Parallel Noisy Optimization in Front of Simulators: Optimism, Pessimism, Repetitions Population Control

M. Oquab *Facebook AI Research*
J. Rapin *Facebook AI Research*
O. Teytaud *Facebook AI Research*
T. Cazenave *Lamsade, Univ. Paris Dauphine*

***Abstract*—We investigate parallel optimization methods for direct policy search, in the case of real-world problems originating in games, using noisy game simulator, with noise originating in the game design or in the multiplicity and randomization of possible players. We consider optimistic, average and pessimistic surrogates of the real fitness value, distinguishing exploration and exploitation. We conclude that (i) population-control methods, recently published in noisy optimization, perform greatly for continuous parameters *but* deceptive noise models exist (ii) uniform mixing of mutation rates combined with optimism in front of uncertainty performs greatly in the case of categorical unordered parameters *or* in front of those deceptive noise models in continuous domains (iii) the method of seeds provides easy significant improvements though, usually, it does not scale up with the training computational power.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Given a game, let us have an algorithm (the *game AI*) following a behavioral policy π, mapping states s to actions a. We assume that the behavioral policy is represented by a vector of parameters $\theta \in \mathbb{R}^N$, and our goal is to find θ yielding the highest winning rate for a game. The result (win/lose) is observed long after the actions are taken, and in general it is not possible to compute derivatives. Moreover, the games we consider are not deterministic in general. Therefore, this game AI tuning can be seen as an instance of noisy derivative-free optimization: this is *Direct Policy Search*, with losing rate as a loss function. The present paper is motivated by the reproducibility crisis; and by the need for comparison with the state of the art in recent innovative machine learning papers using derivative-free optimization. Notably, several recent papers [26], [20] use variants of evolution strategies (ES), in particular in a noisy setting; we contribute to revisit this body of work by including the ES with best known convergence rates in the noisy setting [19]; by incorporating optimism in front of uncertainty (OFU) [31]; and by taking into account the recent progresses in terms of mutation rates in discrete settings [16], [12]. We focus on numerically challenging cases, with a focus on reproducibility: we open source our platform, compare with many baselines, consider high dimension cases, work without pixel-level inputs, with either expensive experiments (as for Battleship with expensive maximum likelihood estimation) or cases in which the signal over noise ratio is low due to highly stochastic games in which winning rates over a single game are close to 50% (in GuessWho the dichotomy policy is a strong baseline, in War picking up strong cards first is hard to beat, in Flip the 1-ply search is almost optimal).

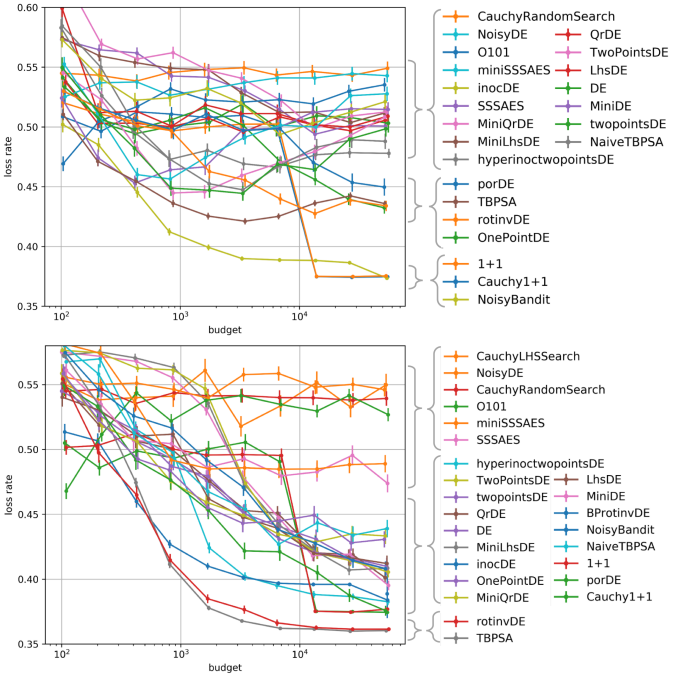

Fig. 1. Results on the game007 with simple linear controller, and neural controller respectively. The dimension is moderate, resp. 12 and 21 - in dimension 12 TBPSA fails (see Fig. 2 for more) and NoisyBandit is excellent. We note the strong performance of Rotation-Invariant DE [23]. TBPSA performed best for the neural net; this will be the case also for bigger nets (Fig. 7).

a) Contributions.: (i) For both continuous and discrete cases, we combine optimism in front of uncertainty (OFU [22]) and evolution strategies (ES), and show that this combination ES+OFU obtains superior results on a variety of games (see Sec. III-B3 and Figs. 5 and 6). This ES+OFU approach was never explored to the best of our knowledge. We incidentally extend the FastGA algorithm introduced by [16] to the case with domain $\{0, 1, 2, \ldots, a-1\}^N$ (compared to $\{0, 1\}^N$).

(ii) For continuous cases, we implement and test the population control component introduced by [19]; we provide

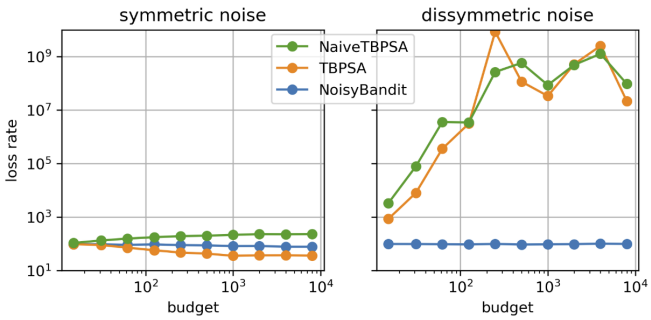

Fig. 2. How a dissymetric noise model can make TBPSA diverge. Left: symmetric noise model, TBPSA wins easily; also against many optimizers in [5], removed for clarity. Right: dissymetric noise model; TBPSA is outperformed by the simple bandit. The noise model has variance $\Theta(1)$ symmetric around the optimum in the former case; and with variance set to zero when the first coordinate is < 0 ($\Theta(1)$) otherwise) in the latter case. Dimension 1, sphere function, noise level 10. Details in the open source[5] (reproducible by "python -m nevergrad.benchmark metanoise –plot –num_workers 12" with 12 threads); experiments in higher dimension are similar.

an independent and real-world confirmation of its efficacy, in particular where the parameterization of a game AI corresponds to a neural network (see Sec. V-A and Figs. 1 and 7). We also propose a minimal deceptive noise case reproducing a failure mode of this approach (Fig 2); this failure was not apparent in any of our neurocontrol experiments, but sometimes in test cases involving less overparametrized control methods (Fig. 3, 5).

(iii) For categorical unordered cases, we implement and test the hash-seed method introduced in [9], employed to win the world bridge competition [30]; this method optimizes the seeds of randomized subcomponents of a game AI. We experimentally show that this method is effective in the case of partially-observable games, with surprisingly good performance with low budget but not that much improvement for larger training budgets; and less convincing results in fully-observable games for which existing methods are probably more suitable (See Sec. II-B and Fig. 4).

II. Noisy derivative free optimization

A. Different features of optimization problems

The tuning problem is defined by an AI with parameters, and a loss function measuring the performance. We here assume a fixed opponent or a fixed distribution of opponents, and the loss function is the proportion of lost games. The following features of Direct Policy Search problems are discussed in the present paper and correspond to different cases. *Categorical unordered vs continuous*: whether the expected loss function, as a function of each parameter separately, is reasonably smooth. When the problem is categorical unordered, similar parameters can lead to very different loss values, and in this case we are basically solving a discrete optimization problem. This is the case, for example, when we use the hash-seed method (Section II-B); we get unordered variables with a huge alphabet. *Noise*: whether the objective function is noisy, and presents non-negligible variance even when approaching the optimum. This is the case when the game has a non-deterministic nature, or optimal policies are stochastic. *Inoculation*: when there is a baseline that we can reproduce with a specific value of the parameter vector. Without loss of generality, we then adapt (by translation) the behavioral policy such that $\theta = 0_{\mathbb{R}^N}$ corresponds to this baseline, and we use it as a starting point for direct policy search.

B. Hash-openings with random seeds

In this section we describe the hash-seeds method of [9] that allows building easily an adequate parameterized policy function π: let us consider an algorithm based on a randomized subcomponent following a random seed, such as Monte Carlo Tree Search (MCTS) or Monte Carlo (MC, e.g. for defogization). We can define a SeedAI(π) as in Algorithm 1. This approach is successful and allows designing better game AIs [8], notably winning the bridge world competition [30]. We note that in this case, optimizing SeedAI is a problem with categorical unordered variables. The policies for several games are implemented using this approach, for our experiments in Section V. We will observe in the experiments (Sec V-D, Fig. 4) that optimizing the random seed in MC(TS)-based approaches leads to significant improvements in many cases, within a few hundreds games of trainings, though unfortunately the method often plateaus and does not scale up to dozens of thousands of games.

III. Algorithms

Our algorithms are presented in Table I.They are defined by an *exploration* method (providing the next points at which the objective function has to be evaluated), and by a *recommendation* method, specifying what is the approximate optimum as provided by the optimization algorithm when the budget of exploration points is elapsed.

A. Continuous and Discrete $(1+1)$ evolution strategy

A $(1+1)$ evolutionary search is defined as follows in the continuous case [27]. $\theta_0 \in \mathbb{R}^N$ is the initial parameter vector, σ_0 is the initial step-size in $\mathbb{R}$, and at each iteration we define $\theta'_n = \theta_n + \sigma_n \mathcal{N}_d$ with $\mathcal{N}_d$ a standard Gaussian in dimension d. Then θ'_n is evaluated; θ_{n+1} is the best among θ_n and θ'_n; if θ_n loses we decrease the step-size by setting $\sigma_{n+1} = 2^{-\frac{1}{4}}\sigma_n$; otherwise we increase it by setting $\sigma_{n+1} = 2\sigma_n$.

A $(1+1)$ evolutionary search is defined as follows in the discrete case [15]. Let $\theta^0 \in \mathbb{R}^N$ be the initial parameter vector of our policy π. Let $f : \mathbb{R}^N \to \mathbb{R}$ be a loss function. For each time step t, we define θ_{t+1} as follows: (i) choose a mutation rate p_t (see below). (ii) randomly mutate each coordinate $\theta_{t,i}$ with probability p_t, obtain candidate parameter vector c (in the present paper, given that we have many possible values per categorical unordered variable, mutated coordinates are just uniformly independently randomly drawn). (iii) if $f(c) < f(\theta_t)$, set $\theta_{t+1} = c$, otherwise $\theta_{t+1} = \theta_t$. This procedure allows optimizing an individual vector θ according to the loss criterion f. This procedure is often extended to a population of λ individuals, to evolve many vectors θ simultaneously in parallel.

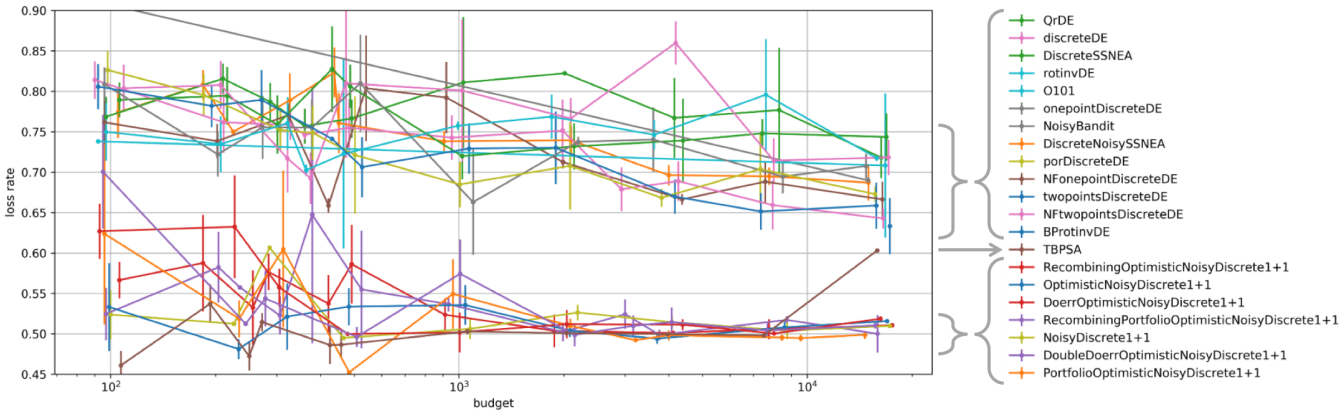

Fig. 3. Flip game. No algorithms managed to outperform the baseline, but all algorithms which do not diverge far from it are based on evolutionary programming + OFU. Surprising results for TBPSA here (converging very decently and then diverging) inspired the deceptive function used in Fig. 2.

Game	Winning rate	Cherry-picked and verified win rate
Partially observable games		
Battleship	56% ± 2%	57 %
Battleship2	54% ± 2%	57 %
Phantomgo	52 % ± 2%	59 %
Phantomgo9	53 % ± 2%	55 %
Golois	70 % ± 1%	
Metawar	57 % ± 1%	
Fully observable games		
Knightthrough	52 % ± 2%	53 %
Atarigo	53 % ± 1%	
Nogo	52 % ± 1%	
Breakthrough	51 % ± 1%	
Domineering	50 % ± 1%	
MisereBreakthrough	49% ± 1%	
MisereDomineering	50% ± 1%	
MisereKnightthrough	53% ± 1%	

Fig. 4. Performance obtained by the hash-seed method with budget 300 (except Metawar:1000). In all these games the optimization algorithm had little impact on the result; we present the average result for the maximum budget we ran, except for games for which there was a big gap, in which case we cherry-picked and verified the performance of the best. Improvements are negligible for fully observable games, but interesting in partially observable games for which learning is by nature hard.

1) Mutation rates selection methods: Portfolios of mutation rates and FastGA.: The mutation rate selection procedure can have a substantial impact on the success of evolution-based algorithms. danglehre investigated p_t uniformly drawn in a portfolio $\{1/N, \dots, (N-1)/N\}$ of mutation rates. This is termed PortfolioDiscrete$(1+1)$. fastga mathematically advocated FastGA, a mutation rate randomly drawn in $\{1/N, \dots, \frac{1}{2}\}$ according to a power law; we extend it to randomly drawn in $\{\frac{1}{N}, \frac{2}{N}, \dots, \frac{N-1}{N}\}$ with a power law; this extension is natural in the context of switching from $\{0,1\}^d$ to $\{0,1,2,\dots,a-1\}^d$. This algorithm is termed DoubleFastGADiscrete$(1+1)$.

B. Algorithms for noise management

1) Non-adaptive resampling: If the loss function f is noisy, then comparing two points may not be reliable. Various papers advocate resampling and averaging [28], [10] for reducing the noise around the optimum; we implement the O101 heuristic of [10] (resampling 1.01^n times for the n^{th} explored point) as well as many other algorithms all available open source in nevergrad. We do not provide further details as these methods

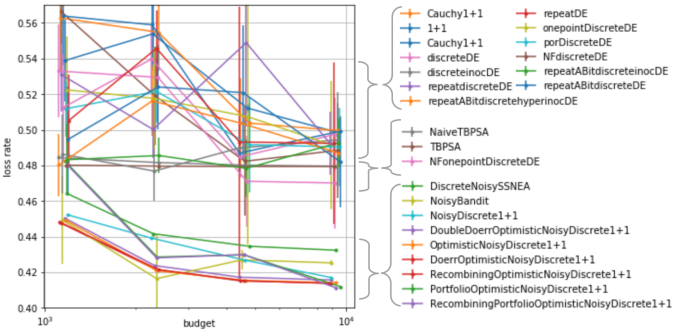

Fig. 5. Results on the Guesswho game (standard case, with 24 characters). We display a subset of the methods from [5] that we applied. Non presented results on the 96 characters are somehow similar: all strong methods use a noisy adaptation of ES; in this low dimensional noisy context, the best methods treat variables as if they were unordered and even bandits (which do not use any structure of the problem) perform reasonably well. TBPSA performs surprisingly poorly, as in Fig. 3; these results inspired the deceptive model in Fig. 2.

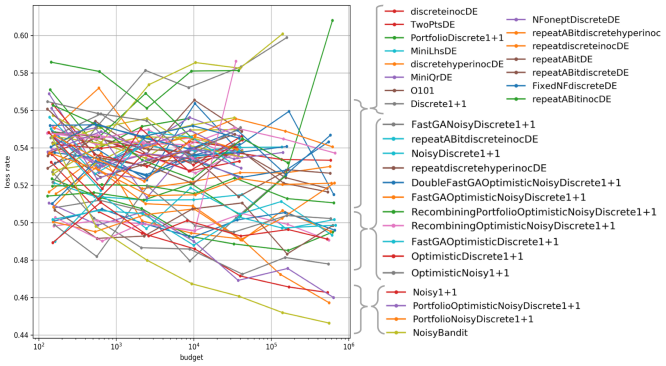

Fig. 6. Game of War: methods based on discrete ES + OFU dominate, in particular with portfolios of mutation rates, except the great score of the simple bandit method (which is basically random search + OFU). We also experimented the variant Batawaf with similar results though NoisyBandit was less impressive.

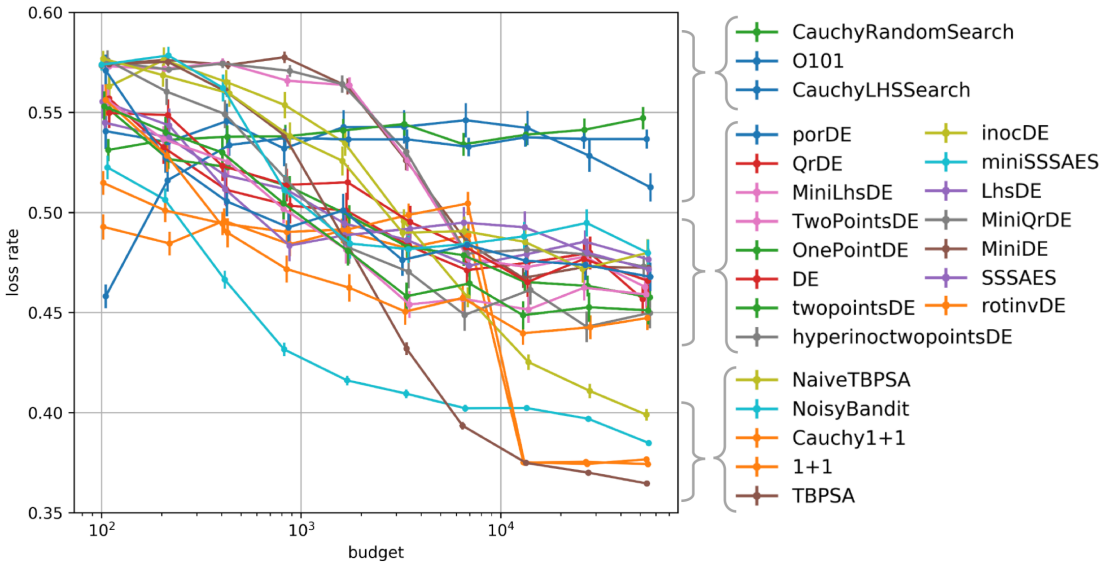

Fig. 7. Results on the game007 with neural controller with 2 hidden layers; unpresented experiments with 3 hidden layers (39 parameters) are similar and not presented. The number of weights is moderate, respectively 30 and 39. TBPSA performed best.

Algorithm 1 SeedAI: Game AI using hash-opening method.

Input: a state s
Parameters: hash function H, list of N seeds θ, a randomized policy π so that action = π(state, random seed)
$\hat{r} \leftarrow H(s) \,\%\, N$
$\hat{a} \leftarrow \pi(s, \hat{r})$
return $\hat{a}$

Note: π might be a policy based on MC or MCTS or any stochastic decision method. θ is the N-dimensional parameter that will have to be optimized by Direct Policy Search.

were not the best performers in our experiments.

2) TBPSA (continuous case): Another approach is to evaluate points further away (stronger mutation rate) and perform small steps in the best direction, relying on the longer-range trends of the objective landscape. Several papers, in the continuous setting, have followed this "mutate large, inherit small" principle (MLIS [3]), consistently with theoretical results in [1], [13], leading to fast convergence rates [19] - not that far from concepts in [17]. In the following, we describe two methods that we study experimentally in Section V. Such an approach is implemented in pc-CMSA-ES [19]. Besides other traditional features of evolutionary algorithms with covariance matrix, the method features a population control (PC) component, regularly adjusting the size of the population with a statistical test. In addition, it provides, as a recommendation, the center of the current distribution estimate, rather than the individual that got the best objective value. We implement in our platform a similar version of this component (Alg. 3); this method outperforms SPSA [24] on noisy sphere functions testbeds [5].

TBPSA, as well as pc-CMSA-ES and consistently with MLIS, uses the center of the current Gaussian as a recommendation. Another version, NaiveTBPSA, designed for rugged noise-free cases, uses the best point so far rather than the Gaussian center; NaiveTBPSA keeps the PC mechanism. Experiments show consistently high performance for TBPSA in continuous noisy cases (Sec. V-A, Fig. 7), in particular in high dimension when everything else fails. However, after initial convergence, TBPSA sometimes diverges (Fig. 3); an outcome of the present paper is the understanding of this phenomenon, as shown in Fig. 2.

TABLE I
OUR OPTIMIZATION ALGORITHMS. IMPLEMENTATIONS AVAILABLE IN [5].

Algorithms, variants and references
Continuous $(1+1)$-Evolution Strategy: • Original: [27], [25], [14] • Naturally adapted to $(1, \lambda)$ by the ask and tell interface [4] • Noisy variants (Section III-B3) • Step-size self-adaptive ES (SSSAES [4])
Discrete $(1+1)$-Evolution Strategy: • Original: [25] • Portfolio / uniform mixing of mutation rates [12] • FastGA [16] • Noisy / optimistic variants (Section III-B3, Table 3)
TBPSA (Test-Based Population Size Adaptation): • Original: [19]: population control, center of Gaussian as a recommendation policy. • Naive variant: best-so-far recommendation policy.
NoisyBandit: • Original: UCB [2]. • Progressive Widening [11], [31] (note: PW is used in all our experiments).
Differential Evolution (DE): • Original: [29] • Rotationally Invariant (RotInvDE): [23]
Particle Swarm Optimization (PSO): • Original: [21]

Algorithm 2 Test-based population size adaptation. Adapted from [19].

Input: population size λ, loss function f
$P1$:= losses of the 20% oldest of the last 5λ points
$P5$:= losses of the 20% most recent of the last 5λ points
$mean(P1)$-$mean(P5) > 2\sqrt{\frac{var(P1)}{\lambda-1} + \frac{var(P2)}{\lambda-1}}$
$\lambda \leftarrow 2\lambda$

$\lambda \leftarrow \max(4d, 2^{-\frac{1}{4}}\lambda)$ where d is the dimensionality

Note: the minimal population size is equal to $4d$ where d is the dimensionality of the parameter θ.

Algorithm 3 OptimisticNoisy (resp. Noisy) algorithm based on an existing algorithm A.

Input: loss function f, budget (max. num. evaluations of f).
Initialize at the empty set an archive of n_{points} vectors $(\theta)_i$; n_{points} := 0.
n_{ask} := 0
$n_{ask} <$ budget
$n_{ask} < n_{points}^3$
launch one more evaluation of the best optimistic point
(resp. of a random $(\theta)_i$ in the archive)
increment n_{ask}
$n_{ask} \geq n_{points}^3$
Take care that A sees pessimistic bounds as objective values in its archive.
Use A for choosing a new point θ for evaluation.
add θ to the archive, increment n_{points} by 1.
return as a recommendation the best pessimistic point θ^*

3) Adaptive resampling: progressive widening and optimism in front of uncertainty: Methods presented here can be applied for adapting evolution strategies, both in discrete and in continuous domains, to cases with stochastic objective values. Given the archive $(\theta_i)_{0 \leq i \leq n_{points}}$ of all points at which the objective function has been evaluated at least once, we consider statistics over the loss evaluations $f(\theta_i)$, where θ_i has been evaluated $n_{evals,i}$ times and an estimate of the standard deviation is σ_i: (i) the optimistic bound $\mu_i - \frac{\sigma_i}{\sqrt{n_{evals,i}-1}}$, (ii) the pessimistic bound $\mu_i + \frac{\sigma_i}{\sqrt{n_{evals,i}-1}}$. This lets us augment an existing algorithm A to its "OptimisticNoisy" or "Noisy" counterpart (Alg. 3). Both use progressive widening as in [11], [31]; the "Noisy" counterpart uses, for exploration, a uniform allocation among visited points whereas the "OptimisticNoisy" counterpart uses upper confidence bounds for distributing allocation. The way new arms are pulled depends on the original algorithm, but using the pessimistic bounds as objective values when a concept of best so far is needed. As pointed out in [6], the superiority of the optimistic counterpart is not obvious given the asymptotic optimality of uniform allocation for pure exploration bandits; however, consistently with the experiments in [6], Optimism is validated in our experiments.

Using such optimistic and pessimistic bounds is inspired by the *Upper Confidence Bound* [2] method; increasing the size of the pool when $n_{ask} > n_{points}^3$ follows the *Progressive Widening* method [11], [31]. With these modifications, the method PortfolioDiscrete$(1+1)$ becomes PortfolioOptimisticNoisyDiscrete$(1+1)$ and DoubleFastGADiscrete$(1+1)$ becomes DoubleFastGAOptimisticNoisyDiscrete$(1+1)$. We observe in the experiments (Sec V-A, Fig.5) that these algorithms perform well in continuous noisy cases, in particular in the *inoculated* cases.

C. Adaptation of Differential Evolution: discrete case, noisy case, genetic crossover, and inoculation

We use Differential evolution (DE, by de) in its curr-to-best form: for a current individual i, we define a donor $i + F1(a-b) + F2(best-i)$ with a and b randomly drawn, with $best$ the current best individual, and with $F1 = F2 = 0.8$; then, each parameter is mutated to the value of the donor with independent probability $CR = \frac{1}{2}$, producing a candidate. $(a-b)$ preserves diversity in the population, while $(best-i)$ improves the objective values of the population. DE is frequently a component of winning methods in derivative-free optimization competitions. Rotationally invariant versions are based on crde; the exactly rotationally invariant form uses $CR = 1$ but then all candidates lie in the vector space generated by the initial population; this can be mitigated by using a big population (BP suffix), i.e. for us 7 times the dimension; or by using $CR = 0.9$ (almost rotationally invariant DE), which performed quite well in many cases. We designed adaptations of DE for the discrete case and we included genetic crossovers in DE, as well as specific initialization schemes and inoculation methods; results were reasonably good but no game changer for problems as in the present paper so that details are reported in [5].

IV. Games and baselines

AtariGo is a variant of Go in which the first capture wins. PhantomGo is another variant, with partial observation. Golois is a strong implementation of PhantomGo. Knightthrough, Breakthrough and Domineering are classical board games from computer game competitions; their Misere versions correspond to "who lost for the classical version wins for the Misere version". War is a card game in which the only action consists in choosing the order in which cards are picked up when a ply is won. Batawaf is a variant of the War game, usually played by small kids for learning elementary mathematics. Flip is another card game, in which players must get rid of their cards first; the traditional version is not turn-based but our version is. Game007 is sometimes known as "standoff James Bond"; we consider the variant in which blocking 5 times in a row is forbidden.

In Battleship, the 20 first parameters are seeds for randomly drawing a position; a policy is made stochastic by using a randomly drawn seed among these 20 seeds. The next 20 parameters are also seeds, for the randomized shooting part: the shootings are chosen by sampling, randomly, several possible hidden states (defogized states) and picking up one of the positions with maximum frequency of being a hit over the defogized states. Battleship2 uses a bigger board (10x10 instead

TABLE II
PARAMETRIZATION OF OUR AIS.

Golois	seeds for Monte Carlo for time step i for $i \in \{0, 1, 2, \ldots, 9\}$
Atarigo (Misere-)Breakthrough (Misere-)Domineering (Misere-)Knightthrough Nogo	Seeds for Monte Carlo Tree Search: one seed for each of the 50000 possible values of $hash(state)$ modulo 50000
Battleship Battleship2	20 seeds corresponding to 20 different pure policies for choosing positions and 20 seeds corresponding to the stochastic defogization for shooting
War Batawaf	Seed for each of 3380 randomly drawn permutations (resp. 1080 for Batawaf) used for choosing the order (see text) depending on 3380 distinct contexts.
Game007 Game007NN Game007DoubleNN Game007TripleNN	12 parameters of a linear mapping building logit for 3 actions using 4 descriptors (my munitions, your munitions, my number of successive protections, your number of successive protections) 21 parameters of a 1-layer neural net with same inputs/outputs 30 parameters of a 2-layers (...) 39 parameters of a 3-layers (...)
Flip	3249 parameters of a quadratic value function used in 1-ply search

of 7x7), more boats (sizes 2, 3, 3, 4, 5 instead of 4, 5); it is the most classical version of Battleship. "Meta" versions of games correspond to cases in which the reward is averaged over 5000 games, i.e. the noise is almost cancelled. In War or Batawaf, each parameter is a seed, used for generating a permutation, corresponding to one category of states; there are 3380 categories of states [1]. The permutation is the order in which cards are picked up when we win a ply; zero corresponds to pick up best cards first when winning a ply, which is a strong baseline[7]. The parametrizations of our algorithms are presented in Table II. For several games (AtariGo, Breakthrough, Domineering, Golois, and Misere variants thereof) the default policy is to use a randomized seed for launching the MCTS: the parameters are the seeds for each bucket of hash codes as detailed in Section II-B. The baseline for Game007 is to randomly uniformly draw a legal non-stupid action;for Battleship and PhantomGo, the baseline uses a true random seed in the Monte Carlo defogization.

V. EXPERIMENTAL RESULTS

Unless stated otherwise, all experiments are performed with population-size 30. For this we use the ask-and-tell interface of our open source platform[5], which naturally extends algorithms to the parallel case; for example, even the $(1+1)$ evolution strategy becomes population-based and is actually a $(1, \lambda)$ asynchronous evolution strategy (see terminology of evolution strategies in Beyer:bookES). We do not discuss MetaGuesswho and MetabigGuesswho, which are continuous, low dimensional and noise-free and for which many methods reach the same performance, namely 38% and 40% losing rate respectively, incidentally confirming that the dichotomy policy is suboptimal.

A. Continuous and noisy, with/without inoculation

Our first experiment is on the Shotgun007 game (see Figs. 1 and 7), where the parameters correspond to the weights of a neural network, initialized randomly (no inoculation); in this case, we observe that TBPSA is among the best-performing algorithms. We note that RotInvDE can also be excellent in spite of not being designed for noise; this is a surprise for future investigation. Our experiment on the Flip game (see Fig. 3), where the (1-ply-search baseline) inoculation is strong - possibly optimal - shows that our ES+OFU algorithms are stable and do not diverge when initialized with a good starting point, as they do not discard this inoculation during optimization. Similarly, our experiment on GuessWho (see Fig. 5), where the (dichotomy baseline) inoculation is good but not optimal also shows the performance and stability of ES+OFU.

[1]States correspond to the ply and the number of remaining cards; $3380 = 10 \times 26 \times 13$, 10 possible values for the number of remaining cards in $\{0, 1, \ldots, 8, \geq 9\}$, 26 possible values for the (even) number of cards in the ply, 13 possible values for the best card in the ply.

TABLE III
GAMES USED IN THE PRESENT PAPER. COMPUTATION TIMES VARY FROM SECONDS (MOST GAMES) TO MINUTES (PHANTOMGO) AND HOURS (BATTLESHIP). ALL GAMES WHICH ARE TUNED WITH MORE THAN 10000 PARAMETERS, AS WELL AS BATTLESHIP GAMES, CORRESPOND TO THE HASH SEED METHOD. FOR BATTLESHIP VARIANTS, THE SEEDS CORRESPOND TO SEVERAL PURE POLICIES USED IN A COMBINED MANNER, SO THAT WE OPTIMIZE A MIXED POLICY. FOR GOLOIS, WE USE ONE SEED FOR EACH TIME STEP OVER THE 10 FIRST TIME STEPS, INDEPENDENTLY OF THE STATE (IN MANY CASES THE STATE CONTAINS NO INFORMATION DURING THE FIRST TIME STEPS IN PHANTOMGO); WE USE A FAST SETTING WITH TIGHT TIME CONSTRAINTS.

Game	Dimension	Is scrambled	0 is great	Noisy	Section	Figure
flip	3249	False	True	True	V-A	3
GuessWho	4	False	True	True	V-A	5
BigGuessWho	4	False	True	True	V-A	5
MetaGuessWho	4	False	True	False	V	
MetaBigGuessWho	4	False	True	False	V	
Game007	12	False	False	True	V-A	1
Game007nn	21	False	False	True	V-A	1
Game007doublenn	30	False	False	True	V-A	7
Game007triplenn	39	False	False	True	V-A	7
Battleship	40	True	False	A bit	V-D	4
Battleship2	40	True	False	A bit	V-D	4
phantomgo	20000	True	False	A bit	V-D	4
phantomgo9	20000	True	False	A bit	V-D	4
batawaf	1080	True	True	True	V-B	6
war	3380	True	True	True	V-B	6
metawar (5000)	3380	True	True	False	V-C	4
atarigo	50000	True	False	True	V-D	4
breakthrough	50000	True	False	True	V-D	4
domineering	50000	True	False	True	V-D	4
knightthrough	50000	True	False	True	V-D	4
misereBreakthrough	50000	True	False	True	V-D	4
misereDomineering	50000	True	False	True	V-D	4
misereKnightthrough	50000	True	False	True	V-D	4
nogo	50000	True	False	True	V-D	4
golois	10	True	False	True	V-D	4

B. Categorical unordered with high noise and inoculation

A problem with several categorical unordered variables is not the same as a bandit (fully unstructured) problem; we still have some structure in the combination between variables, i.e. the problem might be fully or partially separable [18]. We now consider the game of War and its variant Batawaf (Fig. 6); there is inoculation because 0 is the known policy [7] consisting in making best cards more frequent by picking up the best cards first when winning a ply. Our variants of DE aimed at dealing with noise (details in [5]) did not perform well, methods from [5] using resampling also failed; TBPSA, which is by nature designed for continuous ordered variables does not make any sense. Again, the best methods for this game are ES (FastGA and Portfolio) augmented with OFU. We observed that optimism generally improved noisy methods.

TABLE IV
CONCLUSIONS.

Settings, and recommended method
Categorical unordered noisy optimization, **or** Continuous noisy with dissymetric noise (Fig. 2), **or** Continuous noisy optimization with strong inoculation: • Use uniform mixing of mutation rates [12] and optimism in front of uncertainty ([31], Sect. III-B3).
Policy based on a randomized subcomponent such as MC or MCTS: • Use random seed optimization with the hash-opening method of [9], and optimize as above.
Continuous noisy optimization without too much noise disymetries (incl. overparametrized neurocontrol): • Use population-control as in [19].

C. Categorical unordered noise-free with inoculation

Compared to the original War game, we here consider Metawar, i.e. the game of War averaged over 5000 games (Fig. 4), so that noise is considerably reduced. In this reduced noise setting, methods which fight noise perform poorly as they use the computational budget for dealing with non-existent noise. All other methods perform nearly equivalently.

D. Categorical unordered without inoculation

We here refer to Battleship, Phantomgo, Golois, Atarigo, Nogo, Breakthrough, Knightthrough, and misere variants. We observed that the hash-seed method provides significant improvements especially in partially-observable games, but less so in fully-observable games (Fig. 4).

VI. CONCLUSION

A short version of the conclusion is in Table IV.

a) Merging ES and OFU.: Our simple methods for adding noise management into ES are surprisingly powerful and compatible with both discrete and continuous settings. Our "optimisticnoisydiscrete1+1" performs well on a wide range of problems. This is just a resampling rule (namely, "resample the current point with best optimistic bound if the number of distinct points is more than the cubic root of the number of *ask*") plus "use the best point from a pessimistic bound point of view for operating the mutations" and "use the best point from a pessimistic bound point of view for the recommendation"- a fairly simple modification of the classical evolution strategy. The portfolio version (corresponding to the mixing in [12]) performs great overall. This is particularly visible in Fig. 5 and 6 (all algorithms with this add-on perform well and almost only them). All algorithms based on non-adaptive repetitions in [5] were outperformed in nearly all cases. ES with OFU were even the best methods in some continuous cases, namely when the noise model makes TBPSA unreliable - which happened in real world cases first and was then explained by a deceptive function as in Fig. 2.

b) Population control for noisy continuous optimization.: Test-based population-size adaptation (TBPSA in the present paper), directly inspired from [19], performs well for continuous noisy optimization when the variance at the optimum is significant and there is no strong inoculation; in artificial experiments[5], but also real-world experiments in Fig. 3, 1, 7. There are however cases in which TBPSA diverges after coming close to the optimum, presumably due to "domain dissymetry" of the noise (Fig. 3, 5). Importantly, most artificial functions used in optimization papers have (almost) symmetric noise models (i.e. same noise for candidates on the left of the optimum and on the right of the optimum, with left/right corresponding to the first axis); from a detailed investigation of our experimental results, we managed to build a simple deceptive function for population-control ES (Fig. 2). **Other methods for noise in continuous domains:** We point out a few great results of rotationally invariant DE (RotInvDE), with $CR = 1$; this surprising result (no reason for this algorithm to converge in the noisy setting) is left for further investigation. Bandits methods[31], in spite of their incredible simplicity and their ignorance of the domain structure, were sometimes not that bad.

c) The simplicity and effectiveness of hash-based methods.: The seed method (Table 1) is surprisingly effective for learning a policy when the problem is untractable otherwise. Significant and sometimes strong benefits are obtained with a few lines of code. For example, we get significant improvements in Fig. 6 and 4 (game of war and variants, in which the baseline is an almost optimal policy; Battleship game, in which the baseline already uses a defogization method for choosing where to shoot); Fig. 5 for which the baseline is the usual dichotomy policy; also a few other fully and partially observable games in Fig. 4. The method failed in rare cases. With the game of War, we have an application of the seeds method which does not involve MC or MCTS.

REFERENCES

[1] S. Astete-Morales, M.-L. Cauwet, and O. Teytaud. Analysis of different types of regret in continuous noisy optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 205–212. ACM, 2016.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[3] H.-G. Beyer. Mutate large, but inherit small! on the analysis of rescaled mutations in ($(1, \lambda)$-es with noisy fitness data. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature — PPSN V*, pages 109–118. Springer, 1998.

[4] H.-G. Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer, Heideberg, 2001.

[5] D. Blind. "double blind (open source)". In *https://anonymous.4open.science/repository/3c02d4aa-1927-4eb6-900a-472bd555aa6f/*, 2018. Accessed: 2018-12-21.

[6] S. Bubeck, R. Munos, and G. Stoltz. Pure exploration in finitely-armed and continuous-armed bandits. *Theor. Comput. Sci.*, 412(19):1832–1852, 2011.

[7] M.-L. Cauwet and O. Teytaud. Surprising strategies obtained by stochastic optimization in partially observable games. In *CEC 2018 - IEEE Congress on Evolutionary Computation*, pages 1–8, July 2018.

[8] T. Cazenave, J. Liu, F. Teytaud, and O. Teytaud. Learning opening books in partially observable games: using random seeds in phantom go. *CoRR*, abs/1607.02431, 2016.

[9] T. Cazenave, J. Liu, and O. Teytaud. The Rectangular Seeds of Domineering, Atari-Go and Breakthrough. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 530 – 531, Aug. 2015.

[10] S.-Y. Chiu, C.-N. Lin, J. Liu, T.-C. Su, F. Teytaud, O. Teytaud, and S.-J. Yen. Differential Evolution for Strongly Noisy Optimization: Use 1.01^n Resamplings at Iteration n and Reach the -1/2 Slope. In *2015 IEEE Congress on Evolutionary Computation (IEEE CEC)*, May 2015.

[11] R. Coulom. Computing Elo Ratings of Move Patterns in the Game of Go. In H. J. van den Herik, M. Winands, J. Uiterwijk, and M. Schadd, editors, *Computer Games Workshop*, June 2007.

[12] D. Dang and P. K. Lehre. Self-adaptation of mutation rates in non-elitist populations. In *Parallel Problem Solving from Nature - PPSN XIV - 14th International Conference*, pages 803–813, 2016.

[13] J. Decock and O. Teytaud. Noisy optimization complexity under locality assumption. In *Proceedings of the Twelfth Workshop on Foundations of Genetic Algorithms XII*, FOGA XII '13, pages 183–190. ACM, 2013.

[14] L. Devroye. The compound random search. In *Proceedings of the International Symposium on Systems Engineering and Analysis*, pages 195–210, 1972.

[15] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative drift analysis. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 1449–1456, 2010.

[16] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen. Fast genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '17, pages 777–784. ACM, 2017.

[17] V. Fabian. Stochastic approximation of minima with improved asymptotic speed. *Ann. Math. Statist.*, 38(1):191–200, 02 1967.

[18] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger. Impacts of Invariance in Search: When CMA-ES and PSO Face Ill-Conditioned and Non-Separable Problems. *Applied Soft Computing*, 11:5755–5769, 2011.

[19] M. Hellwig and H.-G. Beyer. Evolution under strong noise: A self-adaptive evolution strategy can reach the lower performance bound - the pcCMSA-ES. In J. Handl, E. Hart, P. R. Lewis, M. López-Ibáñez, G. Ochoa, and B. Paechter, editors, *Parallel Problem Solving from Nature – PPSN XIV*, pages 26–36. Springer International Publishing, 2016.

[20] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu. Population based training of neural networks. *CoRR*, abs/1711.09846, 2017.

[21] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948, 1995.

[22] T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.*, 6(1):4–22, Mar. 1985.

[23] J. Montgomery and S. Chen. An analysis of the operation of differential evolution at high and low crossover rates. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[24] P. Rastogi, J. Zhu, and J. C. Spall. Efficient implementation of enhanced adaptive simultaneous perturbation algorithms. In *2016 Annual Conference on Information Science and Systems, CISS*, pages 298–303, 2016.

[25] I. Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, 1973.

[26] T. Salimans, J. Ho, X. Chen, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *CoRR*, abs/1703.03864, 2017.

[27] M. Schumer and K. Steiglitz. Adaptive step size random search. *IEEE Transactions on Automatic Control*, 13(2):270–276, 1968.

[28] F. Stonedahl and S. H. Stonedahl. Heuristics for sampling repetitions in noisy landscapes with fitness caching. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, GECCO '10, pages 273–280. ACM, 2010.

[29] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *J. of Global Optimization*, 11(4):341–359, Dec. 1997.

[30] V. Ventos, Y. Costel, O. Teytaud, and S. T. Ventos. Boosting a bridge artificial intelligence. In *29th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2017, Boston, MA, USA, November 6-8, 2017*, pages 1280–1287, 2017.

[31] Y. Wang, J.-Y. Audibert, and R. Munos. Algorithms for infinitely many-armed bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1729–1736. Curran Associates, Inc., 2009.