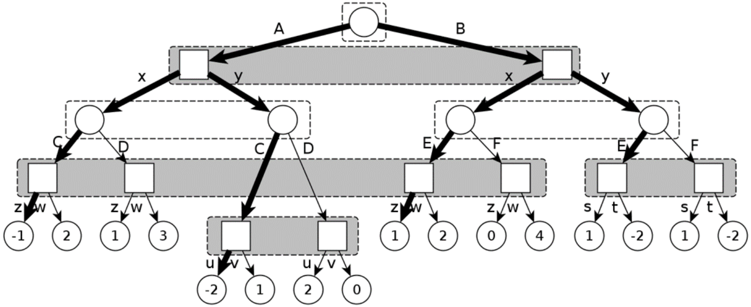
Double Oracle Algorithm for EFGs

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



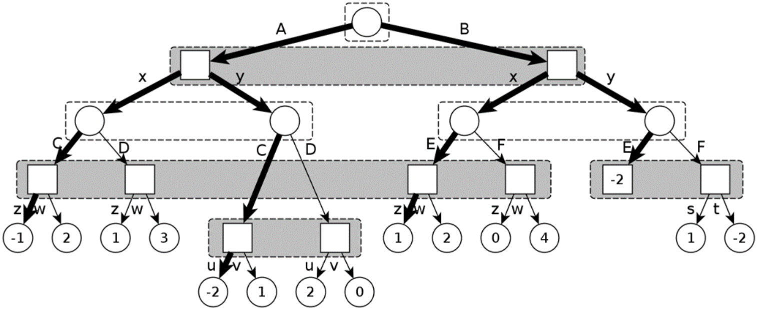
Double Oracle Algorithm for EFGs

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



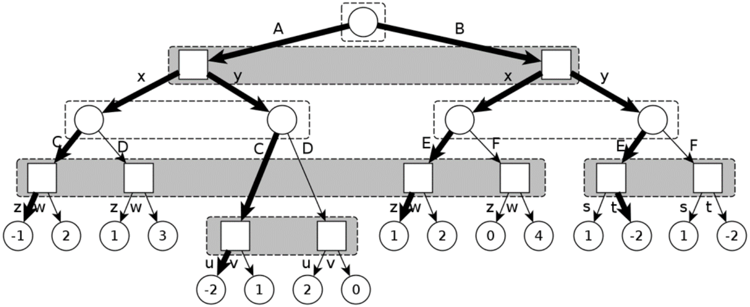
Double Oracle Algorithm for EFGs

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*

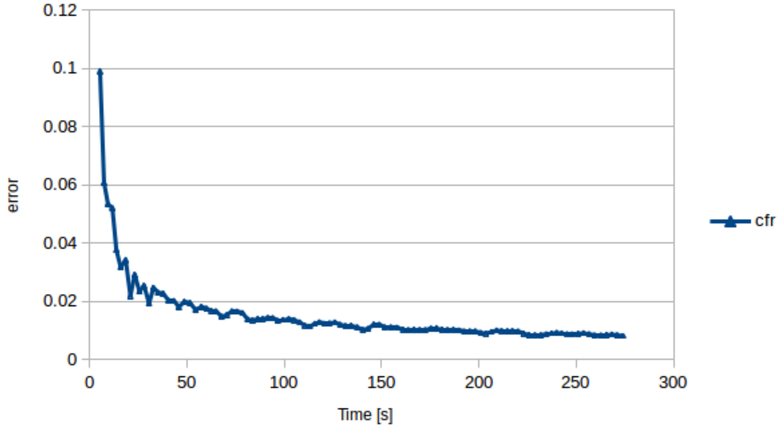
$$R_i^T = \sum_{t=1}^{T} r_i^t(I, a), \qquad R_i^{T,+}(I, a) = \max\{R_i^T(I, a), 0\}$$

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

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

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+)

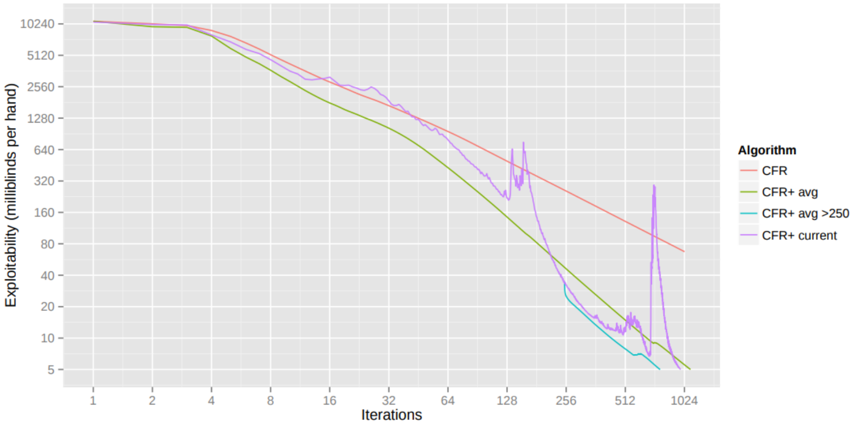


Figure 2: No Limit Texas Hold'em flop subgame [2]

[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?

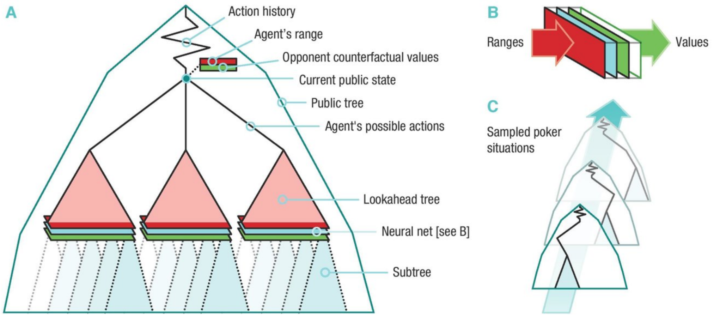
Recently, new methods that allow limited-lookahead algorithm for imperfect information games for poker
[Moravcik et al., 2017, Brown and Sandholm, 2017].

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))

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.