Surprising strategies obtained by stochastic optimization in partially observable games

Marie-Liesse Cauwet, Olivier Teytaud

Surprising strategies obtained by stochastic optimization in partially observable games

Marie-Liesse Cauwet
Mines Saint-Etienne, Univ Clermont Auvergne,
CNRS, UMR 6158 LIMOS, Institut Henri Fayol, Departement GMI,
F - 42023 Saint-Etienne France
marie-liesse.cauwet@emse.fr

Olivier Teytaud
TAO
Inria Saclay
Gif-Sur-Yvette, France
olivier.teytaud@inria.fr

***Abstract*—This paper studies the optimization of strategies in the context of possibly randomized two players zero-sum games with incomplete information. We compare 5 algorithms for tuning the parameters of strategies over a benchmark of 12 games. A first evolutionary approach consists in designing a highly randomized opponent (called naive opponent) and optimizing the parametric strategy against it; a second one is optimizing iteratively the strategy, i.e. constructing a sequence of strategies starting from the naive one. 2 versions of coevolutions, real and approximate, are also tested as well as a seed method. The coevolution methods were performing well, but results were not stable from one game to another. In spite of its simplicity, the seed method, which can be seen as an extremal version of coevolution, works even when nothing else works. Incidentally, these methods brought out some unexpected strategies for some games, such as Batawaf or the game of War, which seem, at first view, purely random games without any structured actions possible for the players or Guess Who, where a dichotomy between the characters seems to be the most reasonable strategy. All source codes of games are written in Matlab/Octave and are freely available for download.**

I. Introduction

In game theory, the case of adversarial problems where two agents try to optimize antagonist rewards under incomplete information is an important class of problems with real-life applications in games, in robust optimization and in military applications such as mission planning [1].

In a game, if the number of deterministic strategies (called pure strategies) is finite, some stochastic strategies (called mixed strategies or arms) are usually represented by vectors of non-negative numbers summing to 1: a player following a stochastic strategy $x = (x_1, \ldots, x_N)$ will adopt the i^{th} deterministic strategy with probability x_i. Defining optimality is delicate in such a setting. We might have x performing better than x' against y, but x' performing better than x against y'. A classical approach is to find an optimum in the Nash sense [2], [3]. In the case of finite and moderate number of pure strategies, the literature proposes some efficient solutions [4], [5], [6].

We are focusing on real world partially observable games for which computing an approximate Nash equilibrium requires heavy computations [7], [8], [9] and computing an exact Nash equilibrium is undecidable [10], [11] when the horizon is unbounded. Indeed, there are infinitely many pure strategies, so that the existence of a Nash equilibrium is not guaranteed.

In this framework, we propose a simpler but more practical approach: we will consider some parametric strategies. If the parameters take a finite number of values, this boils down to the classical multi-armed bandits problem [12]. Here, we consider some parameters taking values in an interval of $\mathbb{R}$ (hence infinitely many arms) and optimize them against a baseline [13].

We implemented 5 methods to optimize a parametric strategy: a naive evolutionary algorithm, a more sophisticated (iterative) one, 2 variants of coevolution and a seed method. The naive approach consists in first designing a very randomized baseline, covering "approximately equally" all the state space and then optimizing a parametric strategy against it. The more elaborate approach is to optimize iteratively a strategy π_{n+1} against π_n. Coevolution is a different approach [14], [15]. A population P_1 of agents evolves for the role of agent 1, and another population P_2 evolves for the role of agent 2. Population P_1 is optimized against population P_2, whereas population P_2 is optimized against population P_1. The seed method is an extreme case of coevolution without crossover or mutation (just random generation and selection at the end) - generate a population as large as you can for statistically analyzing it and just pick up the best. Section II describes these methods.

We first compare these methods on various games (presented in Section III) in Section IV. Then we use the obtained optima in Section V to determine some specific, detailed strategies in games where it is not so obvious that strong strategies exist (a strategy is strong if the probability of winning - when using it against a fixed set of other strategies - is significantly higher than 50%):

- Batawaf and War (card games) for which most people believe that there is no clever strategy;
- Guess who, for which most people believe that dichotomy is the best strategy;
- Battleship, where it is not so obvious that there is something beyond the Monte-Carlo estimation of hitting probabilities.

II. Algorithms

A. Preliminaries: Bernstein race as a comparison operator

Let us assume that we wish to compare two players, p_a and p_b. There are several possibilities.

In the simplest method, we play 101 games between p_a and p_b and select the one which wins at least 51 games. Selection can go wrong: if the two players have close levels, the weakest player can be selected, with probability arbitrarily close to 50%.

Let us therefore introduce a statistical test. A simple statistical test consists in playing games between p_a and p_b, and perform statistical tests until one of them has won significantly more games than the other. A statistical test has a risk of failure δ. If 20 tests are performed, then the risk that at least one test fails is $1-(1-\delta)^{20} \approx 20\delta$ for δ small enough (by Taylor expansion). Therefore, without any correction, the naive statistical test method might fail with probability arbitrarily close to 50%.

An alternative is to play games between p_a and p_b, and perform statistical tests until one of them has won significantly more games than the other, taking into account the Bonferroni correction [16] or other tests taking into account the multiple nature of these tests. This is the *Bernstein race* [17]. Races take care of correcting the statistical test, in order to guarantee a given error rate δ. Whatever are the two players, the probability that the weakest is selected is at most δ. A drawback is that it might take a lot of time, and if the two players have exactly the same level (in the sense that the probability of winning is 50%), then there is a probability $1-\delta$ that the race never halts. We then apply a *limited race*: this is the same as for Bernstein races, except that when the winning rate is known up to a given precision (0.01 in our experiments), the race is stopped and the winner is the current best.

B. Naive evolutionary algorithm

A first method, termed naive (see Alg. 1) consists in simply optimizing the average performance against a baseline, for example the default strategy using random independent standard Gaussian parameters.

C. Iterative evolutionary algorithm

A second simple intuitive algorithm is to accept a search point as a new baseline as soon as it is statistically better (winning rate $> 50\%$) than the previous one. Many evolutionary algorithms are comparison-based and can be adapted easily to that framework. We can for example apply the $(1+1)$ Evolution Strategy[1] [18] with games as a comparison operator, as proposed in Alg. 2.

At first view, this algorithm looks better than the naive one. We compare against a stronger opponent, so winning rates should be more informative. A key question is however whether we might have a *red queen effect*: algorithms winning against the previous generation, but becoming very specialized, and unable to compete with even simple tools [19], [20].

[1]Note that here, and only here, the denomination "strategy" refers to a type of evolutionary algorithm. In the rest of the paper, "strategy" denotes the player game plan, hence there is no ambiguity.

Algorithm 1 Naive algorithm. $\mathcal{N}(a,b)$ denotes a Gaussian variable of mean a and standard deviation b. x_i denotes the i^{th} component of x.

Input: a game simulator, a precision parameter ϵ, parameters of the random opponent x^0
$\sigma \leftarrow 1$ ▷ Initial step-size
$x \leftarrow x^0$ ▷ Initial strategy
while (termination criterion is not met) **do**
  for $i = 1$ to length of x **do**
    $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$ ▷ Componentwise Mutation
  end for
  repeat
    play a game:
- between x and x^0
- between x' and x^0

  until the limited Bernstein race of precision ϵ stops
  if x' performed better than x against x^0 **then**
    $x \leftarrow x'$
    $\sigma \leftarrow 2\sigma$
  else
    $\sigma \leftarrow 0.84\sigma$
  end if
end while
Output: an approximation x of the optimal strategy

Algorithm 2 Iterative approach. $\mathcal{N}(a,b)$ denotes a Gaussian variable of mean a and standard deviation b. x_i denotes the i^{th} component of x.

Input: a game simulator and a precision parameter ϵ, parameters of the initial opponent x
$\sigma \leftarrow 1$ ▷ Initial step-size
while (termination criterion is not met) **do**
  for $i = 1$ to length x **do**
    $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$ ▷ Componentwise Mutation
  end for
  repeat
    play games between x' and x
  until the limited Bernstein race of precision ϵ stops
  if x' better than x **then**
    $x \leftarrow x'$
    $\sigma \leftarrow 2\sigma$
  else
    $\sigma \leftarrow 0.84\sigma$
  end if
end while
Output: an approximation x of the optimal strategy

D. Coevolution

A typical coevolution considers Black players and White players (in case of a game with a Black player and a White player), and the fitness of the Black player is the average score against the White population, while the fitness of a White player is the average score against the Black population. A classical technique (e.g. applied in [5]) considers "extended", "doubled" players, who have two parts: one for playing as Black, and one for playing as White. Then, we need only one population. This is applied in the present paper.

In the first coevolution variant, called *real coevolution*, a search point is accepted in the population if it is statistically better than *every* point of this population. It is presented in Alg. 3.

Algorithm 3 Real coevolution. $\mathcal{N}(a,b)$ denotes a Gaussian variable of mean a and standard deviation b. x_i denotes the i^{th} component of x.

Input: a game simulator and a precision parameter ϵ, parameters of the initial opponent x.
$\sigma \leftarrow 1$ ▷ Initial step-size
$P \leftarrow \{x\}$ ▷ Best points population
while (termination criterion is not met) **do**
  for $i = 1$ to length of x **do**
    $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$ ▷ Mutation
  end for
  repeat
    play a game between x' and every point of P
  until each limited Bernstein race of precision ϵ stops
  if x' performed better than all points in P **then**
    $x \leftarrow x'$
    $P \leftarrow \{P, x'\}$
    $\sigma \leftarrow 2\sigma$
  else
    $\sigma \leftarrow 0.84\sigma$
  end if
end while
Output: an approximation x of the optimal strategy

We might consider the population as a representation of a stochastic strategy, i.e., the global strategy is randomly drawn in the population for each game the artificial intelligence had to play: this is the so-called "Parisian approach" [21]. Our second variant of coevolution, called *approximate coevolution* (see Alg. 4), is related to this approach. Instead of comparing the search point to each point of the population, it is only compared to one point of the population drawn at random.

Algorithm 4 Approximate coevolution. $\mathcal{N}(a,b)$ denotes a Gaussian variable of mean a and standard deviation b. x_i denotes the i^{th} component of x.

Input: a game simulator and a precision parameter ϵ, parameters of the initial opponent x.
$\sigma \leftarrow 1$ ▷ Initial step-size
$P \leftarrow \{x\}$ ▷ Best points population
while (termination criterion is not met) **do**
  for $i = 1$ to length of x **do**
    $x'_i \leftarrow x_i + \sigma\mathcal{N}(0,1)$ ▷ Componentwise Mutation
  end for
  Draw at random an integer $rand$ between 1 and the size of P
  repeat play a game between x' and the $rand^{th}$ individual of P
  until the limited Bernstein race of precision ϵ stops
  if x' performed better **then**
    $x \leftarrow x'$
    $P \leftarrow \{P, x'\}$
    $\sigma \leftarrow 2\sigma$
  else
    $\sigma \leftarrow 0.84\sigma$
  end if
end while
Output: an approximation x of the optimal strategy

E. Method of seeds

[22], [23], [24], [25], [26] proposed a new method for optimizing an artificial intelligence, which can be applied to our setting. The principle is to generate a population of K random individuals for the first player, and another population of K random individuals for the second player. Then, we consider all pairwise games (a more subtle method was proposed in [25] but we do not consider this in the present paper). We then select the player with best average performance. Both populations can be equal when the game is symmetric, and we might increase K until the time budget is elapsed in order to be anytime.

III. Games

The games are fully described in our open source platform (https://gforge.inria.fr/projects/gametestbed/). We provide a short description below.

A. Brief description

Batawaf and game of War: The cards, kept unknown from the two players, are randomly distributed among them. At each turn, each player reveal simultaneously the card at the top of their deck. The player with the highest valued card wins the cards and put them at the bottom of his/her deck. In case of draw, the two players place a second card, face down, on the pile, then a third one face up. The process is possibly repeated until the winner of the turn can be determined. The player who gets all the cards wins the game. The difference between Batawaf and War is the total number of cards and in the number of cards of same strength.

Battleship: Two players place secretly four ships of size 2, 3, 4 and 5 on a 9×9 board without crossing, horizontally or vertically. In each round, each player in turn propose a position on the board, trying to find out a ship of the adversary. The first player to have found all parts of all ships of her/his opponent wins. At each turn the player is informed of whether he has touched something.

Cheat: The cheat game uses 52 cards of 4 different colors. The cards are randomly distributed among the players. One of the players puts a first card on the table. The color of this card is termed the "current color". Then, each player, in turn, either puts a card (face hidden) on top of the previous one[2] or claims "cheat". In the latter case, if the last played card was of the current color, the person who claimed "cheat" receives all the cards currently on the table. Otherwise, the previous player receives all the cards currently on the table. In both cases, a new round start, with the player who received all the cards putting a chosen card, which decides the new current color. The first player who gets rid of all his/her cards wins the game.

Flip: In the flip game, cards are randomly distributed among the two players. Only the five topmost cards are visible (for both players). When a card is removed, another one is made visible. Both players, in turn[3], put one of their cards (if they can) on top of one of the two visible cards on the table. A card may be put on top of the card if it is the card just above or just below in the ranking - except that the strongest card can be put on the weakest and the weakest can be put on the strongest. The first player who gets rid of all his cards wins the game.

[2](s)he can choose the color of the card.

[3]The game is also played as a speed game, with players playing as soon as they can instead of playing in turn.

Guess who: Each player privately draws a number between 1 and 128. At each round, in turn, each player asks the other if his/her number is in the first x numbers (the player chooses x). The answer must be sincere, so that some possibilities can be can removed. The first player who determines the number chosen by the other wins.

Morra: In the Morra game, both players simultaneously announce (i) a number of fingers (usually by showing with the hand) between 0 and 5 and (ii) a guessed number between 0 and 10. If the sum of the numbers of fingers is equal to the guess of one (and only one) player, then this player wins the game (otherwise, the game is a draw).

Phantom 4 in row: It is the stochastic version of 4 in row. In a 4x7 grid, at each turn, the players choose a permutation of the 7 locations and a move, without observation of the game. If the chosen location is already full, the stone goes to the next location according to the permutation. The first player with 4 stones aligned wins.

Phantom tic tac toe: The tic tac toe game, in 3x3, is well known and not very challenging. In the phantom version, a permutation of the 9 locations is chosen. A player cannot see her opponent's moves. Whenever a player chooses a location already occupied, the move is switched to the next location in the order of the permutation. Finding a strong strategy in Phantom tic tac toe is far less trivial than for tic tac toe.

Other games: Other games in the platform are not partially observable: Pig is a dice game, Nim (a.k.a. Marienbad game) is well known as a misery game; four-in-a-row is famous. For these games our approaches are not likely to be meaningful and these results are just included for shedding light on the genericness of our methods.

B. Parametric strategies

The strategies will not be detailed here. They can be downloaded at http://www.lri.fr/∼teytaud/gametestbed.html. They are roughly described in Table I.

IV. Results

Scores against the baseline: Table II presents the score of each method against the original default parametrization. Each learning is performed only once; methods in which no iteration was obtained are not displayed. Each method was run 24 hours. Seed was not run for "Guess Who" variants.

Comparison between 4 optimization methods: We now compare strategies obtained by our different methods against each other rather than against the original baseline. Numerical results are displayed in Table III. All results are obtained after 24 hours of learning.

V. Detailed strategies

This section studies some specific games, and shows that our parametric strategies could find out some surprisingly strong strategies against the baselines. Some of these strategies can make you really strong at Guess who (which was already known, but we provide human readable strategies) and at Batawaf or War (which is more unexpected).

A. Batawaf and game of War: good cards first!

The rules do not specify in which order the cards won by a player should be added at the end of the deck. This order has an impact on the length of games, and on the probability that the game finishes: loops are possible, see [28]. We here stop the game and consider it as a draw after 1 million moves without conclusion.

Considered strategies: We consider 4 parameters, namely A, B, C, D, which are converted into non-negative parameters using $a = \exp(A)$, $b = \exp(B)$, $c = \exp(C)$, $d = \exp(D)$. Then, when we must put k won cards in our deck (at the end), their order is chosen as follows:

1) d is used as a seed for generating a permutation π of $\{1, 2, \ldots, k\}$;
2) σ is a permutation of $\{1, 2, \ldots, k\}$:
   - uniformly drawn with probability $a/(a + b + c)$;
   - equal to $\{1, 2, \ldots, k\}$ with probability $b/(a + b + c)$;
   - equal to $\{k, k-1, \ldots, 1\}$ with probability $c/(a+b+c)$;
3) the cards are put in ascending order and shuffled using permutation $\pi \circ \sigma$.

Results & discussion : For some specific values, we therefore get the following human-readable strategy: put the cards at the end of your deck in decreasing order, the best card first. This strategy is termed "descending" in the rest of this paper. The naive strategy consists in randomly sorting the card. The ascending strategy is the opposite of the descending one. An alternate version with 3 parameters uses only A, B, C and the identity for π. Its optimization was much faster, immediately converging to the use of descending order, our best strategy so far.

In batawaf, the descending strategy gets winning rate 68.8% against the naive strategy, 57.5% against the ascending strategy, and 50% against itself (tested on 10000 games). In War, the corresponding performances are 70.0%, 53.3% and 50%.

B. Guess who: play risky moves when late!

In many cases, we have an intuitive understanding of what is a "risky" move, and more precisely a "good risky move". It is a move which, simultaneously, maximizes the expected long term reward, and has a large probability of making, at a tactical level, the situation worse than if we had played a simple move.

In games, choosing the right level of risk depends on the current situation. If the goal is to win, then, in a territory game such as Go, ensuring an almost sufficient territory is meaningless; it is better to have a 5% probability of having more territory than your opponent, rather than a probability 100% of having just a little bit less territory than your opponent. Therefore, the "shobute" (http://senseis.xmp.net/?Shobute) principle states that, if you are behind in a game of Go, it is time for a risky attack. Another case occurs when playing a sequence of games, with, e.g., half a point in case of draw, one point in case of win, and zero in case of loss. Then, the player in advance might prefer to play very safely, ensuring a draw, whereas the other must play risky. Here, we investigate the case of Guess

Game	Number of parameters	Info	Stochasticy of the strategies	The optimal strategy must be stochastic
War	3/4	Probability of selecting the ascending order, prob. of selecting the descending order, probability of random order, perturbation of the selected ranking, see Section V-A	No/Yes	?
Batawaf	3/4	idem War	idem	idem
Battleship	33	Parameters of the probability distribution for the ships	Yes	Yes
Cheat	28/29	Parameters of a randomly perturbed quadratic value function; the 29^{th} optional parameter (variant called "special") is an arbitrary perturbation of the value function, built pseudo-randomly from that parameter	Yes	Yes
Flip	420	Parameters of a quadratic value function	No	No
Guess who (deterministic/stochastic)	4/5	Nonlinear expert strategy based on a risk level based on the gap (see [27] and Section V-B)	No/Yes	No
Morra	66	Vector of probabilities for all possible joint actions	Yes	Yes
Nim	383	Complete value function over the 383 non-terminal states	No	No
Phantom 4-in-a-row	2	Complete pure strategy (i.e. index of the chosen list of moves when playing first and index of the chosen list of moves when playing second)	No	?
Phantom tic-tac-toe	18	Complete pure strategy	No	Yes
4-in-a-row	14	Parameter of the Monte-Carlo strategy	Yes*	No

TABLE I: Characteristics of the parametric strategies for each game: number of parameters and stochasticity or not of the strategies. * denote "yes" for strategies which are stochastic only by the finiteness of samples and/or draws in Monte-Carlo simulations.

Game	Method	Score against baseline
4-in-a-row	Seed	0.63 ± 0.03
	Iterative	1 ± 0
	Naive	0.75 ± 0.19
	Real coevol	0.442308 ± 0.10
Batawaf4	Seed	0.606 ± 0.004
	Coevol	0.626 ± 0.005
	Iterative	0.626 ± 0.004
	Naive	0.620 ± 0.004
	Real coevol	0.544 ± 0.003
Batawaf	Seed	0.627 ± 0.004
	Coevol	0.626912 ± 0.005
	Iterative	0.626671 ± 0.004
	Naive	0.628467 ± 0.004
	Real coevol	0.443975 ± 0.003
Cheat	**Seed**	0.845 ± 0.003
	cheat, Coevol	0.783318 ± 0.006
	Iterative	0.817416 ± 0.029
	Naive	0.802905 ± 0.011
	Real coevol	0.810882 ± 0.007
Flip	Seed	0.554 ± 0.04
	Coevol	0.597727 ± 0.017
	Iterative	0.588 ± 0.022
	Naive	0.593143 ± 0.017
	Real coevol	0.591314 ± 0.016

Game	Method	Score against baseline
Guess who-deter	**Coevol**	0.785714 ± 0.113804
	Naive	0.613484 ± 0.000513042
	Real coevol	0.597514 ± 0.00160714
Guess who	Coevol	0.679± 0.130
	Iterative	0.771± 0.0876
	Naive	1 ± 0
	Real coevol	0.615 ± 0.00158
Cheat-Special	Seed	0.6405 ± 0.02
	Coevol	0.64 ± 0.016
	Iterative	0.58 ± 0.01
	Naive	0.65 ± 0.027
	Real coevol	0.62 ± 0.027
Battleship	**Seed**	0.77
Morra	Seed	0.519 ± 0.0013
	Coevol	0.526 ± 0.0012
	Iterative	0.527 ± 0.0016
	Naive	0.523± 0.001
	Real coevol	0.518 ± 0.001
Nim	**Seed**	0.74 ± 0.0003
	Coevol	0.500 ± 0.0007
	Iterative	0.589 ± 0.002
	Real coevol	0.669 ± 0.002

Game	Method	Score against baseline
Phantom-4-in-a-row	**Seed**	0.69 ± 0.003
	Coevol	0.549± 0.00096
	Iterative	0.657± 0.004
	Real coevol	0.598 ± 0.003
Phantom-tic-tac-toe	**Seed**	0.75 ± 0.0012
	Coevol	0.445 ± 0.000497
	Iterative	0.595 ± 0.00138
	Naive	0.445 ± 0.000497
	Real coevol	0.696 ± 0.0011
Pig	Seed	0.505 ± 0.003
	Coevol	0.507889 ± 0.0012
	Iterative	0.522151 ± 0.002
	Naive	0.512262 ± 0.0016
	Real coevol	0.508092 ± 0.0016
War4	Seed	0.623 ± 0.017
	Coevol	0.623288 ± 0.016
	Iterative	0.669± 0.05
	Naive	0.605± 0.018
	Real coevol	0.553± 0.010
War	Seed	0.632 ± 0.003
	Coevol	0.517 ± 0.00666
	Iterative	0.653 ± 0.012
	Naive	0.632 ± 0.018
	Real coevol	0.487 ± 0.005

TABLE II: Scores (winning rates) of each method against the original default parametrization. The standard deviation is shown after ±. Despite the fact that the naive method is directly optimizing the target criterion, other methods are competitive. The seed method is particularly robust. Batawaf4 (resp. War4) denotes the Batawaf (resp. War) strategy with 4 parameters.

who and show that, contrarily to what is usually assumed, the optimal strategy must consider the necessity of taking risks.

Let us discuss the Guess who game. First, for any arbitrary subset of your characters, you can design a question which discriminates exactly this subset. This can be done by using long questions ("is your character Tom or Jean or Pierre or Chang-Shing?"), or the alphabetic order. Randomizing the permutation, we get a strategy which is invariant by permutation of the characters. As a consequence, the order and traits do not matter. We can therefore work in a permutation-invariant manner, and just split between the c first and the $n-c$ last characters, when we have a list of n characters. $c<1$ or $c>n-1$ do not make any sense; therefore we have $n-2$ possible actions (up to symmetries) when we have n characters. Using these symmetries, the state of our set of characters become simply a number between 1 and N, where N is the original number of characters. This state is just the number of remaining characters. We can use the number of characters in the opponent's set as well, as a side information; we might want to make more risky decisions (farther from equal split) when we have a bad situation. This is precisely the point in this paper. As a consequence, up to symmetries (permutations of our characters and of the opponent's characters) our strategy uses as an input the pair (n,m) (n is our number of characters and m is the number of characters of our opponent) and outputs $1<c<n-2$.

A simple first remark is that dichotomy (constructing a question such that approximately half characters correspond to "yes") is a reasonably good strategy. As pointed out in boardgamegeek.com/thread/302791/advanced-strategies, when you are late, you should not apply the dichotomy otherwise you are just going to lose. Instead, you should use risky strategies, with which you might win early. This reference also considers hiding the current size of the set of characters. They however do not provide any experiment or formalization of strategies.

Cheat (28 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.0269(+-0.003)	0.017(+-0.002)	0.977(+-0.002)
iterative	0.973(+-0.003)	0.5(+- 0)	0.5(+-0.008)	0.502(+-0.008)
approx. coevol	0.983(+-0.002)	0.5(+-0.008)	0.5(+- 0)	0.671(+-0.008)
real coevol	0.0235(+-0.002)	0.498(+-0.008)	0.329(+-0.008)	0.5(+- 0)
GuessWho-deter (4 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.8(+-0.004)	0.489(+-0.005)	0.498(+-0.005)
iterative	0.2(+-0.004)	0.5(+- 0)	0.208(+-0.004)	0.14(+-0.004)
approx. coevol	0.511(+-0.005)	0.792(+-0.004)	0.5(+- 0)	0.502(+-0.005)
real coevol	0.502(+-0.005)	0.86(+-0.004)	0.498(+-0.005)	0.5(+- 0)
GuessWho (5 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.276(+-0.005)	0.159(+-0.004)	0.34(+-0.005)
iterative	0.724(+-0.005)	0.5(+- 0)	0.424(+-0.005)	0.583(+-0.005)
approx. coevol	0.841(+-0.004)	0.576(+-0.005)	0.5(+- 0)	0.563(+-0.005)
real coevol	0.66(+-0.005)	0.417(+-0.005)	0.437(+-0.005)	0.5(+- 0)
Phantom-4-in-a-row (2 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.5(+-0.005)	1(+- 0)	0.75(+-0.004)
iterative	0.5(+-0.005)	0.5(+- 0)	1(+- 0)	0(+- 0)
approx. coevol	0(+- 0)	0(+- 0)	0.5(+- 0)	0.5(+-0.005)
real coevol	0.25(+-0.004)	1(+- 0)	0.5(+-0.005)	0.5(+- 0)
Nim (383 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.5(+-0.005)	0(+- 0)	0.5(+-0.005)
iterative	0.5(+-0.005)	0.5(+- 0)	0(+- 0)	0(+- 0)
approx. coevol	1(+- 0)	1(+- 0)	0.5(+- 0)	0(+- 0)
real coevol	0.5(+-0.005)	1(+- 0)	1(+- 0)	0.5(+- 0)
Phantom-tic-tac-toe (18 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.75(+-0.004)	0.5(+-0.005)	0.5(+-0.005)
iterative	0.25(+-0.004)	0.5(+- 0)	0(+- 0)	0.5(+-0.005)
approx. coevol	0.5(+-0.005)	1(+- 0)	0.5(+- 0)	0(+- 0)
real coevol	0.5(+-0.005)	0.5(+-0.005)	1(+- 0)	0.5(+- 0)
Morra (66 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.472(+-0.005)	0.472(+-0.005)	0.477(+-0.005)
iterative	0.528(+-0.005)	0.5(+- 0)	0.475(+-0.005)	0.483(+-0.005)
approx. coevol	0.528(+-0.005)	0.525(+-0.005)	0.5(+- 0)	0.482(+-0.005)
real coevol	0.523(+-0.005)	0.517(+-0.005)	0.518(+-0.005)	0.5(+- 0)
4-in-a-row (14 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.427(+-0.05)	0.494(+-0.06)	0.503(+-0.06)
iterative	0.573(+-0.05)	0.5(+- 0)	0.494(+-0.06)	0.57(+-0.06)
approx. coevol	0.506(+-0.06)	0.506(+-0.06)	0.5(+- 0)	0.604(+-0.05)
real coevol	0.497(+-0.06)	0.43(+-0.06)	0.396(+-0.05)	0.5(+- 0)

Flip (420 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.428(+-0.005)	0.439(+-0.005)	0.439(+-0.005)
iterative	0.572(+-0.005)	0.5(+- 0)	0.437(+-0.005)	0.432(+-0.005)
approx. coevol	0.561(+-0.005)	0.563(+-0.005)	0.5(+- 0)	0.437(+-0.005)
real coevol	0.561(+-0.005)	0.568(+-0.005)	0.563(+-0.005)	0.5(+- 0)
Pig (1 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.496(+-0.005)	0.502(+-0.005)	0.505(+-0.005)
iterative	0.504(+-0.005)	0.5(+- 0)	0.506(+-0.005)	0.498(+-0.005)
approx. coevol	0.498(+-0.005)	0.494(+-0.005)	0.5(+- 0)	0.502(+-0.005)
real coevol	0.495(+-0.005)	0.502(+-0.005)	0.498(+-0.005)	0.5(+- 0)
Batawaf (3 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.49(+-0.005)	0.501(+-0.005)	0.701(+-0.005)
iterative	0.51(+-0.005)	0.5(+- 0)	0.497(+-0.005)	0.698(+-0.005)
approx. coevol	0.499(+-0.005)	0.503(+-0.005)	0.5(+- 0)	0.693(+-0.005)
real coevol	0.299(+-0.005)	0.302(+-0.005)	0.307(+-0.005)	0.5(+- 0)
War (3 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.453(+-0.006)	0.495(+-0.006)	0.508(+-0.006)
iterative	0.547(+-0.006)	0.5(+- 0)	0.563(+-0.006)	0.708(+-0.005)
approx. coevol	0.505(+-0.006)	0.437(+-0.006)	0.5(+- 0)	0.492(+-0.006)
real coevol	0.492(+-0.006)	0.292(+-0.005)	0.508(+-0.006)	0.5(+- 0)
Batawaf4 (4 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.311(+-0.005)	0.301(+-0.005)	0.385(+-0.005)
iterative	0.689(+-0.005)	0.5(+- 0)	0.507(+-0.005)	0.562(+-0.005)
approx. coevol	0.699(+-0.005)	0.493(+-0.005)	0.5(+- 0)	0.573(+-0.005)
real coevol	0.615(+-0.005)	0.438(+-0.005)	0.427(+-0.005)	0.5(+- 0)
War4 (4 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.489(+-0.008)	0.495(+-0.008)	0.493(+-0.008)
iterative	0.511(+-0.008)	0.5(+- 0)	0.424(+-0.008)	0.47(+-0.008)
approx. coevol	0.505(+-0.008)	0.576(+-0.008)	0.5(+- 0)	0.576(+-0.008)
real coevol	0.507(+-0.008)	0.53(+-0.008)	0.424(+-0.008)	0.5(+- 0)
Cheat-Special (29 params)				
	vs naive	vs itera.	vs ap. coevol	vs coevol
naive	0.5(+- 0)	0.497(+-0.01)	0.499(+-0.01)	0.5(2^{th} +-0.01)
iterative	0.503(+-0.01)	0.5(+- 0)	0.658(+-0.01)	0.5(+-0.01)
approx. coevol	0.501(+-0.01)	0.342(+-0.01)	0.5(+- 0)	0.5(+-0.01)
real coevol	0.5(+-0.01)	0.5(+-0.01)	0.5(+-0.01)	0.5(+- 0)

TABLE III: Comparison in terms of winning rate between the naive and iterative approaches and coevolution. The value after '$\pm$' is the standard deviation. The best method (if any) if the one which wins with probability at least 50% against all others. It is indicated in bold. It is possible that more than one method wins with probability at least 50% against all others (e.g., for "Phantom-tic-tac-toc", it is the case for naive and real coevolution, however, naive is better). The coevolution methods dominate in most case (in 8 games over 13).

Strategies: The dichotomy strategy is simply:

$$(n, m) \mapsto \lfloor n/2 \rfloor.$$

A general strategy, which covers the dichotomy case, is:

$$\pi_{\alpha,\beta}(n,m) = \max\left(1, \left\lfloor \beta r \frac{n-1}{2} + (1-\beta)\left(\frac{n}{2} - \alpha \frac{\max(n-m,0)}{2}\right)\right\rfloor\right), \quad (1)$$

with $\alpha \in \mathbb{R}$, $\beta \in \mathbb{R}$ and $r \in [0,1]$ drawn uniformly at random. It will turn out, however, that $\beta > 0$ is a poor strategy. We can further extend it by including non-linear terms:

$$\pi_{\alpha,\beta,\gamma}(n,m) = \max\left(1, \left\lfloor \beta r \frac{n-1}{2} + (1-\beta)\left(\frac{n}{2} - \alpha \frac{\delta_m}{2} + \gamma \frac{n}{2}\frac{\delta_m^2}{n^2}\right)\right\rfloor\right), \quad (2)$$

where $\delta_m = \max(n-m,0)/2$, $\alpha \in \mathbb{R}$, $\beta \in \mathbb{R}$, $\gamma \in \mathbb{R}$ and r as in Eq. 1.

We can consider yet another formula, using $\beta = 0$ and introducing two new parameters $\zeta \in \mathbb{R}$ and $\iota \in \mathbb{R}$, as follows:

$$\pi_{\alpha,\gamma,\zeta,\iota}(n,m) = \max\left(1, \left\lfloor \frac{n}{2} - \alpha \frac{\delta_m}{2} + \gamma \frac{n}{2}\frac{\delta_m^2}{n^2} + \zeta \frac{n}{2}\frac{\delta_m^3}{n^3} + \iota \frac{n-m}{2}\right\rfloor\right), \quad (3)$$

with $\alpha \in \mathbb{R}$, $\gamma \in \mathbb{R}$ and δ_m as in Eq. 2.

Results and discussion: Strategies of Eqs. 1, 2 and 3 are first optimized in terms of winning rate against the dichotomy strategy (4 hours of optimization). It is a simple application of the naive approach. Since it appeared that $\beta > 0$ in Eqs. 1 and 2 leads to poor strategies, we dismissed this term (i.e. $\beta = 0$). At this step, the optimal strategy then corresponds to $\alpha = 1$ and $\beta = 0$ in Eq. 1, or equivalently , $\alpha = 1$ and $\beta = 0$ and $\gamma = 0$ in Eq. 2, or equivalently $\alpha = 1$ and $\gamma = \zeta = \iota = 0$ in Eq. 3. We call this strategy *optimal linear*, shortened in OL.

Second, applying the iterative approach, we optimize a family of non-linear strategies (Eq. 3) by maximizing the winning rate against the OL strategy, using Eq. 2 under constraint $\beta = 0$ (4 hours of optimization). The optimal parameters are $\alpha = -\frac{1}{4}$ and $\gamma = -\frac{3}{2}$, $\zeta = \iota = 0$. This strategy is termed *optimal nonlinear*, denoted ONL.

Finally, in one more step of the iterative approach, a better

4-parameter strategy, termed ' Best", optimal strategy against ONL (4 hours of optimization), is obtained with $\alpha = -0.56$, $\gamma = -1.58$, $\zeta = -0.06$, $\iota = -0.022$ in Eq. 3.

[27] pointed out the existence of an analytical optimal strategy: it is used in our tests. Results are presented in Fig. 1. We see (right) that ONL wins against OL, as well as Best. Best wins against ONL (middle), but against God (the optimal strategy) only OL reaches 50%. The naive approach can be "exploited" (in the Nash sense: a strategy specialized against it will outperform it) but it is quite strong both against the baseline and against God (approximately 50%: see left).

C. Cheat game: ensuring some draw

At the cheat game, both the naive and iterative methods found a strategy with a lot of draws. Observing games, we could extract the following optimal strategy, ensuring a draw:

1) If your opponent has more than 3 cards, play an arbitrary strategy.
2) If your opponent has 2 or 3 cards, and it is your turn to choose a current color, then put a card of a color that he does not have, and say "cheat" when he plays; he has now more than 2 cards.
3) If your opponent has 3 cards and it is your turn to put a card (the current color already being set), then say "cheat". Either he has cheated and has now more than 3 cards, or not and you are now in the situation of point 2.

This strategy ensures that the opponent has never less than 2 cards.

VI. Conclusion

A. Comparing algorithms

For many games, simulating the current hidden state conditionally to partial observation is tricky, and mathematically principled algorithms for this[4] are hardly available. For such games, implementing a parametric strategy and optimizing the parameters (possibly with zero human knowledge) works: we obtained new interesting strategies in Cheat (never-losing strategy, found before humans solved it), Guess who (on par with mathematically exact method), War & Batawaf (strategy choosing the order of won cards put at the end of one's desk). In phantom cases and some others we just optimize the complete deterministic strategy, i.e., we use zero human knowledge. Inside this general principle of little or zero expertise parametric policy optimization, comparing algorithms is difficult; the conclusions are never clear and universal. We cannot claim a strong superiority of any of our methods against other ones. We might advocate a portfolio of methods, i.e., testing several methods.

For partially observable games, rankings between players are not clear. The Elo model [29] does not apply because superiority at a given game is not transitive (see the 4 strategies obtained for phantom-tic-tac-toe by the naive method, the iterative method, coevolution and approximate coevolution). Cycles make the naive and iterative method risky and sensitive to the red queen effect.

The iterative method is appealing compared to the naive one when the Elo model applies, because optimizing against a fixed baseline might become very hard, in particular when the success rate is already quite high, so that fitness evaluations are not very informative - a Bernstein race between candidates close to 100% winning rate is very slow.

Results on Guess who, including the mathematical exact optimum, show that we can compete in a couple of lines of code and a few hours of computation with a mathematical analysis (see the OL method, obtained by naive optimization against the dichotomy). However, our method can be (slightly) exploited by a method optimized specifically against it, whereas the mathematical method cannot (by definition, and our experiments confirm this) be exploited (Fig. 1).

The most surprising results for us was the successes of the seed method. This method just generates a large population, and simulates games (possibly in a more clever manner than full Round-Robin, see e.g. [25]) for choosing the best - coevolution, but with no mutation, no iteration, no crossover, just random generation and selection.

In particular, it does work when nothing works, e.g. in the battleship case (too expensive for other methods - we have time for only 54 games in this platform in the considered time setting!), or when the strategy is very poorly parametrized so that even a trivial game becomes complicated (e.g. Nim). It was never far from the best, except when it was possible to solve exactly the game with 100% winning rate against the baseline as in 4-in-a-row.

B. Results on specific games

In Batawaf and the card game of War, we provide a surprisingly strong heuristic. To the best of our knowledge, nobody ever published such strong strategies at these games, while War is one of the most widely played games in the world. We get close to 70% winning rate against the baseline that most people play. In Battleship, we provide both a probability distribution for the initial positioning of the fleet and for the Monte-Carlo simulation of the opponent's fleet. Only the seed method could find something meaningful. In Guess who, we provide a strategy which is on par with the mathematically derived optimum, which is the key output of a paper published as a mathematical study. The algorithm derived, by itself, an application of the "shobute" principle, namely we should have more risky decisions when we are late. We ran our experiments before the mathematical solution was published. At Cheat, we get a strong strategy, including against humans. This was found by computers before being understood by humans.

[4]Keep in mind the undecidability result in [10], which shows that a general algorithm for belief state information cannot exist in the general case of partially observable game.

References

[1] A. Menif, C. Guettier, and T. Cazenave, "Planning and execution control architecture for infantry serious gaming," *Planning in Games Workshop*, vol. 31, 2013.

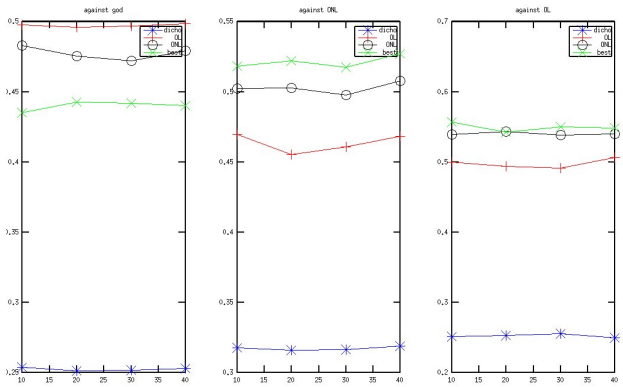

Fig. 1: Results at Guess who (winning rate on the y-axis, as a function of the initial number of characters on the x-axis). We present results of OL, ONL, Best and the simple dichotomy (which performs very poorly). Left: performance against the optimal strategy (termed "God") designed in [27]. Middle: performance against ONL. Right: performance against Dichotomy. Essentially the OL method performs well against God, though it could be slightly exploited by a method optimized against it. Interestingly methods optimized against OL were on the other hand weaker against God.

[2] J. Neumann, "Zur theorie der gesellschaftsspiele," *Mathematische Annalen*, vol. 100, no. 1, pp. 295–320, 1928. [Online]. Available: http://dx.doi.org/10.1007/BF01448847

[3] J. Nash, "Some games and machines for playing them," Rand Corporation, Tech. Rep. D-1164, 1952.

[4] B. von Stengel, "Computing equilibria for two-person games," *Handbook of Game Theory*, vol. 3, pp. 1723 – 1759, 2002.

[5] M. D. Grigoriadis and L. G. Khachiyan, "A sublinear-time randomized approximation algorithm for matrix games," *Operations Research Letters*, vol. 18, no. 2, pp. 53–58, Sep 1995.

[6] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2/3, pp. 235–256, 2002.

[7] M. Mundhenk, J. Goldsmith, C. Lusena, and E. Allender, "Complexity of finite-horizon markov decision process problems," *J. ACM*, vol. 47, no. 4, pp. 681–720, Jul. 2000. [Online]. Available: http://doi.acm.org/10.1145/347476.347480

[8] J. Rintanen, "Complexity of planning with partial observability." in *ICAPS*, 2004, pp. 345–354.

[9] D. Auger, *Applications of Evolutionary Computation: EvoApplications 2011: EvoCOMPLEX, EvoGAMES, EvoIASP, EvoINTELLIGENCE, EvoNUM, and EvoSTOC, Torino, Italy, April 27-29, 2011, Proceedings, Part I*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, ch. Multiple Tree for Partially Observable Monte-Carlo Tree Search, pp. 53–62. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-20525-5_6

[10] D. Auger and O. Teytaud, "The frontier of decidability in partially observable recursive games," *International Journal of Foundations of Computer Science*, vol. 23, no. 07, pp. 1439–1450, 2012.

[11] A. Saffidine, O. Teytaud, and S.-J. Yen, "Go Complexities," in *Advances in Computer Games*, ser. Advances in Computer Games, Leiden, Netherlands, 2015. [Online]. Available: https://hal.inria.fr/hal-01256660

[12] Y. Wang, J.-Y. Audibert, and R. Munos, "Infinitely many-armed bandits," in *Proceedings of Advances in Neural Information Processing Systems*, vol. 22. MIT Press, 2008.

[13] D. Robilliard and C. Fonlupt, "Towards human-competitive game playing for complex board games with genetic programming," in *Artificial Evolution 2015*, ser. Proceedings of Artificial Evolution 2015 (EA2015), Lyon, France, 2015. [Online]. Available: http://www-lisic.univ-littoral.fr/publis/1445328959.pdf

[14] J. B. Pollack and A. D. Blair, "Co-evolution in the successful learning of backgammon strategy," *Machine Learning*, vol. 32, no. 3, pp. 225–240. [Online]. Available: http://dx.doi.org/10.1023/A:1007417214905

[15] W. Jaśkowski and M. Szubert, "Coevolutionary CMA-ES for Knowledge-Free Learning of Game Position Evaluation," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 8, no. 4, pp. 389–401, Dec 2016.

[16] C. E. Bonferroni, "Teoria statistica delle classi e calcolo delle probabilità," *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commerciali di Firenze*, vol. 8, pp. 3–62, 1936.

[17] V. Mnih, C. Szepesvári, and J.-Y. Audibert, "Empirical bernstein stopping," in *Proceedings of the 25th Annual International Conference on Machine Learning (ICML 2008)*, A. McCallum and S. Roweis, Eds., 2008, pp. 672–679.

[18] I. Rechenberg, *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, ser. Problemata. Stuttgart-Bad Cannstatt: Frommann-Holzboog, 1973, no. 15.

[19] S. A. Kauffman, "Escaping the red queen effect," *The McKinsey Quarterly*, no. 1, pp. 118–130, 1995.

[20] L. Van Valen, "A new evolutionary law," *Evolutionary Theory*, vol. 1, pp. 1–30, 1973.

[21] P. Collet, E. Lutton, F. Raynal, and M. Schoenauer, "Polar IFS+Parisian Genetic Programming=Efficient IFS Inverse Problem Solving," *Genetic Programming and Evolvable Machines*, vol. 1, no. 4, pp. 339–361, Oct 2000. [Online]. Available: https://doi.org/10.1023/A:1010065123132

[22] D. L. St-Pierre, J. Hoock, J. Liu, F. Teytaud, and O. Teytaud, "Automatically reinforcing a game AI," *CoRR*, vol. abs/1607.08100, 2016. [Online]. Available: http://arxiv.org/abs/1607.08100

[23] T. Cazenave, J. Liu, and O. Teytaud, "The rectangular seeds of domineering," in *2015 IEEE Conference on Computational Intelligence and Games, CIG 2015, Tainan, Taiwan, August 31 - September 2, 2015*. IEEE, 2015, pp. 530–531. [Online]. Available: https://doi.org/10.1109/CIG.2015.7317904

[24] T. Cazenave, J. Liu, F. Teytaud, and O. Teytaud, "Learning opening books in partially observable games: Using random seeds in phantom go," in *IEEE Conference on Computational Intelligence and Games, CIG 2016, Santorini, Greece, September 20-23, 2016*. IEEE, 2016, pp. 1–7. [Online]. Available: https://doi.org/10.1109/CIG.2016.7860389

[25] J. Liu, O. Teytaud, and T. Cazenave, "Fast seed-learning algorithms for games," in *Computers and Games - 9th International Conference, CG 2016, Leiden, The Netherlands, June 29 - July 1, 2016, Revised Selected Papers*, ser. Lecture Notes in Computer Science, A. Plaat, W. A. Kosters, and H. J. van den Herik, Eds., vol. 10068. Springer, 2016, pp. 58–70. [Online]. Available: https://doi.org/10.1007/978-3-319-50935-8_6

[26] V. Ventos, Y. Costel, O. Teytaud, and S. Thépaut Ventos, "Boostring a bridge artificial intelligence," in *International Conference on Tools with Artificial Intelligence*. IEEE, 2017, pp. 1280–1287.

[27] M. Nica, "Optimal strategy in "guess who?": Beyond binary search," *Probability in the Engineering and Informational Sciences (accepted)*, 2015. [Online]. Available: http://arxiv.org/abs/1509.03327

[28] E. Lakshtanov and V. Roshchina, "On finiteness in the card game of war," *The American Mathematical Monthly*, vol. 119, no. 4, pp. 318–323, 2012. [Online]. Available: http://www.tandfonline.com/doi/abs/10.4169/amer.math.monthly.119.04.318

[29] A. E. Elo, *The rating of chessplayers, past and present*. New York: Arco Pub., 1978. [Online]. Available: http://www.amazon.com/Rating-Chess-Players-Past-Present/dp/0668047216