Herding Evolutionary Algorithm

Arnaud Berny
arnaud@courros.fr

ABSTRACT

In this paper, we address the problem of black box optimization over binary vectors. We introduce a novel evolutionary algorithm called Herding Evolutionary Algorithm which relies on herding to generate individuals with empirical moments close to those of selected individuals. We report experiments with diverse fitness functions and compare the results of HEA with those of simulated annealing, local search, and PBIL. Although in its early developments, HEA shows promising results.

CCS Concepts

•Mathematics of computing → Evolutionary algorithms; •Computing methodologies → Discrete space search;

Keywords

Herding; black box optimization; binary vectors

1. INTRODUCTION

Herding [7, 6] is an efficient deterministic algorithm which approximately solves the moment matching problem. To the best of our knowledge, it has not yet been applied to evolutionary algorithms or, more generally, to black box optimization over binary vectors.

We introduce a novel evolutionary algorithm called Herding Evolutionary Algorithm (HEA) which is reminiscent of PBIL [2]. HEA departs from traditional Estimation of Distribution Algorithms in that it does not build any explicit statistical model of the population; instead, it explores the search space by pursuing the empirical first and second moments of selected individuals.

The paper is organized as follows. In Section 2, we recall the principles of herding. In Section 3, we introduce HEA. In Section 4, we report the results of HEA and other search algorithms. Section 5 concludes the paper.

GECCO'15 Companion, July 11–15, 2015, Madrid, Spain.

ACM ISBN 978-1-4503-3488-4/15/07.

DOI: http://dx.doi.org/10.1145/2739482.2764660

2. HERDING

Let $\mathcal{X}$ denote the sample space. We will consider the cases $\mathcal{X} = \{0,1\}^n$ (binary variables) and $\mathcal{X} = \{-1,1\}^n$ (spin variables). Let $\mathcal{F}$ denote the feature space and Φ the feature map from $\mathcal{X}$ to $\mathcal{F}$. Since we aim at modeling and simulating correlated variables, let $\Phi(x)$ be the vector made of all x_i and all $x_i x_j$, $i < j$, and let $\mathcal{F} = \mathbb{R}^{n(n+1)/2}$. Let p be some probability distribution on $\mathcal{X}$. Let μ denote the expectation of Φ w.r.t. p or $\mu = \mathbb{E}_{x\sim p}\,\Phi(x)$. In particular, $\mu_i = \mathbb{E}_{x\sim p}(x_i)$ and $\mu_{ij} = \mathbb{E}_{x\sim p}(x_i x_j)$. The moment matching problem consists in sampling from a probability distribution q on $\mathcal{X}$ such that expectations of Φ w.r.t. p and q coincide. Herding is an approximate solution to the moment matching problem.

The empirical moment of a sequence $(x^k)_{k=1}^t$ is defined by $\tilde{\mu}^t = \frac{1}{t}\sum_{k=1}^t \Phi(x^k)$. Given $(x^k)_{k=1}^t$, define $\tilde{\mu}^{t+1}(x) = \frac{1}{t+1}\left(\sum_{k=1}^t \Phi(x^k) + \Phi(x)\right)$. Herding generates samples by minimizing the distance between empirical and target moments. More precisely, define the sequence $(x^t)_{t=1}^\infty$ by $x^{t+1} \in \arg\min_x \left\|\mu - \tilde{\mu}^{t+1}(x)\right\|$. This dynamical system is deterministic, insofar as all minimizers x^t are unique. Let us write $\mu - \tilde{\mu}^{t+1}(x) = \frac{\Delta\mu^t - \Phi(x)}{t+1}$, where $\Delta\mu^t = (t+1)\mu - t\tilde{\mu}^t$. Then, $x^{t+1} \in \arg\min_x \left\|\Delta\mu^t - \Phi(x)\right\|$ and $\Delta\mu^{t+1} = \Delta\mu^t + \mu - \Phi(x_{t+1})$. The original herding algorithm [7, 6] is defined by $x^{t+1} \in \arg\max_x \langle \Delta\mu^t, \Phi(x)\rangle$. Its theoretical analysis provides a $O(1/t)$ bound on moment discrepancy [7, 4, 1], instead of $O(1/\sqrt{t})$ in the case of i.i.d. random vectors distributed according to p. Both herding algorithms are equivalent if $\|\Phi(x)\|$ does not depend on x. This is the case if $\mathcal{X} = \{-1,1\}^n$ but not if $\mathcal{X} = \{0,1\}^n$. We will consider three different herding algorithms: $h_{\min}$ (binary variables, norm minimization), $h_{\max}$ (binary variables, inner product maximization), and $h_\pm$ (spin variables).

Minimizing $\left\|\Delta\mu^t - \Phi(x)\right\|$ over $\{0,1\}^n$ is NP-hard, hence the need for an approximation algorithm for $h_{\min}$. Let $\nu = 1 - 2\Delta\mu^t$. Up to some constant, $\left\|\Delta\mu^t - \Phi(x)\right\|^2 = \sum_i s_i x_i$, where $s_i = \nu_i + \sum_{j<i} \nu_{ij} x_j$ does not depend on x_j for $j \geq i$. The greedy algorithm determines each component from x_1 to x_n. In order to compute x_i given $(x_j)_{j=1}^{i-1}$, it minimizes the partial sum $\sum_{j=1}^i s_j x_j$, which reduces to minimizing $s_i x_i$ alone: if $s_i < 0$ then $x_i = 1$ else $x_i = 0$. We apply the same method to $h_{\max}$ and $h_\pm$. Up to some constant, $\langle \Delta\mu^t, \Phi(x)\rangle = \sum_i s_i x_i$, where $s_i = \Delta\mu_i^t + \sum_{j<i} \Delta\mu_{ij}^t x_j$. We also consider the following randomization technique: at each iteration, the algorithm uniformly samples a permutation π and computes each component from $x_{\pi(1)}$ to $x_{\pi(n)}$. Herding algorithms with random permutations will be denoted by $h_{\max}^p$, $h_{\min}^p$, and $h_\pm^p$ respectively.

Algorithm	J_5	J_{10}	FP_5	FP_{10}	BQP	MS	NK	LA	SC	SCs	EP
SA	95	90	194	189	727.89	956	97.90	4.72	1.88	0.85	1.03×10^{-23}
LS	95	90	118	24	718.11	954	91.27	4.42	2.02	0.91	4.82×10^{-24}
PBIL	100	90	194	189	681.65	954	91.64	3.85	0.08	0.06	6.97×10^{-24}
HEA ($h_{\max}$)	100	90	194	100	723.08	949	84.15	4.09	1.63	0.87	2.10×10^{-23}
HEA ($h^p_{\max}$)	100	100	194	189	719.90	955	93.61	3.89	0.20	0.28	2.86×10^{-23}
HEA ($h_{\min}$)	95	90	194	100	685.27	952	81.74	3.76	0.86	0.74	6.53×10^{-23}
HEA ($h^p_{\min}$)	100	100	194	189	716.51	953	93.61	3.98	0.22	0.39	3.99×10^{-23}
HEA ($h_{\pm}$)	95	90	161	100	698.63	955	88.55	3.41	0.33	1.42	1.57×10^{-22}
HEA ($h^p_{\pm}$)	100	90	194	189	725.00	955	93.50	4.07	0.50	0.30	2.03×10^{-23}

Table 1: Results. For stochastic algorithms, the median of 20 extrema is displayed.

3. HEA

The target moment μ is initialized to the moment of the uniform distribution on $\mathcal{X}$. An HEA iteration is as follows: 1. Sample P individuals by means of herding. 2. Evaluate the population with the fitness function. 3. Select P' individuals according to their ranks. 4. Compute their empirical moment μ^*. 5. Update the target moment according to $\mu \leftarrow \mu + \alpha(\mu^* - \mu)$, where α is the learning rate.

As a consequence of the evolutionary process, the target moment is now time-varying. Experimentally, we observe a divergence (rapid increase) of the moment discrepancy in the case of HEA algorithms with a *constant* learning rate. HEA with $h_{\min}$, $h_{\max}$, $h_{\pm}$, and $h^p_{\pm}$ diverge at iteration $k = 1/\alpha$ whereas HEA with $h^p_{\min}$ and $h^p_{\max}$ diverge at $k = 1/4\alpha$. Moreover, HEA algorithms almost never improve on the best solution after the divergence. This suggests to set $\alpha = 1/K$ or $1/4K$ where K is the total number of iterations. Alternatively, we propose an exponentially decreasing learning rate $\alpha_k = \alpha_0 e^{-k/\tau}$, where $\alpha_0 = e/\tau$ or $e/4\tau$.

4. EXPERIMENTS

We have considered different classes of fitness functions (to be maximized unless otherwise stated): *Jump* (see below, $t = 5$ or 10), *Four Peaks* [2] (threshold $t = 5$ or 10), *Boolean Quadratic Programming* (random instance), *MaxSAT* (random instance ms-s3v80c1000-1 from Max-SAT 2013), *NK landscapes* (random instance, $k = 3$), *Low Autocorrelation Binary String* (see below), *Summation Cancellation* [3] (minimization), *Summation Cancellation with sinus* [5] (minimization), and *Equal Products* [3] (minimization, random instance). The Jump function is defined by $f(x) = 0$ if $n - t < \|x\|_1 < n$, $f(x) = \|x\|_1$ otherwise. The LABS function is defined by $f(x) = F(s)$, where $s = 2x - 1 \in \{-1, 1\}^n$, $F(s) = n^2/2E(s)$, $E(s) = \sum_{k=1}^{n-1} C_k(s)^2$, and $C_k(s) = \sum_{i=1}^{n-k} s_i s_{i+k}$. The dimension of the search space is $n = 100$ for all problems except MaxSAT ($n = 80$) and Summation Cancellation problems ($n = 99$).

Each algorithm has been allocated 2×10^5 function evaluations. For randomized HEA algorithms, the learning rate $\alpha = 5 \times 10^{-5}$ is constant. For deterministic HEA algorithms, the learning rate is exponentially decreasing with $\tau = 10^4$ (time constant). For all HEA algorithms, $P = 10$ and $P' = 1$. For PBIL [2], $\alpha = 5 \times 10^{-3}$, $P = 10$ and $P' = 1$. For simulated annealing, we have chosen a geometric cooling. The rate of the inverse temperature β is 1.05 and $\beta_0 = \log(0.6)/\Delta f$, where Δf is the average fitness function variation over 100 random fitness decreasing transitions. The temperature is kept constant until 50 fitness decreasing transitions have been accepted. For (stochastic) local search, an approximate local maximum has been reached after 50 consecutive rejected moves. The algorithm is then randomly restarted until the maximum number of function evaluations has been reached. All stochastic algorithms have been run 20 times. The results are shown in Tab. 1. To sum up, with the convention win-draw-lose across all 11 fitness functions, the score of HEA with $h^p_{\max}$ is 5-3-3 against PBIL, 4-2-5 against SA, and 9-0-2 against LS.

5. CONCLUSION

We have introduced HEA, a family of evolutionary algorithms which use herding to sample the search space. HEA shows promising results which need to be confirmed by additional experiments. A better understanding of the learning rate control and the greedy approximation of herding is required to improve the performances of HEA. Finally, we suggest two potential research directions. One is to extend HEA to continuous search spaces, as herding itself has already been [4]. Another one is to consider feature spaces other than those of first and second moments.

6. REFERENCES

[1] F. Bach, S. Lacoste-Julien, and G. Obozinski. On the equivalence between herding and conditional gradient algorithms. In *International Conference on Machine Learning*, 2012.

[2] S. Baluja and R. Caruana. Removing the genetics from the standard genetic algorithm. In *Proceedings of the 12th Annual Conference on Machine Learning*, pages 38–46, 1995.

[3] S. Baluja and S. Davies. Using optimal dependency-trees for combinatorial optimization: learning the structure of the search space. Technical Report CMU-CS-97-107, Carnegie-Mellon University, January 1997.

[4] Y. Chen, M. Welling, and A. Smola. Super-samples from kernel herding. In *Uncertainty in Artificial Intelligence*, pages 109–116, 2010.

[5] M. Sebag and M. Schoenauer. A society of hill-climbers. In *Proc. IEEE Int. Conf. on Evolutionary Computation*, pages 319–324, Indianapolis, April 1997.

[6] M. Welling. Herding dynamic weights for partially observed random field models. In *Uncertainty in Artificial Intelligence*, pages 599–606, 2009.

[7] M. Welling. Herding dynamic weights to learn. In *International Conference on Machine Learning*, pages 1121–1128, 2009.