comparison-based algorithms are robust and randomized algorithms are anytime.

Sylvain Gelly, Sylvie Ruette, Olivier Teytaud

Comparison-based algorithms are robust and randomized algorithms are anytime

Sylvain Gelly gelly@lri.fr
Equipe TAO (Inria), LRI, UMR 8623 (CNRS - Université Paris-Sud), bat. 490 Université Paris-Sud 91405 Orsay Cedex France

Sylvie Ruette sylvie.ruette@math.u-psud.fr
Laboratoire de Mathématiques, CNRS UMR 8628, Bâtiment 425, Université Paris-Sud, 91405 Orsay cedex, France

Olivier Teytaud olivier.teytaud@inria.fr
Equipe TAO (Inria), LRI, UMR 8623 (CNRS - Université Paris-Sud), bat. 490 Université Paris-Sud 91405 Orsay Cedex France

Abstract

Randomized search heuristics (e.g., evolutionary algorithms, simulated annealing etc.) are very appealing to practitioners, they are easy to implement and usually provide good performance. The theoretical analysis of these algorithms usually focuses on convergence rates. This paper presents a mathematical study of randomized search heuristics which use comparison based selection mechanism. The two main results are: (i) comparison-based algorithms are the best algorithms for some robustness criteria, (ii) introducing randomness in the choice of offspring improves the anytime behavior of the algorithm. An original Estimation of Distribution Algorithm combining (i) and (ii) is proposed and successfully experimented.

Keywords

Theory, robust optimization, randomized search heuristics, anytime optimization, estimation of distribution algorithms.

1 Introduction

Many evolutionary optimization tools are based on two ideas: (i) to use random numbers, (ii) to use only comparisons between fitness values, and not the fitness values themselves. We study these two principles from two different points of view, namely robustness and anytime behavior.

We first introduce some notations for the sake of clarity, and thereafter we present the state of the art for comparison-based algorithms and for the anytime behavior of randomized algorithms. We then give an overview of the paper.

Definitions and notations

We will work on families of *optimization algorithms*. Thus we need to formalize optimization algorithms. Let Opt be a map from $\{\emptyset\} \cup \bigcup_{n\in\mathbb{N}}(D \times \mathbb{R})^n$ to D^2, where D is some fixed domain. Given a real-valued fitness function f defined on D, we define the following sequence:

$$\begin{aligned}(x_1, x'_1) &= Opt(),\\ \forall n \geq 1,\ (x_{n+1}, x'_{n+1}) &= Opt(x_1, y_1, x_2, y_2 \ldots, x_n, y_n),\\ \forall n \geq 1,\ y_n &= f(x_n),\end{aligned}$$

where $x_i, x'_i \in D$ and $y_i \in \mathbb{R}$. The optimization algorithm associated to Opt is the mapping $f \mapsto Opt_f = (x_n, x'_n)_{n\in\mathbb{N}}$; this mapping will be called Opt too. The x_n are the visited points, and the x'_n are the approximations of the optimum suggested by the algorithm; let $(Opt_f)_n = x'_n$ be the point suggested at the n-th step. Usually, but not necessarily, $x'_{n+1} = x_{i(n)}$ with $i(n) = \arg\min_{i\leq n} y_i$. The performance is evaluated e.g. by $\|x'_N - \arg\min f\|^2$, f being assumed to have one global minimum (this criterion is sometimes averaged on f, or considered for the worst case among some family of fitness functions).

In some cases, we consider an algorithm designed for N iterations and a performance criterion depending only on the last epoch x_N (e.g. $\|x_N - \arg\min f\|^2$). In this case, using both x_n and x'_n is useless; we then consider a map $Opt\colon \{\emptyset\} \cup \bigcup_{n=1}^{N}(D \times \mathbb{R})^n \to D$ and we define $Opt_f = (x_n)_{1\leq n\leq N}$, where the sequence is given by

$$x_1 = Opt() \text{ and } \forall n \in \{1, \ldots, N-1\},\ x_{n+1} = Opt(x_1, f(x_1), \ldots, x_n, f(x_n)).$$

We introduce the set of increasing functions $G = \{g : \mathbb{R} \to \mathbb{R}; \forall (x, y) \in \mathbb{R}^2, x < y \Rightarrow g(x) < g(y)\}$. An algorithm $f \mapsto Opt_f$ is said *comparison-based* if, for every $g \in G$, the output of the map Opt is the same if the values $y_1, \ldots, y_n$ in its inputs are replaced by $g(y_1), \ldots, g(y_n)$, that is, for all $x_1, \ldots, x_n \in D, y_1, \ldots, y_n \in \mathbb{R}$ and $g \in G$, $Opt(x_1, g(y_1), \ldots, x_n, g(y_n)) = Opt(x_1, y_1, \ldots, x_n, y_n)$. Since a comparison-based algorithm identically acts on f and $g \circ f$ for all $g \in G$, it will be natural to consider sets F of fitness functions that are stable by composition with increasing functions, that is, $f \in F \Rightarrow \forall g \in G,\ g \circ f \in F$. Many evolutionary algorithms are comparison-based, but not all of them (e.g., fitness-proportional selection is not comparison-based). Algorithms that are not comparison-based are called *fitness-based*.

We consider *robustness properties*. In the robust case, the quality of an optimizer is estimated by its worst case among a family of functions: if F is a space of fitness functions, N is a number of iterations and x'_N is the N-th estimate of the optimum proposed by the algorithm, the quality criterion is $\sup_{f\in F} \|x'_N - x^*(f)\|$ where $x^*(f)$ is the optimum of the fitness function f. We will see that, for this robust criterion, if F is stable by composition with G, then every optimization algorithm can be replaced without loss of efficiency by a comparison-based algorithm (Section 2). Please notice

that this form of robustness can be defined on any domain (finite or not), and does not imply that the set of fitness-functions is only one function and its compositions with increasing transformations. Any set of functions H can be embedded in a set of functions F stable by composition with G by letting $F = \{g \circ h; (g, h) \in G \times H\}$.

We consider also a *greedy criterion*. A greedy algorithm is an algorithm in which each iteration is chosen as if it was the last one: if the points $x_1, \ldots, x_n$ are already chosen (and hence fixed with regard to the future iterations), the next point x_{n+1} is chosen in order to optimize the value x_{n+1} itself, regardless of its influence on the choice of future values x_i with $i > n + 1$. Formally, $x_1, \ldots, x_n$ are fixed, their fitness values $y_1 = f(x_1, w), \ldots, y_n = f(x_n, w)$ depend on the function f which is a random variable, and the greedy quality criterion at the $(n + 1)$-th step is the expected distance $\mathbb{E}\left(\|x_{n+1} - x^*(f)\|^2\right)$. Therefore the greedy optimal algorithm chooses the point $x_{n+1} = Opt(x_1, y_1, \ldots, x_n, y_n)$ as follows:

$$x_{n+1} \in \arg\min_x \mathbb{E}\left(\|x - x^*(f)\|^2 \,|f(x_1, w) = y_1, \ldots, f(x_n, w) = y_n\right).$$

But other criteria could be considered as well. In this paper, we will consider a robust and greedy criterion:

$$\mathbb{E}\left(\sup_{g \in G} \|x_{n+1} - x^*(g \circ f)\|^2\right). \tag{1}$$

We will see in Equation (13) how to define the robust and greedy optimal algorithm.

Finally, we consider *anytime properties*. An algorithm is said to have a good anytime behavior if it runs without knowing in advance its computation time (or its number of iterations) and if it is reasonably good whatever may be the time at which it is stopped. We will see that some algorithms, optimal for each iteration independently (i.e. optimal for the greedy criterion, presented above), are indeed very poor for the long-term behavior (in that sense, they are not anytime at all; see Section 3), and are in particular strongly outperformed for most numbers of iterations by their randomized counterparts.

Comparison-based algorithms

In evolutionary computation, algorithms are not systematically comparison-based, but this property holds more and more often. (14) uses fitness-proportional selection, and the simple-GA popularized in (11) uses more than comparisons, but suffers from various drawbacks, among which super-individuals (leading to premature convergence), and the puzzling non-invariance when constants are added to the fitness. The first part of this paper deals with the advantages of comparison-based methods in terms of robustness.

Many algorithms, in spite of this restriction (they only use comparisons, and loose all other information), have been proved to be linear in the sense that the $\log$-distance to the optimum converges to $-\infty$ linearly in the number of iterations (i.e. $\frac{1}{n} \log \|x'_n - x^*(f)\|$ converges to a negative constant); see e.g. (1; 2; 7; 20). Therefore

these algorithms have a reasonably good convergence rate. Some linear lower bounds also exist in various cases (16; 23), and they show that the constant in the linear convergence decreases to 0 linearly with the inverse of the dimension. We introduce below the state of the art of the analysis of comparison-based methods. We then prove the optimality of comparison-based methods in a robust sense (Section 2).

In (23), it is shown that comparison-based algorithms can at best be linear w.r.t the number of comparisons, with a constant $1 - O(1/d)$ as the dimension d increases to ∞, even for very easy fitness functions. As the computational cost is at least linear in the number of comparisons, this result impacts the computation cost of these algorithms. In this paper we prove that comparison-based algorithms are, in spite of the limitation of the $1 - O(1/d)$ for the convergence rate with respect to the number of comparisons, optimal in a robust sense. This generalizes the known fact that some increasing transformations of e.g. the sphere function are much harder than the sphere function itself for e.g. Newton's methods. Typically $x \mapsto \sqrt{\|x\|}$ is much harder than the sphere function for the Newton-algorithm; also, the Newton-algorithm is much worse than random-search on this function.

Anytime behavior of randomized offsprings

Whereas the idea of comparison-based algorithms is now widely accepted even outside the evolutionary community (see e.g. (6)), randomized algorithms are not well accepted yet in many fields of optimization. Intuitively, randomization does not look appealing (and is often used as an argument against evolutionary algorithms). How random offsprings can perform better choices than a mathematically determined deterministic algorithm? In order to address this question, we look for mathematically optimal algorithms for a given distribution of fitness functions. Unfortunately, the optimal algorithm depends on the number of iterations (Section 3.1) and is not tractable. Due to the complexity of this optimal algorithm, pertinent experiments would cost much time, and thus we have no experiments to decide whether this "perfect" algorithm is strongly suboptimal or almost optimal for a number of iterations that is not the one expected by the algorithm.

Therefore, we focus on a simpler case, namely the greedy-optimal algorithm. This means that each iteration is done as if it was the last one: we just choose the n-th point x_n by optimizing the expected squared distance from x_n to the optimum, conditionally to past observations[1]. This greedy-optimal algorithm is tractable thanks to billiard-techniques (Section 3.2). This is in particular the intuitive principle behind many optimization tools: estimating the position of the optimum, and choosing this value for the next iteration, regardless of future iterations (see however the idea of conditioning-preservation in (6) and references therein; see also (Villemonteix et al.)). In particular, such greedy-optimal algorithms are deterministic. We mathematically derive a greedy-optimal offspring generation algorithm (Theorem 2 for $\lambda = 1$ and Theorem 2′ for $\lambda > 1$,

[1] This assumes that the distribution of fitness functions is known in advance.

where λ is the size of the offspring), and compare this mathematically optimal (in a greedy sense) algorithm, that we call BEDA, to its randomized counterpart, that we call BREDA. The precise definitions of BEDA and BREDA are in Section 3.2. The interesting conclusion is that the randomized counterpart is much better than the greedy-optimal deterministic one. For the greedy criterion, BEDA is better (as it is mathematically optimal!), but for almost all values of the number of iterations the randomized BREDA is in fact much better. In this sense, our results show that randomized algorithms have a better anytime behavior than some (provably deterministic) greedy-optimal counterparts. The anytime efficiency of evolutionary algorithms has been pointed out in e.g. (8, chap. 2); see also (21).

Overview of the paper

In Section 2 we show that comparison-based methods are indeed optimal for the robust case defined above (worst case in F stable by composition with G). We use a corollary of this result to build fitness functions that are easy for comparison-based methods and that are very hard for other methods. Experiments illustrate this method (Section 4.1).

In Section 3.1 we show that some optimality criterion and corresponding optimal algorithms can be derived for a fixed number of iterations and a given distribution on fitness functions. However, they are not easily tractable and depend on the number of iterations. Then we define a robust and greedy approximation, called BEDA, of this algorithm; "robust" means that we consider the worst case among some family of functions and "greedy" means that the algorithm is independent of the number of iterations (and thus it is much more tractable). We show that this algorithm BEDA is optimal for the robust and greedy criterion, formalized in Equation (1) (see Equation (13) in Section 3.2 for the definition of this greedy-optimal algorithm).

We experiment the algorithms in Section 4.2 and conclude that there exists a strong gap between optimal optimization algorithms and greedy-optimal optimization algorithms: the greedy-optimal algorithms are far from being the best possible ones. An important point is that this gap is partially filled by randomized counterparts, called BREDA (an idealized form of Estimation-Of-Distribution algorithms), of the greedy optimal algorithms BEDA.

These results show that randomized comparison-based algorithms are not so bad, in spite of their limitations in terms of convergence rate: (i) they are optimal in the robust sense (Theorem 1), and (ii) randomized counterparts provide a much better approximation to optimality than the deterministic greedy approach (Section 4.2), leading to a reasonable anytime behavior without any exploration/exploitation parameter or a priori given number of fitness-evaluations. As a by-product of this theoretical analysis, the new BREDA algorithm defined in the following of the paper is empirically very efficient at least for a small number of iterations, and it has no free parameter, thanks to an original formalization (called BREDA) of the Estimation of Distribution Algorithms. The offsprings of BREDA are generated accordingly to the conditional distribution of

the optimum, conditionally to previously visited points and their fitness values; this is an idealized form of Estimation Of Distribution Algorithm (EDA), and it can be implemented through billiards. By the way, BREDA, an idealized form of EDA (inspired by (10) in the field of EA but also (Villemonteix et al.) which uses Kriging in an original way) has very impressive results in favor of EDA, in particular when dimension increases.

2 Some robustness results for comparison-based methods

Notations. Recall that G is the set of increasing mappings from $\mathbb{R}$ to $\mathbb{R}$. Let G^0 denote the set of increasing continuous functions from $\mathbb{R}$ to $\mathbb{R}$. We let $sign(x) = 1$ if $x > 0$, $sign(x) = -1$ if $x < 0$ and $sign(0) = 0$. If A, B are two sets and b is a point, we let $d(A, b) = \inf_{a \in A} \|a - b\|$ and $d(A, B) = \inf_{a \in A, b \in B} \|a - b\|$.

It is known that Newton-based algorithms do not necessarily converge on non-convex functions. It is also known that even in quasi-convex cases like $x \mapsto \sqrt{\|x\|}$, Newton's algorithm does not converge, whereas comparison-based algorithms, in spite of their relative slowness, have exactly the same behavior on $x \mapsto \sqrt{\|x\|}$ as on $x \mapsto \|x\|^2$. We provide a wide generalization of this type of result in the theorem below. We will consider the worst case among functions in a set of fitness-functions; this is a classical robustness criterion ((19)).

Theorem 1 below states that, for this robustness criterion, for every optimization algorithm Opt, there is another optimization algorithm Opt' that has the same efficiency and such that Opt' is comparison-based. More precisely, we state that if for some N and ϵ, Opt ensures that the N-th iteration is the optimum within precision ϵ, then there exists Opt' which is comparison-based and ensures the same precision. Note that the following theorem does not assume anything on the domain D, which can be continuous or discrete.

Theorem 1: Optimality of comparison-based algorithm in the robust case. *Consider F a space of real-valued functions defined on a given domain D such that each $f \in F$ has one and only one global minimum, and assume that, for all $f \in F$ and all $g \in G$, $g \circ f$ belongs to F. Consider a deterministic optimization algorithm $Opt : f \mapsto Opt_f = (x_n, x'_n)_{n \in \mathbb{N}}$ (see the definitions in Section 1). We consider $x'_N = (Opt_f)_N$ as a function of f. Assume that, for some $\epsilon > 0$, there exists an integer N such that,*

$$\forall f \in F,\ \|(Opt_f)_N - \arg\min f\| \leq \epsilon. \tag{2}$$

Then, there exists a deterministic algorithm Opt' that only depends on comparisons, in the sense that

$$\begin{aligned}&(\forall i, j,\ sign(y_i - y_j) = sign(y'_i - y'_j))\\ &\Longrightarrow Opt'(x_1, y_1, \ldots, x_n, y_n) = Opt'(x_1, y'_1, \ldots, x_n, y'_n),\end{aligned}$$

and Opt' is such that Equation (2) holds with the same ϵ and the same N, i.e.,

$$\forall f \in F,\ \|(Opt'_f)_N - \arg\min f\| \leq \epsilon. \tag{3}$$

Proof. First, we define a map $Opt' : \{\emptyset\} \cup \bigcup_{n\in\mathbb{N}} (D \times \mathbb{R})^n \to D^2$. We set $Opt'() = (x_1, x'_1)$ and, $\forall n \geq 1, \forall x_1, \ldots, x_n \in D, \forall z_1, \ldots, z_n \in \mathbb{R}$, we define $Opt'(x_1, z_1, \ldots, x_n, z_n) = Opt(x_1, y_1, \ldots, x_n, y_n)$, where the y_i are defined by induction as follows:

- $y_1 = 0$.
- For $i \in \{2, \ldots, n\}$, y_i

$$\begin{aligned} &= y_j \text{ if } z_i = z_j \text{ for some } j < i, \\ &= \max_{j<i} y_j + 1/i^2 \text{ if for all } j < i, z_j < z_i, \\ &= \min_{j<i} y_j - 1/i^2 \text{ if for all } j < i, z_j > z_i, \\ &= \frac{1}{2}(y_j + y_k) \text{ if for some } j, k < i, z_j < z_i < z_k \\ &\quad \text{and for all } p \notin \{i, j, k\}, z_p \leq z_j \text{ or } z_p \geq z_k. \end{aligned}$$

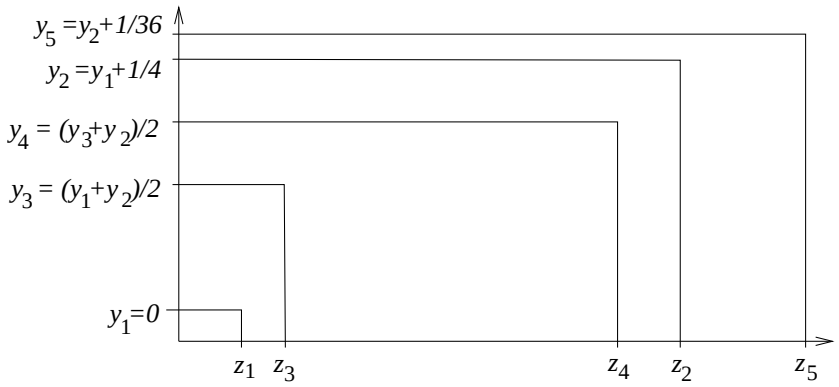

Figure 1: The points $z_1, \ldots, z_6$ are given, the points $y_1, \ldots, y_5$ are built from $z_1, \ldots, z_5$ as in the proof of Theorem 1.

Figure 1 illustrates this construction. We see that y_i only depends on $\{sign(z_j - z_k)\}_{j,k\leq i}$. This means that Opt' only depends on comparisons. It remains to prove that Opt' verifies Equation (3).

Let $f \in F$. We define inductively $x_1, \ldots, x_N$, $x'_1, \ldots, x'_N$, $z_1, \ldots, z_N$ by $(x_1, x'_1) = Opt'()$, $z_i = f(x_i)$, $(x_i, x'_i) = Opt'(x_1, z_1, \ldots, x_{i-1}, z_{i-1})$. Consider $y_1, \ldots, y_N$ defined as above. By construction, for all $i, j \in \{1, \ldots, N\}$, $sign(y_i - y_j) = sign(z_i - z_j)$. Since the two sets of points are finite and equally ordered, there exists $g \in G$, such that, for all $i \in \{1, \ldots, N\}$, $g(z_i) = y_i$. Let f' denote the function $g \circ f$, which belongs to F by assumption.

We now consider $\tilde{x}_1, \ldots, \tilde{x}_N$, $\tilde{x}'_1, \ldots, \tilde{x}'_N$, $\tilde{y}_1, \ldots, \tilde{y}_N$ defined inductively by $(\tilde{x}_1, \tilde{x}'_1) = Opt()$, $\tilde{y}_i = f'(\tilde{x}_i)$, $(\tilde{x}_i, \tilde{x}'_i) = Opt(\tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_{i-1}, \tilde{y}_{i-1})$. By construction, $\tilde{x}_i = x_i$, $\tilde{x}'_i = x'_i$ and $\tilde{y}_i = y_i$ for all $i \in \{1, \ldots, N\}$. This shows that $Opt'_f = Opt_{f'}$. This provides the expected result. □

Corollary. *In Theorem 1, it is sufficient to assume that F is stable by C^∞ mappings of G (i.e., for every $f \in F$ and every map g such that g is increasing and infinitely differentiable, one has $g \circ f \in F$).*

Proof. In the proof of Theorem 1, g can be chosen C^∞. □

Many algorithms ensure quadratic or superlinear convergence rates. This is not the case for most evolution strategies. Therefore, one could think that for convergence rates, the result above does not hold. Interestingly, however, for the worst-case on increasing transformations of the fitness function, the result above also holds in the sense of convergence rates as defined below.

Corollary: asymptotic convergence rate. *Consider F a space of functions defined on a domain D such that each $f \in F$ has one and only one global minimum. Suppose that for some deterministic algorithm Opt and some sequence $(\epsilon_n)_{n\in\mathbb{N}}$, one has:*

a) for all $f \in F$ and all integers N, $\|(Opt_f)_N - \arg\min f\| \leq \epsilon_N$;

b) for all $f \in F$, $\lim_n f(x_n) = \inf f$, where $Opt_f = (x_n, x'_n)_{n\in\mathbb{N}}$;

c) for all $f \in F$ and all $g \in G^0$, $g \circ f \in F$.

Then, (a) also holds for some deterministic algorithm Opt' that only depends on comparisons.

Proof. We consider Opt' and $(y_i)_{i\in\mathbb{N}}$ as defined in Theorem 1 above. We are going to show that there exists $g \in G^0$ such that

$$\forall i \in \mathbb{N},\ g(f(x_i)) = y_i \tag{4}$$

(whereas in Theorem 1, we only need such a property for $i \in \{1, \dots, N\}$).

We define $z_n = f(x_n)$ and $z_\infty = \lim z_n$, i.e. $z_\infty = \inf f$ by (b). The sequences of points $(y_i)_{i\in\mathbb{N}}$ and $(z_i)_{i\in\mathbb{N}}$ are equally ordered by construction. We let $y_\infty = \inf y_n$. By construction, y_n is lower bounded because $y_n \geq \inf_{i<n} y_i - 1/n^2$, $y_1 = 0$, and thus $y_n \geq -\sum_{i\geq 1} 1/i^2 > -\infty$. Let g_n be the piecewise linear interpolation of the points $\{(z_\infty, y_\infty), (z_1, y_1), \dots, (z_n, y_n), (z_M, y_M)\}$, where $z_M = \max_i z_i$ and $y_M = \max_i y_i$. All the maps g_n are defined on $[z_\infty, z_M]$.

Step 1: We show that g_n converges to some limit g. For all $\epsilon > 0$, $\exists n_0 \geq 1; \forall k \geq n_0, z_k \leq z_\infty + \epsilon$. Let $n \geq \max(n_0, 1/\epsilon)$. Because of (b), we can define $n_\epsilon \in \arg\max_{k\geq n} z_k$. Then for all $k \geq n_\epsilon$, $g_{n_\epsilon} = g_k$ on $[z_{n_\epsilon}, z_M] \supset [z_\infty + \epsilon, z_M]$. Thus the continuous maps g_n converge pointwise to some map g, and

$$\forall k \geq n_\epsilon, \forall z \in [z_{n_\epsilon}, z_M], g_k(z) = g(z), \tag{5}$$

$$\text{with } \lim_{\epsilon\to 0} n_\epsilon = \infty \tag{6}$$

$$\text{and } z_{n_\epsilon} \leq z_\infty + \epsilon.$$

Step 2: We show that g is increasing and g_n converges uniformly to g. Equation (5) implies that g is increasing on $[z_\infty, z_M]$. We are going to show that g_n uniformly

converges to g. We will first prove Step 2a below. Then we will show in Step 2b that Step 2a implies the uniform convergence of g_n to g.

Step 2a: We show that $\lim_{n\to\infty} y_n = y_\infty$. If $\exists m, \forall n \geq m, z_n = z_\infty$, then $\forall n \geq m, y_n = y_m$; thus $y_\infty = y_m = \lim y_n$ and Step 2a holds. In the following, we suppose that we are not in this case, that is:

$$\exists \text{ infinitely many } n; z_n \neq z_\infty. \tag{7}$$

Before proving Step 2a, we are going to prove Equation (8):

$$\exists m; y_m < y_\infty + \epsilon \text{ and } z_m > z_\infty. \tag{8}$$

Let us show Equation (8):

- If $\forall n, z_n \neq z_\infty$ then, by definition of y_∞, Equation (8) holds.
- Otherwise, $\exists n_0; z_{n_0} = z_\infty$. Then, thanks to Equation (7), we can define by induction an increasing sequence $(n_i)_{i\geq 0}$ (n_0 has already been defined) such that $n_1 = \min\{n > n_0; z_n > z_\infty\}$ and, for all $i \geq 2$, n_i is minimal under the following constraint:
$$z_\infty < z_{n_i} < z_{n_{i-1}}.$$
Then, by construction, $y_{n_i} = \frac{y_{n_{i-1}} + y_\infty}{2}$, and Equation (8) holds with $m = n_i$ for i sufficiently large.

We now have to prove that Equation (8) implies Step 2a. Let m verify Equation (8). Then $z_m > z_\infty$ and $z_n \to z_\infty$. Therefore, $\exists p; \forall n \geq p, z_n \leq z_m$. So, $\forall n \geq p, y_n \leq y_m \leq y_\infty + \epsilon$. This concludes Step 2a.

Step 2b: we show that g_n converges uniformly to g. Since $n_\epsilon \to \infty$ as $\epsilon \to 0$ by Equation (6), and according to Step 2a,

$$\forall \epsilon' > 0, \exists \epsilon > 0, \forall n \geq n_\epsilon, y_n \leq y_{n_\epsilon} \leq y_\infty + \epsilon'.$$

Then, $\forall n \geq n_\epsilon$,

$$\begin{gathered}\forall z \geq z_{n_\epsilon}, g_n(z) = g(z) \text{ by Equation } (5) \\ \text{and } \forall z \leq z_{n_\epsilon}, g_n(z) \leq g_n(z_{n_\epsilon}) = y_{n_\epsilon} \leq y_\infty + \epsilon'.\end{gathered}$$

This implies that g_n converges uniformly to g on $[z_\infty, z_M]$, which concludes the proof of Step 2b.

Step 3: conclusion of the proof. Step 2 states that g is increasing and implies that g is continuous because every g_i is continuous. Therefore, g is in G^0. By (c), $g \circ f$ is in F too. Therefore, (a) states that Opt reaches the convergence rate given by $(\epsilon_n)_{n\in\mathbb{N}}$ on $g \circ f$, i.e. that

$$\text{for all } N, \|(Opt_{g\circ f})_N - \arg\min g \circ f\| \leq \epsilon_N,$$

and Opt' has the same property on f, namely

$$\text{for all } N, \|(Opt'_f)_N - \arg\min f\| \leq \epsilon_N$$

This is the expected result. □

Remark. *In the corollary above, the sequence of precisions $(\epsilon_n)_{n\in\mathbb{N}}$ is ideally aimed to tend to 0; then the algorithm converges to $\arg\min f$. Assumption (b) can be replaced by the slightly weaker assumption that $\lim f(x_n) = \inf f(x_n)$. This assumption can not be removed; for example, suppose that $z_1 = f(x_1) = 0$ and $z_n = f(x_n) = 1 + 1/n$ for all $n \geq 2$. Then $z_n \to 1$, $y_1 = 0$ and $y_n = 1/2^n$ for all $n \geq 2$. Every non decreasing function g such that $g(z_n) = y_n$ is equal to 0 on $[0, 1]$, and so is not increasing.*

3 Greedy robust-optimal algorithms, and how random-offsprings outperform them

It is easy to see that randomized algorithms have some advantages in terms of robustness. For example, only randomized algorithms can ensure almost sure convergence to the essential infimum of any fitness function on $[0, 1]^d$. Random search is enough for that, whereas there is no deterministic algorithm that ensures this property. We will here focus on a more concrete case, by comparing

- the long-term behavior of random offsprings according to the distribution of the optimum conditionally to the visited points (i.e. algorithm BREDA, see algorithm 2), and
- the long-term behavior of deterministic greedy-optimal offsprings (i.e. algorithm BEDA, see algorithm 1).

We will conclude that somewhat surprisingly the long-term behavior is better in the random case (BREDA) than in the greedy-optimal deterministic algorithm (BEDA). Moreover, the first algorithm is an EDA, without any other parameter than an a priori distribution of fitness functions (and is experimentally very efficient in particular when the dimension increases).

In order to define the second algorithm previously mentioned, we must define formally greedy-optimality and design the corresponding greedy-optimal algorithm. This is done in Section 3.2 (Equation (11) and Theorems 2 and 2'). The formalization relies on a distribution of fitness functions; the algorithm is optimal only with respect to this distribution of fitness functions – as it has been emphasized in NFL-theorems (26), unless a priori information is available, there is no good nor bad algorithms on average, hence no optimal algorithm[2]. Can optimal algorithms be designed when such an a priori information is available? We will see in Section 3.1 that the answer is yes, but the optimal algorithm is hardly tractable. Then, in the spirit of Section 2, we will focus on robustness with respect to increasing transformations of the fitness-functions, in a simplified

[2]However some other a priori information (than the complete distribution of probability of the fitness function) can be provided and used efficiently; e.g., smoothness assumption, or differentiability, can lead to positive results; see e.g. (5; 25).

(greedy) sense for the sake of computational tractability. The greedy optimality means that each iteration will be chosen only in order to optimize its own efficiency, regardless of future iterations. We will see that this leads to reasonably tractable algorithms that can be implemented with billiards (Section 3.3). The greedy approximation is not pathological: for example, the very powerful Newton's methods use as next iteration the point that is estimated to be the best one, and not the one that provides the most meaningful information for future iterations. Greedy optimal-algorithms consider that each iteration has to be optimal, as if it was the last one. We will see in Section 4 that the greedy-approximation is far from being a weak assumption, and that the randomized counterpart, in spite of its suboptimality for the greedy criterion, is in fact much more efficient. Therefore, (somewhat counter-intuitively) random sampling is not so bad to provide meaningful long-term information (and to avoid bad conditioning). By the way, the resulting algorithm, called BREDA, has very good empirical results in simple cases.

3.1 An optimal algorithm for a given distribution of fitness functions

Consider a family of fitness-functions $f(.,w)$ defined on a same domain D, depending on a random parameter w and assume that each $f(.,w)$ has one and only one minimum at the point $x^*(w)$. Fix an integer N and consider a map $Opt : \{\emptyset\} \cup \bigcup_{n=1}^{N-1} (D \times \mathbb{R})^n \to D$. The associated optimization algorithm is $Opt : f(.,w) \mapsto (x_1, \ldots, x_N)$, where the points $x_1, \ldots, x_N$ are defined by:

- $x_1 = Opt()$;
- for $n \in \{2, \ldots, N\}$, $x_n = Opt(x_1, f(x_1, w), \ldots, x_{n-1}, f(x_{n-1}, w))$.

For every algorithm Opt and $i \in \{0, \ldots, N-1\}$, let $Opt_{|i}$ denote the algorithm restricted to the generation of x_n for $n > i$. There is no difference between Opt and $Opt_{|i}$ except that $Opt_{|i}$ assumes that the i first points are already given. Formally, $Opt_{|i}$ is the map $(f(.,w), x_1, \ldots, x_i) \mapsto (x_{i+1}, \ldots, x_N)$, where $x_n = Opt(x_1, f(x_1, w), \ldots, x_{n-1}, f(x_{n-1}, w))$ for all $n \in \{i+1, \ldots, N\}$.

Choosing the best possible function Opt is exactly a problem of optimal sequential decisions with discrete time steps and finite horizons. The most classical tool for such problems is called *Bellman's optimality principle* (3; 4), which states that the algorithm defined below is optimal.

Let $V(x_1, y_1, \ldots, x_i, y_i) = \inf_{Opt_{|i}} \mathbb{E}\left(\|x_N - x^*(w)\|^2 | \forall j \leq i, y_j = f(x_j, w)\right)$. Bellman's optimality principle states that the following function Opt is optimal, i.e. minimizes[3] $\mathbb{E}(\|x_N - x^*(w)\|^2)$:

$$\forall n \in \{2, \ldots, N\},\ Opt(x_1, y_1, \ldots, x_{n-1}, y_{n-1}) \in$$
$$\arg\min_x \mathbb{E}(V(x_1, y_1, \ldots, x_{n-1}, y_{n-1}, x, f(x,w)) | \forall i \leq n-1, y_i = f(x_i, w)) \qquad (9)$$

[3] We here study the mean squared distance to the optimum after N iterations; other criteria may be considered as well.

where V is computed by backward induction as follows:

$$V(x_1, y_1 \ldots, x_N, y_N) = \mathbb{E}(\|x_N - x^*(w)\|^2 | \forall i \in \{1, \ldots, N\}, f(x_i, w) = y_i),$$
$$V(x_1, \ldots, x_{n-1}, y_1, \ldots, y_{n-1}) = \inf_x \mathbb{E}_y V(x_1, \ldots, x_{n-1}, x, y_1, \ldots, y_{n-1}, y) \quad (10)$$

where y in (10) is distributed like $f(x, w)$ conditionally to $\forall i \leq n-1, f(x_i, w) = y_i$.

The algorithm described above is optimal (by Bellman's optimality principle), with respect to (i) a given number of iterations N, (ii) a criterion (mean quadratic fitness above, but many other criteria are possible), (iii) a distribution of fitness functions. One can envisage to use this algorithm only in the case of very expensive fitness-functions, because the internal cost of the algorithm is huge (in particular the backward-computation of V) and the algorithm has a very strong dependence in N. In the next section, we will focus, for the sake of tractability, on an easier form of optimality, *greedy-optimality*, in the case of robustness with respect to increasing transformations of the fitness function.

3.2 The greedy criterion

As the algorithm above is very complicated and beyond the scope of this paper, it will be the object of a future work. We are going to introduce a greedy-version of optimality and a robust criterion for this greedy-optimality as in Section 2. The greedy criterion refers to the fact that each offspring is generated in order to optimize the best point among the offspring, in an expectation sense defined precisely below.

Let us define mathematically the greedy criterion. Fix x_1, ..., x_n n points in a given bounded convex domain $D \subset \mathbb{R}^d$. Consider a probability distribution $P(w)$ on some domain W and a random variable $f(., w)$ which is a real-valued function defined on D. We assume that $f(., w)$ has a unique global minimum. For some $g \in G$ and all $i \in \{1, \ldots, n\}$, let $y_i = g(f(x_i, w))$ (implicitly, y_i depends on g). We want to choose a point $x_{n+1} = Opt(x_1, y_1, \ldots, x_n, y_n)$ in order to minimize the following quantity:

$$\mathbb{E}\left(\sup_{g \in G} \|x_{n+1} - \arg\min g \circ f(., w)\|^2\right). \quad (11)$$

This quality criterion if *robust* because we consider the worst case among increasing transformations, and it is *greedy* in the sense that (i) the n first values $x_1, \ldots, x_n$ have already been fixed, (ii) the quality criterion considered to choose the next point x_{n+1} depends only on the value of x_{n+1} itself. We point out that many derivations of constants in evolution strategies are based on optimal progress rate, which is exactly a greedy criterion. We consider below the optimal estimator of $\arg\min g \circ f(., w)$ for the greedy criterion (11). Notice that, for every $g \in G$, $\arg\min g \circ f(., w) = \arg\min f(., w)$; let $x^*(w)$ denote this common value of $\arg\min$.

Intuitively, Theorem 2 states that the optimal comparison-based greedy algorithm is deterministic and states explicitly what this optimal algorithm is. Afterwards, we will compare the long-term behavior of the optimal greedy algorithm BEDA defined

by Theorem 2 and the long-term behavior of its randomized counterpart BREDA. We show that the latter is much better. We conclude that randomized algorithms have a much better anytime behavior than the greedy-optimal algorithm: an algorithm is said to have a good anytime behavior if it runs without knowing in advance its number of iterations and if the value $\mathbb{E}(\|x_n - x^*(w)\|^2)$ is reasonably small whatever may be the iteration number n.

Theorem 2: Expliciting the optimal greedy algorithm. *Assume that for all w, $f(.,w)$ is a real-valued function defined on a bounded convex domain $D \subset \mathbb{R}^d$ which has one and only one global minimum, located at $x^*(w)$, i.e. $\forall w, \forall x \in \mathbb{R}^d, x \neq x^*(w) \Rightarrow f(x,w) > f(x^*(w),w)$. Fix an integer n. Then, for every map $Opt : (D \times \mathbb{R})^n \to D$, one has*

$$\mathbb{E} \sup_{g \in G} \|Opt(x_1, y_1(w,g), \ldots, x_n, y_n(w,g)) - x^*(w)\|^2 \geq \mathbb{E} \sup_{g \in G} \|x^{\sigma(w)} - x^*(w)\|^2 \quad (12)$$

where $y_i(w,g) = g(f(x_i,w))$, $\sigma(w) = (sign(f(x_i,w) - f(x_j,w)))_{(i,j) \in \{1,\ldots,n\}^2}$ and $x^{\sigma(w)} = \mathbb{E}(x^*(w)|\sigma(w))$ (i.e., $x^{\sigma(w)}$ is the expectation of $x^*(w)$ conditionally to the ranking $\sigma(w) = (sign(f(x_i,w) - f(x_j,w)))_{(i,j) \in \{1,\ldots,n\}^2}$). Moreover, equality in Equation (12) holds only if almost surely in w, $\forall g \in G$, $Opt(x_1, y_1(w,g) \ldots, x_n, y_n(w,g)) = x^{\sigma(w)}$.

Interpretation. Theorem 2 proposes a choice of the offspring, computed by conditional expectations. According to this theorem, the algorithm which is optimal for the robust and greedy criterion (11) is given by:

$$Opt(x_1, f(x_1,w), \ldots, x_n, f(x_n,w)) = x^{\sigma(w)}. \quad (13)$$

Proof. We consider a fixed map Opt and some fixed points $x_1, \ldots, x_n$ in D. For every $g \in G$, we write $x'_g(w) = Opt(x_1, y_1(w,g), \ldots, x_n, y_n(w,g))$.

First, it is clear that Equation (12) holds with equality if almost surely in w, for all $g \in G$, $x'_g(w) = x^{\sigma(w)}$. Therefore, it remains to prove that if there exists some $g_w \in G$ such that $x'_{g_w}(w) \neq x^{\sigma(w)}$ on a set of w of positive probability, then Equation (12) holds without equality.

For every ranking $\sigma \in \{-1,0,1\}^n$, we set $\Omega_\sigma = \{w \in W | \sigma(w) = \sigma\}$. These sets form a partition of W. We consider σ such that $P(w \in \Omega_\sigma) > 0$. Suppose that $\forall w \in \Omega_\sigma$, we have $\forall g \in G, x'_g(w) = x^{\sigma(\omega)}$. Then we clearly have

$$\mathbb{E}\left(\sup_{g \in G} \|x'_g(w) - x^*(w)\|^2 | w \in \Omega_\sigma\right) = \mathbb{E}\left(\|x^{\sigma(w)} - x^*(w)\|^2 | w \in \Omega_\sigma\right). \quad (14)$$

Otherwise, there exists at least one $w' \in \Omega_\sigma$ such that, for some $g' \in G$, one has $x'_{g'}(w') \neq x^{\sigma(w')}$. We write $x' = x'_{g'}(w')$ and $x^\sigma = x^{\sigma(w')}$.

For every $w \in \Omega_\sigma$, let $g_w \in G$ be a map such that $g_w(f(x_i,w)) = g'(f(x_i,w'))$ for all $i \in \{1,\ldots,n\}$. Such a map g_w exists because the two sets of points are equally ordered by definition of Ω_σ. Then,

$$\forall w \in \Omega_\sigma,\ x'_{g_w}(w) = x' \neq x^\sigma.$$

By definition, $\sup_g \|x'_g(w) - x^*(w)\|^2 \geq \|x'_{g_w}(w) - x^*(w)\|^2$. This implies that

$$\mathbb{E}\left(\sup_{g\in G} \|x'_g(w) - x^*(w)\|^2 | w \in \Omega_\sigma\right) \geq \mathbb{E}\left(\|x' - x^*(w)\|^2 | w \in \Omega_\sigma\right)$$
$$\geq \mathbb{E}\left(\|\underbrace{\mathbb{E}\left(x^*(w)|w \in \Omega_\sigma\right)}_{x^{\sigma(w)}} - x^*(w)\|^2 | w \in \Omega_\sigma\right).$$

(the last inequality comes from the fact that the expectation $\mathbb{E}(X)$ minimizes $t \mapsto \mathbb{E}(t - X)^2$). Moreover, this is not an equality because $x' \neq x^\sigma$. Therefore,

$$\mathbb{E}\left(\sup_{g\in G} \|x'_g(w) - x^*(w)\|^2 | w \in \Omega_\sigma\right) > \mathbb{E}\left(\|x^{\sigma(w)} - x^*(w)\|^2 | w \in \Omega_\sigma\right). \quad (15)$$

Now we write

$$\mathbb{E}\left(\sup_{g\in G} \|x'_g(w) - x^*(w)\|^2\right) = \sum_\sigma \mathbb{E}\left(\sup_{g\in G} \|x'_g(w) - x^*(w)\|^2 | \Omega_\sigma\right) P(\Omega_\sigma).$$

We use (14) for the rankings σ such that $\forall g \in G, x'_g = x^\sigma$ a.s. in Ω_σ and (15) for the other rankings σ, and we conclude that

$$\mathbb{E}\left(\sup_{g\in G} \|x'_g(w) - x^*(w)\|^2\right) \geq \mathbb{E}\left(\|x^{\sigma(w)} - x^*(w)\|^2\right), \quad (16)$$

which is (12). Moreover, if there exists σ with $P(w \in \Omega_\sigma) > 0$ such that, for some $w' \in \Omega_\sigma$ and some $g' \in G$, $x'_{g'}(w') \neq x^{\sigma(w')}$, then we have shown that (15) holds, which leads to a strict inequality in (16). This is the expected result. □

Theorem 2 deals with offsprings of one point. An equivalent of Theorem 2 for offsprings of λ-points can be derived:

Theorem 2'. *Assume that for all w, $f(.,w)$ is a real-valued function defined on $D \subset \mathbb{R}^d$ which has one and only one minimum at $x^*(w)$. Consider a map $Opt : (D \times \mathbb{R})^n \to D^\lambda$ and let $\sigma(w) = (sign(f(x_i,w) - f(x_j,w)))_{(i,j)\in\{1,\dots,n\}^2}$ be the ranking of the $f(x_i,w)$ for $i \in \{1,\dots,n\}$. Fix $x_1,\dots,x_n$ in D and write $y_i = g(f(x_i,w))$. Then for every ranking σ_0, one has*

$$\mathbb{E}\left(\sup_{g\in G} d(Opt(x_1,y_1(w,g),\dots,x_n,y_n(w,g)),x^*(w))^2 | \sigma(w) = \sigma_0\right)$$
$$\geq \inf_{x\in D^\lambda} \mathbb{E}\left(\inf_{z\in x} \|z - x^*(w)\|^2 | \sigma(w) = \sigma_0\right) \text{ (intra-class variance)}, \quad (17)$$

where the notation $z \in x$ means that z is equal to one of the coordinates of the λ-tuple $x \in D^\lambda$ (hence $z \in D$). Moreover, Equation (17) is an equality if and only if for almost all w such that $\sigma(w) = \sigma_0$, the λ-tuple $Opt(x_1, f(x_1,w)\dots,y_m, f(x_m,w))$ realizes the minimum of Equation (17).

Interpretation. This theorem proposes a choice of the offspring, computed by conditional expectations. This choice is optimal for the *greedy* criterion, i.e. if only the next

iteration is considered as a quality criterion. The theorem is stated for all σ_0; thus it can also be read as the fact that any greedy-optimal Opt must choose

$$\arg\min_{x\in D^\lambda} \mathbb{E}(\inf_{z\in x} \|z - x^*(w)\|^2 | \forall i,j \leq n, sign(f(x_i,w) - f(x_j,w)) = sign(y_i - y_j)).$$

An interesting point is that, whereas in the case $\lambda = 1$ this choice is fully deterministic, here, the $\arg\min$ is not necessarily unique and therefore the choice is not necessarily deterministic. However, it is likely that it is deterministic when there is no particular symmetry.

Proof. The proof is very similar to the proof of Theorem 2. For every $g \in G$, we write $u_g(w) = Opt(x_1, y_1(w,g) \ldots, x_n, y_n(w,g))$.

For every ranking σ, we set $\Omega_\sigma = \{w \in W | \sigma(w) = \sigma\}$. Suppose that for some σ with $P(w \in \Omega_\sigma) > 0$ there exist $w' \in \Omega_\sigma$ and $g' \in G$ such that $u = u_{g'}(w')$ does not minimize $u \mapsto \mathbb{E}\left(\inf_{z\in u} \|z - x^*(w)\|^2 | \Omega_\sigma\right)$. For all $w \in \Omega_\sigma$ there exists $g_w \in G$ such that $u_{g_w}(w) = u_g(w)$. Thus

$$\begin{aligned} \mathbb{E}\left(\sup_{g\in G} d(u_g, x^*(w))^2 | w \in \Omega_\sigma\right) &\geq E\left(d(u, x^*(w))^2 | \Omega_\sigma\right) \\ &> \inf_{x\in D^\lambda} \mathbb{E}\left(d\left(x, x^*(w)\right)^2 | \Omega_\sigma\right). \end{aligned}$$

We conclude the proof by doing a summation on the various possible rankings σ, as in Theorem 2. □

Theorem 2 shows how to reach optimality conditionally to $x_1, y_1, \ldots, x_n, y_n$, for the worst case among increasing transformations g. However, we are also interested in approximate optimality, i.e. in showing that any approximately optimal algorithm is approximately equal to the algorithm above. This is the object of the following corollary:

Corollary of Theorem 2: approximate optimality. *Let $\epsilon > 0$. Then every optimization algorithm $f \mapsto Opt_f = (x_n)_{n\in\mathbb{N}}$ associated to a map $Opt : \{\emptyset\} \cup \bigcup_{n\in\mathbb{N}} (D \times \mathbb{R})^n \to D$ such that*

$$\forall n \in \mathbb{N}; \mathbb{E}\left(\sup_{g\in G} \|x_n - x^*(w)\|^2\right) \leq Var(x^*(w)) + \epsilon \tag{18}$$

verifies $\mathbb{E}\left(\|Opt(x_1, f(x_1,w), \ldots, x_n, f(x_n,w)) - x^\sigma\|^2\right) \leq \epsilon$.

Remark. *This criterion of optimality is a greedy criterion. We do not say that optimizing a greedy criterion is a good idea; we will in fact precisely develop the reverse idea in the rest of the paper. An obvious drawback of this criterion is that, if the distribution is symmetric by rotations, e.g. $x^*(w)$ uniformly distributed on $\{x \in \mathbb{R}^d; ||x|| \leq 1\}$, and if we use the greedy-optimal algorithm to define the very first point x_1, and then x_2, x_3, ..., then it has the very inappropriate behavior that $\forall n,\ x_n = \mathbb{E}x^*(w) = 0$.*

At least two solutions are conceivable to avoid this trouble: randomly generate the first points, or add noise. The former preserves the optimality criterion in the theorem above; thus we choose this solution. A possible algorithm, satisfying Equation (18) and

for which no immediate counter-example has been found, is algorithm 1. However notice that the solution with noise is properly justified by the corollary above: with a small noise, the algorithm is "almost" optimal.

Algorithm 1 Algorithm BEDA.

Randomly generate $x_1, \ldots, x_{d+1}$ independently in the domain and compute $y_i = f(x_i)$ for $i = 1, \ldots, d+1$.
for $n \geq d+2$ **do**
  Define
  $x_n = \mathbb{E}\left(x^*(w) | \forall (i,j) \in \{1, \ldots, n-1\}^2, sign(f(x_i, w) - f(x_j, w)) = sign(y_i - y_j)\right)$
  (x_n is computed by billiard, see Section 3.3)
  Compute $y_n = f(x_n)$.
end for

We call this algorithm BEDA (Billiard-Estimation-of-Distribution-Algorithm) because in our implementation a billiard is used to compute x^σ. BEDA is well-defined for any distribution of fitness functions having one and only one global minimum almost surely provided that the variance of $x^*(w)$ is finite (but some distributions might be much harder than others for a real implementation). In order to ensure that no pathological behavior appears, the $d+1$ first points are randomly chosen in the domain $D \subset \mathbb{R}^d$.

We call BREDA, for Billiard-Random-Estimation-Of-Distribution-Algorithm, the version in which x_n is generated according to the distribution of $x^*(w)$, conditionally to σ (instead of choosing its expectation). BREDA is presented in algorithm 2.

Algorithm 2 Algorithm BREDA.

Randomly generate $x_1, \ldots, x_{d+1}$ independently in the domain and compute $y_i = f(x_i)$ for $i = 1, \ldots, d+1$.
for $n \geq d+2$ **do**
  Randomly choose x_n according to the conditional distribution $(x^*(w) | \forall (i,j) \in \{1, \ldots, n-1\}, sign(f(x_i, w) - f(x_j, w)) = sign(y_i - y_j))$
  (x_n is computed by billiard, see Section 3.3)
  Compute $y_n = f(x_n)$.
end for

The offspring in BREDA is therefore uniformly independently distributed according to the distribution of optima conditionally to the ranking of the previously visited points. BEDA is proved greedy-optimal and deterministic; BREDA is just a random generation of the offspring according to the conditional distribution used in BEDA. It is an idealized form of Estimation-Of-Distribution algorithm, as all its memory is the distribution of the optimum conditionally to all past information.

We can also consider a parallel version of our algorithm based on Theorem 2' in-

stead of Theorem 2, in which λ points (instead of one point) are generated at each epoch. This leads to algorithm 3.

Algorithm 3 Algorithm λ-BEDA.

Randomly generate $d+1$ points uniformly and independently in the domain and compute $y_i = f(x_i)$ for $i = 1, \ldots, d+1$.
for $n \geq d+2$ **do**
  Intra-class-variance-minimization: define $x_n^1, \ldots, x_n^\lambda$ a family of λ points minimizing $\mathbb{E}\left(\inf_i \|x^*(w) - x_n^i\|^2 | sign(f(x_i, w) - f(x_j, w)) = sign(y_i - y_j)\right)$ where σ is the ranking of all previously visited points.
  Compute $y_n^i = f(x_n^i)$.
end for

3.3 How to compute x^σ or the λ generated points in practice?

The expectation (w.r.t the random variable w, conditionally to the ranking of previously visited points) defining x^σ in BEDA, which has been shown to be an optimal estimate of the $\arg\min$ for the worst case among $g \in G$, can be computed by *ergodic billiard*. The same technique can be used to generate a random offspring as in BREDA (and offsprings of size greater that 1 as in λ-BEDA or λ-BREDA). Ergodic billiard can be used as in Bayesian inference (see e.g. (13)). Consider a uniform prior probability for w on some set E; E is defined by a set of constraints. Then the billiard algorithm is algorithm 4.

Algorithm 4 Billiard algorithm.

Find one point w^0 such that $\forall i, j \leq n, sign(f(x_i, w^0) - f(x_j, w^0)) = sign(y_i - y_j)$.
Choose randomly one direction d^0 in the unit sphere of $\mathbb{R}^d$.
for $t = 0, \ldots, T-1$ **do**
  $w^{t+1} = w^t + \eta_t d^t$ with $\eta_t > 0$ as small as possible such that at least one constraint among the followings becomes active:

  - constraints ensuring that $sign(f(x_i, w^{t+1}) - f(x_j, w^{t+1})) = sign(y_i - y_j)$;
  - constraints ensuring that $w^{t+1} \in E$.

  Get d^{t+1} by symmetrization of the direction d^t w.r.t active constraints.
end for
Let W_t be the random variable uniformly distributed on $[w_t, w_{t+1}]$.
if mode = compute expectation **then**
  Output $\frac{1}{\sum_{t=0}^{T-1} \eta_t} \sum_{t=0}^{T-1} \eta_t \mathbb{E} x^*(W_t)$
else
  With probability η_i, output an outcome of $x^*(W_i)$.
end if

If the billiard is ergodic, what is not proved but conjectured at least for constraints in general position, the sequence of segments $[w^t, w^{t+1}]$ weighted by the prior probability on w approximates the posterior probability conditionally to the constraints. We simply take the average value of $x^*(w)$ associated to this ergodic billiard as a mean-square approximation x^σ of the optimum. Moreover, randomly sampling on the trajectory (of the billiard) provides a random sampling as required by BREDA. We can also consider the case of λ points generated simultaneously (algorithm λ-BEDA, corresponding to Theorem 2').

Since the ergodic billiard provides points uniformly distributed in the domain, k-means can be used to find the x_t^i by minimization of the intra-class-variance. This is a derandomization of the offspring generation. The algorithm is as follows: (1) sample N points by billiard, with their weights; (2) apply k-means on these points.

So, we have shown the optimality of BEDA for a "greedy" criterion, in which the ultimate goal is always the next iteration. This might be very far from the optimality for, e.g. 100 iterations. This problem will be discussed in Section 4.2.

4 Experiments

We present experiments with the algorithms BEDA and BREDA defined in Section 3. Combining results from (23) and results above, we can claim that:

- solving translations of any fitness function in $[0,1]^d$ with comparisons only is possible only with a constant in the linear convergence rate at best $1 - O(1/d)$ with respect to the number of comparisons;
- for some simple fitness functions, this convergence rate is achieved by some simple algorithms which are independent of composition with increasing functions (e.g. (15)). For the worst case among such compositions, comparison-based methods are in fact optimal (Theorem 1 and Corollary);
- for the greedy criterion defined in Section 3, an optimal algorithm in the greedy case can be defined and implemented through billiards.

We now experimentally study the convergence rate for evolutionary algorithms, in the case of the sphere function and some other benchmarks, after composition by increasing functions.

Theorem 1 has shown how to build, for every fitness-based optimization-algorithm Opt and every fitness function f, a fitness $g \circ f$ which is the composition of this fitness by some $g \in G^0$, and for which this algorithm can not be better than some comparison-based algorithm. Section 4.1 shows that robustness with respect to the worst case among $g \in G^0$ is sufficiently well approximated by the construction of g as in the proof of Theorem 1 to strongly disturb standard non-comparison-based algorithms. Then in Section 4.2 we experiment the efficiency of our billiard-based algorithm compared with some other algorithms, and in particular we compare the

greedy-optimal BEDA (derandomized form of EDA, with a proof of greedy-optimality of the derandomization) and the randomized suboptimal counterpart BREDA, which is a very formalized yet quite natural version of EDA (in which each offspring is generated according to the exact conditional distribution of optima).

4.1 Results on the Cec'05 benchmarks after transformation by increasing functions

Below, we consider the optimization of a fitness $g \circ f$ with g defined as in the proof of Theorem 1, where f is one of the fitness functions in (22). Each optimizer works on $g \circ f$. The result reported below is the best value of f on points visited by the algorithm. Precisely, the experimental setup is as follows:

- Consider Opt an optimizer and f a fitness function.
- Apply Opt to $g \circ f$, where g is built as in Theorem 1 (g is built during the run of the algorithm).
- The result is $r = f(x)$, where x is the best visited point for f (or equivalently, the best visited point for $g \circ f$).

The expectation of r (which is randomized if f is randomized or if the algorithm is randomized), is therefore the expectation of the result for $g \circ f$, where g is built as in Theorem 1; this is a lower bound on the expectation of r associated to $g \circ f$ for the worst case among increasing transformations g. The results show that this lower bound is sufficiently tight to strongly modify the relative efficiency of algorithms.

LBFGSB is the Limited-memory Box-constrained BFGS from (28). Random is the naive random search. GAO is the simple genetic algorithm defined in (10). HJ is the Hooke&Jeeves algorithm ((15; 17; 27), the implementation is available at `http://www.ici.ro/camo/unconstr/hooke.htm`). CMAES is the covariance-matrix-adaptation algorithm from (12; 9) (Beagle version 3.0.1), with $\lambda = 2\lfloor (4. + \lfloor 3. * ln(dimension) \rfloor)/2 \rfloor$. LBFGSB uses finite differences and is the only algorithm that does not depend only on comparisons. All source codes can be found in the freely available sgLibrary, part of the OpenDP project (`http://opendp.sourceforge.net`).

Results for dimension 2 are presented in Table 1. Function f0 is $x \mapsto \|x - w\|^{1/4}$ with w uniformly distributed in the unit ball, functions f1 to f6 are the uni-modal functions in the Cec05 benchmarks ((22)). The number presented is the average fitness after 256 fitness-evaluations, averaged over 33 runs. As LBFGSB does not depend only on comparisons, we presents two columns of results; left, without transformation g; right, with the transformation g defined in the proof of Theorem 1, except that we use ± 1 instead of $\pm 1/n^2$ for the increment corresponding to the n-th point if it is above the maximum visited fitness or below the minimum visited fitness (the $1/n^2$ is needed in the proof of the corollary about asymptotic convergence rates, but any increments are suitable in Theorem 1 itself). The other algorithms behave in the same way with or without the transformation g, and so only one column is given. The comparisons for fitness

functions f1-f6 are moderately significant, as the deterministic algorithm LBFGSB cannot provide standard deviations for deterministic fitness functions f1-f6 and standard deviations for stochastic algorithms are moderately informative for deterministic fitness functions, but the conclusion that, when g is applied, LBFGSB is outperformed by GAO, HJ and CMAES for fitness functions f0 is significant, and the fact that this behavior is reproduced for every fitness among f1, f2, f3, f4, f5 is significant too. Therefore, in view of these experiments and in accordance with the theory above, we conclude that (i) the robustness w.r.t. g is not verified by LBFGSB even in practice, (ii) the worst-case among g makes even very easy functions intractable by non-comparison-based algorithms, (iii) the procedure defined in the proof of Theorem 1 is efficient to find very hard fitness functions.

	LBFGSB	Random	GAO	HJ	CMAES
f0	0.266 / 0.524 $\pm$ 0.075 /0.045	0.562 $\pm$ 0.042	0.366 $\pm$ 0.068	0.179 $\pm$ 0.055	0.367 $\pm$ 0.058
f1	-450 / 2361	-440.126	-449.941	-450	-450
f2	-450 / 8482	-407.200	-449.775	-450	-450
f3	-449.998 / 4852	50980	2131	6360	1962
f4	9080 / 11677	-391	-449.747	-313	-450
f5	-310 / 7788	32	-310.000	-310	-310
f6	416 / 822	5086	466	6234.1e3	464

Table 1: Results in dimension 2. We see that LBFGSB is the best algorithm in the standard case for fitness functions f0, f3 and f6, it is placed first equal for fitness functions f1, f2, f5, and it is outperformed by comparison-based algorithms only for fitness f4. When g is applied, LBFGSB is worse than random search for fitness functions f1, f2, f4, f5, and worse than GAO or CMAES for all fitness functions. We see that the non-differentiability of f0, which comes from the application of $x \mapsto x^{1/8}$ to a differentiable fitness, is not a big trouble for LBFGSB, whereas the composition by the function g built in Theorem 1 is much harder.

Tables 2 and 3 present the results with the same experimental setup but in dimension 10 and 50 respectively.

4.2 Results of BEDA and BREDA on the sphere function: how randomized algorithms outperform deterministic greedy-optimal ones

The trouble in the greedy algorithms is that bad configurations of the population can occur and make the algorithm stall. This fact has been emphasized for example in (6) as a major difference between Torczon's simplex search (24) and the one from (18), and the problem often arises in spite of the faster behavior at first view of methods that do not take into account the conditioning of points. We are going to see that the deterministic greedy-optimized algorithm suffers from this weakness.

	LBFGSB	Random	GAO	HJ	CMAES
f0	0.199 / 1.095 $\pm$ 0.020 /0.010	1.037 $\pm$ 0.002	0.709 $\pm$ 0.007	0.255 $\pm$ 0.003	0.816 $\pm$ 0.010
f1	-450.000 / 43660.536	14872.822	320.201	-449.326	-179.012
f2	-449.408 / 63326.024	15272.491	5159.797	1230.227	3573.682
f3	788.e3 / 15083.e3	103183.e3	39971.e3	5122.e3	25340.e3
f4	237.e3 / 248.e3	17.e3	6100	11.e3	3788
f5	6994 / 32185	14082	6321	783	2232
f6	2882 / 2817154	667211152	67681795	1040	1384025

Table 2: Dimension 10. We see that LBFGSB is the best algorithm in the standard case for fitness functions f0, f1, f2, f3, and it even outperforms CMAES for fitness f6. When the transformation g from Theorem 1 is applied, it is outperformed by CMAES and HJ in all cases and by random-search or GAO for fitness functions f0, f1, f2, f4, f5.

	LBFGSB	Random	GAO	HJ	CMAES
f0	1.095 / 1.357 $\pm$ 0.009 / 0.003	1.346 $\pm$ 0.000	1.214 $\pm$ 0.001	0.614 $\pm$ 0.003	1.300 $\pm$ 0.005
f1	-450 / 326824	195391	104912	5408	85405
f2	436.e3 /5996.e3	504.e3	268.e3	321.e3	303.e3
f3	95.e7/460.e7	491.e7	226.e7	44.e7	280.e7
f4	6144.e3/8981.e3	591.e3	283.e3	483.e3	386.e3
f5	57277/79790	48199	37554	32045	40567
f6	518.e7/977.e7	15758.e7	6214.e7	25.e7	4958.e7

Table 3: Dimension 50. We see that BFGS is the best algorithm for 256 function-evaluations for the easy function f1. For all other functions, even without transformation g, BFGS is outperformed by the Hooke&Jeeves algorithm, and also by all algorithms (even random search) for f4 and f5. This confirms the known fact that BFGS, which is known very efficient for very high-dimensional problems and has a fast asymptotic convergence rate, does not work efficiently when the number of fitness-evaluations is very moderate (since the gradient is not available, finite differences are applied, therefore each iteration costs 51 fitness-evaluations). When the function g built in Theorem 1 is applied, BFGS is the worst algorithm (even worse than random search) for f0, f1, f2, f4 and f5; it is only better than random search for f3, but worse than all other algorithms. BFGS remains reasonably efficient on f6 (however it is outperformed by HJ).

First, it is easy to construct counter-examples for which the fully derandomized greedy-optimal algorithm BEDA does not converge to the minimum of the fitness, even with a sphere function with minimum uniformly distributed in the unit ball, if this ap-

proach has been used since the first iteration (see the discussion before the definition of the BEDA-algorithm in Section 3). This counter-example, based on simple symmetries, can be implemented and experiments confirm this pathological behavior. However, if we just pick at random the first $d+1$ iterations and then apply the greedy algorithm for future iterations, we have no counter-example. Therefore, we will experiment this case for which there is no straightforward counter-example.

We compare three versions of our algorithm: (i) the BEDA algorithm (with billiard), (ii) another algorithm without billiard, simply using a point satisfying the constraints provided by the stochastic gradient algorithm, (iii) the BREDA algorithm using billiard in order to generate one point according to the distribution probability of $\arg\min f(.,w)$ conditionally to previously visited points and their ranking. Results are presented in Figures 2 and 3. HJ as above, and CMAES as above, and the 1+1-ES with isotropic Gaussian mutations and one-fifth-rule ($\sigma \leftarrow 2\sigma$ for each successful mutation and $\sigma \leftarrow \sigma/2^{1/4}$ in other cases) are also included in the comparison.

Results show that (i) BREDA is much more efficient than various existing algorithms; (ii) choosing always x^σ (the greedy-optimal solution) as next point is not a good solution. Randomly choosing a possible candidate, according to the posterior probability, is better. The somewhat surprising second point shows that the derandomized version is far worse than the random version, in spite of the optimality proof of the derandomized version for the greedy-criterion. We believe that this is a clue to the efficiency of random offsprings to avoid bad conditioning of offsprings, which is an important issue in direct search methods in general and in EA in particular.

5 Discussion

Evolutionary algorithms are often comparison-based; we summarize the advantages of this below. In the next section, we will discuss the advantages of randomized offspring in the anytime case.

Discussion on comparison-based algorithms

It has been shown in (23) that comparison-based algorithms are slow for large dimensions. This is consistent with practice, but does not reflect the empirically known fact that comparison-based algorithms have a strong robustness. In this paper we show (Theorem 1 and corollaries, Section 2) that comparison-based algorithms are however as fast as all fitness-based methods for the worst case among C^∞-increasing transformations of the fitness.

Discussion on randomized offsprings versus deterministic offsprings in an anytime case

We showed (Section 3.1) that an optimal algorithm can be designed for a distribution of fitness functions, using principles similar to standard tools of Bayesian learning, i.e. by taking into account a Bayesian prior. This is in particular in accordance with No-

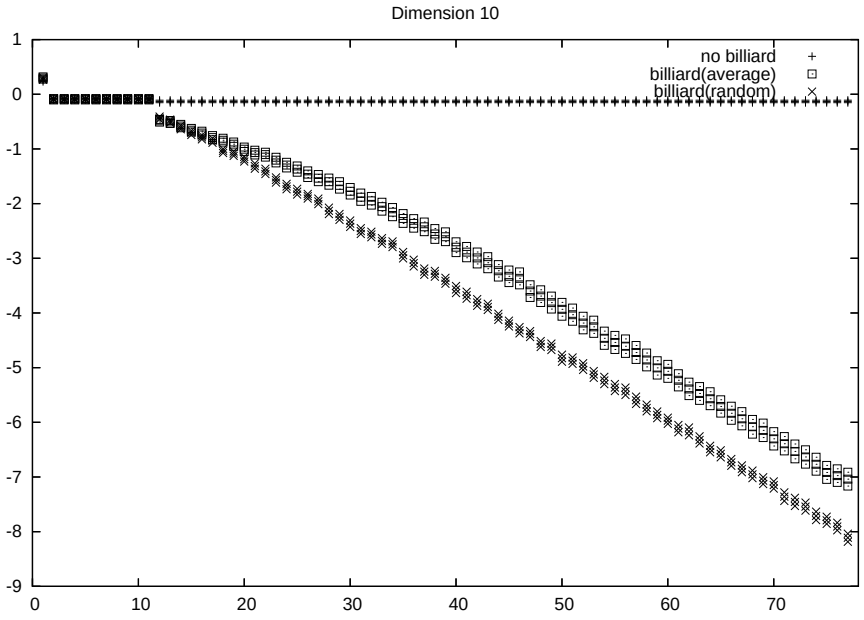

Figure 2: Results in dimension 10. We present the $\ln$ (natural logarithm) of the distance to optimum versus the number of function-evaluations, $\pm$ standard deviation. The case in which the first possible optimum (i.e. the first point found consistent with the ranking of previous points) is used as next iteration does not converge (it is only shown here for comparison). The random offspring in the domain is better than the deterministic solution, in spite of the fact that the latter is optimal in a greedy sense. With 256 function-evaluations, Hooke&Jeeves, isotropic 1+1-ES with one-fifth rule and CMAES parameterized as described in the text reach respectively -5.532 ± 0.78, -3.64028 ± 0.79816 and -1.692 ± 0.75 (-1.736 ± 0.61, -0.64516 ± 0.29034 and 0.064 ± 0.57 with 64 function-evaluations) with domain $[-1, 1]^{10}$. Therefore, our billiard-based algorithm, in the case of the sphere-function in dimension 10, is a comparison-based algorithm much better with 70 fitness-evaluations than Hooke&Jeeves, isotropic 1+1-ES with one-fifth rule and CMAES (parameterized as explained in the text) with 256 fitness-evaluations. Whereas BEDA is shown greedy-optimal, the random-sampling-based BREDA is more efficient than BEDA.

Free-Lunch theorems that show that priors are necessary to prove that an algorithm is better than another, and provides a link between a theory (NFL results, plus Bellman's optimality principle that has already been applied in other areas of mathematical programming), and an optimization algorithm. The parameters of this optimal algorithm detailed in Section 3.1 are all direct consequences of explicit prior knowledge:

- the a priori distribution of fitness-functions;
- the number of iterations;
- the criterion of quality of a run.

Unfortunately, the algorithm is quite intractable (unless perhaps for very small numbers of iterations and moderate dimensions) and depends on these three quantities.

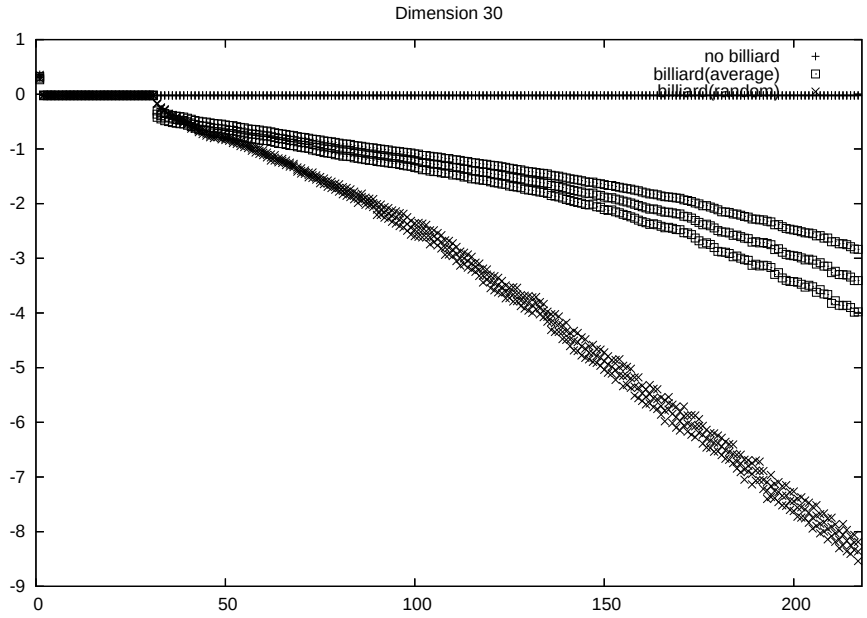

Figure 3: Results in dimension 30. With 256 function-evaluations, Hooke&Jeeves, isotropic 1+1-ES with one-fifth rule and CMAES reach respectively -1.944 ± 0.39166, -0.86954 ± 0.23360 and 0.556 ± 0.17. The billiard approaches BEDA and BREDA clearly outperforms other algorithms; in particular, the difference with existing algorithms is much larger than in lower dimension. Whereas BEDA is shown greedy-optimal, the random-sampling-based BREDA is much more efficient than BEDA, showing that the exploration-exploitation dilemma is not well solved by the greedy-optimal algorithm and that when dimension increases, random sampling is much better.

Interestingly, only the first of these three elements has been discussed often in the literature; e.g., choice of population size depending on the dimension or the multimodal nature of the fitness functions. This first parameter is necessary for optimality as shown by NFL-theorems; optimality for every distribution of problems is meaningless. The third one formalizes the goal of optimization but is not often discussed in the literature.

The second parameter, namely the number of iterations, is required to design this optimal algorithm. This algorithm is not anytime since an anytime algorithm, by definition, does not depend on the number of iterations. The number of iterations is formally required, but is this requirement only theoretical, or can we neglect this dependence? This would remove the exploration/exploitation dilemma. We showed in Section 3.2 that this dependence is in fact quite strong, in the sense that greedy-optimality (Equation (1), optimized by algorithm BEDA) is far from ensuring a good long-term (non-greedy) efficiency. We compare two algorithms which do not depend on the number of iterations:

- BEDA (algorithm 1), which is greedy-optimal, and

- BREDA (algorithm 2), which randomly chooses the offspring according to the current conditional distribution of the optimum.

In both cases, the algorithm does not depend on the number of iterations and has no free hyper-parameter. The two algorithms are tested in the case in which the full conditional distribution of the optimum is computed from (i) the a priori distribution of fitness functions, (ii) the ranking of previously visited points. In this case, this can be implemented in practice thanks to billiard-methods for the estimation of conditional distributions (Section 3.3). The deterministic version, BEDA, is proved to be greedy-optimal (Theorems 2 and 2'). However, the surprising experimental result is that the randomized counterpart, BREDA, is clearly much faster.

The two conclusions are that (i) the greedy-approximation is not a good solution to the anytime problem of optimization[4], (ii) random offsprings provide a simple and efficient way to deal with this anytime trouble.

6 Conclusions

This work analyzes randomized comparison-based optimizers.

The first part shows the robustness of comparison-based algorithms, both for finite-horizon behavior and convergence rates (Theorem 1 and corollary), from the point of view of optimality in a set of functions stable by composition with increasing functions. The main limitation of the first part is that we consider robustness in the sense of compositions with increasing functions. Robustness is only in this sense.

The second part studies optimal algorithms, interprets their exploration-exploitation dilemma by their dependence in the number of iterations, and the experiments show that the greedy-solution to remove this dependence (namely BEDA, that is proved to be greedy-optimal) is far less efficient than its randomized counterpart (namely BREDA; see Figures 2 and 3). The conclusion is that randomized algorithms are efficient from the anytime point of view. The main limitation of the second part is that we have only compared BEDA and BREDA for $\lambda = 1$ and for easy problems.

This fulfills the main goals of this work, namely (i) showing the robustness of comparison-based algorithms, (ii) showing the good anytime behavior of randomized algorithms. Moreover, we point out three interesting by-products of the work.

Firstly, NFL theorems have a positive counterpart, namely the existence of optimal optimization algorithms for a fixed distribution of fitness functions (Section 3.1, to be developed in a further work). These algorithms use Bellman's decomposition for non-linear programming.

Secondly, we point out the possible use of billiard methods to generate offsprings in continuous domains based on an estimated set of possible optima. A limitation is that we have only tested cases in which billiards could be applied in a straightforward

[4]This anytime problem is emphasized by the dependence of the optimal algorithm on the number of iterations (Section 3.1) and by the fact that greedy-optimal algorithms have poor performance outside the greedy criterion.

manner; generalizations to cases in which the conditional distribution of the fitness-function has a non-constant density in its support are possible by the use of Markov-Chain-Monte-Carlo methods or by non-Euclidean metrics in billiards as in (13). The main drawback is that these methods are computationally expensive, but results are impressive for very small numbers of iterations, and this results are therefore promising for expensive optimization problems in which a few minutes of billiard before each offspring is not a trouble.

Thirdly, we derive the algorithm BREDA, an idealized form of EDA, which is very efficient at least for the simple fitness functions under consideration. BREDA has been only tested on easy functions, for small numbers of iterations and moderate dimensions. This is due to the complexity of the algorithm and to the important computational cost. However, (i) algorithmic improvements are probably possible, (ii) industry provides important problems with only a few iterations and in moderate dimensions, (iii) approximations of BREDA are probably possible.

Acknowledgements

We thank Anne Auger, Pascal Hubert and Marc Schoenauer for fruitful talks. This work was supported in part by the Pascal Network of Excellence.

References

[1] Auger, A. (2005). Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible markov chains. *Theoretical Computer Science*, 334:35–69.

[2] Auger, A., Jebalia, M., and Teytaud, O. (2005). Xse: quasi-random mutations for evolution strategies. In *Proceedings of Evolutionary Algorithms, 12 pages*.

[3] Bellman, R. (1957). *Dynamic Programming*. Princeton Univ. Press.

[4] Bertsekas, D. (1995). *Dynamic Programming and Optimal Control, vols I and II*. Athena Scientific.

[5] Christensen, S. and Oppacher, F. (2001). What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function. In *In Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1219–1226.

[6] Conn, A., Scheinberg, K., and Toint, L. (1997). Recent progress in unconstrained nonlinear optimization without derivatives.

[7] Droste, S. (2005). Not all linear functions are equally difficult for the compact genetic algorithm. In *Proc. of the Genetic and Evolutionary Computation COnference (GECCO 2005)*, pages 679–686.

[8] Eiben, A. and Smith, J. (2003). *Introduction to Evolutionary Computing*. springer.

[9] Gagné, C. and Parizeau, M. (2002). Open BEAGLE: A new versatile C++ framework for evolutionary computations. In *Late breaking papers of the GECCO 2002*, pages 161–168.

[10] Gelly, S., Teytaud, O., and Gagné, C. (2006). Resource-aware parameterizations of EDA. In *Proc. of the 2006 IEEE Congress on Evolutionary Computation (IEEE-CEC 2006)*. http://www.lri.fr/∼teytaud/tsm2.pdf.

[11] Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison Wesley.

[12] Hansen, N. and Ostermeier, A. (2003). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 11(1).

[13] Herbrich, R., Graepel, T., and Campbell, C. (2001). Bayes point machines. *Journal of Machine Learning Research*, 1:245–279.

[14] Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor.

[15] Hooke, R. and Jeeves, T. A. (1961). Direct search solution of numerical and statistical problems. *Journal of the ACM, Vol. 8, pp. 212-229*.

[16] Jagerskupper, J. and Witt, C. (2005). Runtime analysis of a (mu+1)es for the sphere function. Technical report.

[17] Kaupe, A. F. (1963). Algorithm 178: direct search. *Commun. ACM*, 6(6):313–314.

[18] Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal 7*, pages 308–311.

[19] Nikulin, Y. (2004). Robustness in combinatorial optimization and scheduling theory: An annotated bibliography. *Optimization online*. http://www.optimization-online.org/DB_HTML/2004/11/995.html.

[20] Rudolph, G. (1997). Convergence rates of evolutionary algorithms for a class of convex objective functions. *Control and Cybernetics*, 26(3):375–390.

[21] Semet, Y. and Schoenauer, M. (2006). On the benefits of inoculation, an example in train scheduling. In *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 1761–1768, New York, NY, USA. ACM Press.

[22] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., and Tiwari, S. (2005). Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report AND KanGAL Report #2005005, IIT Kanpur, India.

[23] Teytaud, O. and Gelly, S. (2006). General lower bounds for evolutionary algorithms. In 10^{th} *International Conference on Parallel Problem Solving from Nature (PPSN 2006)*.

[24] Torczon, V. (1991). On the convergence of the multidirectional search algorithm. *SIAM journal on Optimization*, 1(1):123–145.

[Villemonteix et al.] Villemonteix, J., Vazquez, E., and Walter, E. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization (submitted)*.

[25] Whitley, L. D., Bush, K., and Rowe, J. E. (2004). Subthreshold-seeking behavior and robust local search. In *Proceedings of GECCO (2)*, pages 282–293.

[26] Wolpert, D. and Macready, W. (1995). No free lunch theorems for search. Technical report, Santa Fe Institute.

[27] Wright, M. (1995). Direct search methods: Once scorned, now respectable. *Numerical Analysis (D. F. Griffiths and G. A. Watson, eds.), Pitman Research Notes in Mathematics*, pages 191–208. http://citeseer.ist.psu.edu/wright95direct.html.

[28] Zhu, C., Byrd, R., P.Lu, and Nocedal, J. (1994). L-BFGS-B: a limited memory FORTRAN code for solving bound constrained optimization problems. *Technical Report, EECS Department, Northwestern University*.