Comparison-based algorithms: worst-case optimality, optimality w.r.t a bayesian prior, the intraclass-variance minimization in EDA, and implementations with billiards

Sylvain Gelly, Sylvie Ruette, Olivier Teytaud

To cite this version:

Sylvain Gelly, Sylvie Ruette, Olivier Teytaud. Comparison-based algorithms: worst-case optimality, optimality w.r.t a bayesian prior, the intraclass-variance minimization in EDA, and implementations with billiards. Parallel Problem Solving from Nature BTP-Workshop, Sep 2006, Reykjavik. inria-00112813

HAL Id: inria-00112813
https://hal.inria.fr/inria-00112813
Submitted on 9 Nov 2006

Comparison-based algorithms: worst-case optimality, optimality w.r.t a bayesian prior, the intraclass-variance minimization in EDA, and implementations with billiards

Sylvain Gelly**, Sylvie Ruette*, Olivier Teytaud**

*Laboratoire de Mathématiques, UMR 8628, Bâtiment 425, Université Paris-Sud, 91405 Orsay cedex, France
**Equipe TAO (Inria), LRI, UMR 8623 (CNRS - Université Paris-Sud), bât 490 Université Paris-Sud 91405 Orsay Cedex France
olivier.teytaud@inria.fr

Abstract. This paper is centered on the analysis of comparison-based algorithms. It has been shown recently that these algorithms are at most linearly convergent with a constant $1 - O(1/d)$; we here show that these algorithms are however optimal for robust optimization w.r.t increasing transformations of the fitness. We then turn our attention to the design of optimal comparison-based algorithms. No-Free-Lunch theorems have shown that introducing priors is necessary in order to design algorithms better than others; therefore, we include a bayesian prior in the spirit of learning theory. We show that these algorithms have a nice interpretation in terms of Estimation-Of-Distribution algorithms, and provide tools for the optimal design of generations of λ-points by the way of billiard algorithms.

1 Introduction

Many evolutionary optimization tools are based on two ideas: (i) to use random (ii) to use only comparisons between fitness values, and not the fitness values themselves. The second point is not systematic in evolutionary computation, but holds more and more often: Holland ([9]) uses fitness-proportional selection, and the simple-GA popularized in [6] uses more than comparisons, but suffers from various drawbacks, among which super-individuals (leading to premature convergence), and the puzzling non-invariance when adding constants to the fitness. This paper deals with the advantages of comparison-based methods, whereas [17] points out the drawbacks of comparison-based methods.

Many algorithms, in spite of this restriction (they only use comparisons, and loose any other information), have been proved linear; see e.g. [3, 1, 2, 15]. Some linear lower bounds also exist in various cases ([11, 17]). We introduce below the state of the art with respect to the analysis of comparison-based methods. We then prove the optimality of comparison-based methods at the worst case among any space of fitnesses invariant by composition with an increasing function (section 2).

In [17], it is shown that:
- this kind of algorithms can at best be linear *w.r.t the number of comparisons*, with a constant $1-O(\frac{1}{d})$ as the dimension d increases to ∞, even with very easy fitness functions;
- however, such algorithms can have slightly better constants *w.r.t the number of function evaluations* (theorem 4 in [17]), at least for some fitness-functions that have been specially designed but that are reasonable and in particular that are quasi-convex; an interesting point is that this requires features that are not present in usual (μ, λ)-algorithms;
- in some very particular cases, these non-standard algorithms can be superlinear *w.r.t the number of function-evaluations* if they use ranking-informations and not only selection (theorem 5 in [17]).

This suggests that the algorithm might be better with respect to fitness-evaluations than with respect to comparisons. This is in particular interesting for very-expensive fitness-functions, for which the main goal is to reduce the number of fitness-evaluations. We prove in this paper that ranking-based algorithms are, in spite of the limitation of the $1 - O(1/d)$ for the number of comparisons, optimal for the number of fitness-evaluations with respect to the worst case among increasing transformations of the fitness-functions. 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-methods. E.g. $x \mapsto \sqrt{||x||}$ is much harder than the sphere function for the Newton-algorithm; also, the Newton-algorithm is much worse than random-search on this function.

We show in section 2.1 that comparison-based methods are indeed optimal for the worst case among increasing transformations, in section 2.2 that some optimality criterion and corresponding algorithms can be derived for this worst-case framework. We experiment the algorithms in section 3 and conclude that (i) the introduction of bayesian prior can lead to significant breakthroughs in optimization algorithms as in learning (ii) these priors can be combined with robustness w.r.t increasing transformations of the objective function (iii) that the practical implementation is possible thanks to ergodic billiards.

2 Some new robustness results for comparison-based methods

We study in 2.1 the advantages of comparison-based algorithms and in 2.2 we propose a comparison-based estimator of optimal parameter with some optimality properties. In 2.3 we detail a possible implementation by ergodic billiard.

Notations: We note G the set of increasing functions from $\mathbb{R}$ to $\mathbb{R}$. We note $sign(x) = 1$ if $x > 0$, $sign(x) = -1$ if $x < 0$ and $sign(0) = 0$. Note G^0 the set of increasing continuous functions from $\mathbb{R}$ to $\mathbb{R}$. Note $d(A, b) = \inf_{a \in A} ||a - b||$.

2.1 Optimality of comparison-based methods for the worst case among increasing transformations

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-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 framework ([14]).

Theorem 1. *Consider F a space of functions, each of them having one and only one global minimum in a given domain D, and assume that F is stable by composition with any increasing function. Consider a deterministic optimization algorithm $Opt(x_1, y_1, \ldots, x_k, y_k)$ that computes a pair (x_{k+1}, x'_{k+1}) (and define $Opt() = (x_1, x'_1)$). This leads to a sequence of iterates through $y_i = f(x_i)$ and $(x_{i+1}, x'_{i+1}) = Opt(x_1, y_1, \ldots, x_i, y_i)$.*

Assume that for some N and any $f \in F$,

$$||x'_N - \arg\min f|| \leq \epsilon \tag{1}$$

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

$$(\forall i, j,\ sign(y_i - y_j) = sign(y'_i - y'_j))$$
$$\implies Opt'(x_1, y_1, \ldots, x_n, y_n) = Opt'(x_1, y'_1, \ldots, x_n, y'_n)$$

and Opt' is such that equation 1 holds also if $(x_{i+1}, x'_{i+1}) = Opt'(x_1, y_1, \ldots, x_i, y_i)$ and $y_i = f(x_i)$.

Proof:

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*

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

We see that y_i only depends on the z_j for $j \leq i$. More precisely, y_i only depends on the $sign(z_j - z_k)$ for $j, k \leq i$. This means that Opt' only depends on comparisons. We now only have to prove that Opt' verifies equation 1.

Consider $fitness$ a fitness function in F. We consider the $x_1, \ldots, x_N$, $x'_1, \ldots, x'_N$, $z_1, \ldots, z_N$ defined by $(x_1, x'_1) = Opt'()$, $z_n = fitness(x_n)$, $(x_n, x'_n) =$

$Opt'(x_1, z_1, \ldots, x_{n-1}, z_{n-1})$. We consider $y_1, \ldots, y_N$ defined as above. By construction, for any i and j, $sign(y_i - y_j) = sign(z_i - z_j)$. Since the two sets of points are finite and equally ordered, there exists $g \in G$, such that $\forall i \in [[1, N]]$, $g(z_i) = y_i$. We note $fitness' = g \circ fitness$.

We now consider the $\tilde{x}_1, \ldots, \tilde{x}_N$, $\tilde{x}'_1, \ldots, \tilde{x}'_N$, $\tilde{y}_1, \ldots, \tilde{y}_N$ defined by $(\tilde{x}_1, \tilde{x}'_1) = Opt()$, $\tilde{y}_n = fitness'(\tilde{x}_n)$, $(\tilde{x}_n, \tilde{x}'_n) = Opt(\tilde{x}_1, \tilde{y}_1, \ldots, \tilde{x}_{n-1}, \tilde{y}_{n-1})$.

By construction, $\tilde{x}_i = x_i$, $\tilde{x}'_i = x'_i$ and $\tilde{y}_i = y_i$ for any $i \in [[1, N]]$. This shows that Opt' has the same behavior on $fitness$ as Opt on $fitness'$. This provides the expected result.$\square$

Corollary. *It is sufficient that F is stable by increasing C^∞-transformations.*

Proof: *In the proof of theorem 1, g can be chosen C^∞.* $\square$

We can consider also the asymptotic convergence rate.

Corollary: asymptotic convergence rate. *Assume that each $fitness$ in F has one only minimum in D. If for some algorithm Opt:*

a) for any $fitness$ in F and for any N, $||x'_N - \arg\min f|| \leq \epsilon_N$;
b) for any $fitness$ in F, $fitness(x_n) \to \inf fitness$;
c) for any $fitness$ in F and any $g \in G^0$, $g \circ fitness \in F$.

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

Proof: *We consider Opt' as in theorem 1 above. We show that for some $g \in G^0$,*

$$\forall i \in \mathbb{N}, g(fitness(x_i)) = y_i \tag{2}$$

(whereas in theorem 1, we only need such a property for $i \in [[1, N]]$.

We define $z_n = fitness(x_n)$.

Let's note z_∞ the limit of the z_n, i.e. $\inf fitness$ by assumption b). Let's note $y_\infty = \inf y_n$. We know that y_n is lower bounded by construction ($y_n \geq \inf_{i<n} y_i - 1/n^2$, $y_1 = 0$, therefore $y_n \geq -\sum_{i \geq 1} 1/i^2 > -\infty$). We define g_n the piecewise linear interpolation of the $\{(z_M, y_M), (z_1, y_1), \ldots, (z_n, y_n), (z_\infty, y_\infty)\}$, where $z_M = \max_i z_i$, $y_M = \max_i y_i$.

We first show that g_n converges to some limit g. For all $\epsilon > 0$, $\exists n \geq 1/\epsilon; \forall k \geq n, z_k \leq z_\infty + \epsilon$. Define

$$n_\epsilon \in \arg\max_{k \geq n} z_k$$

Then for any $k \geq n_\epsilon$, $g_{n_\epsilon} = g_k$ on $[z_{n_\epsilon}, z_M] \supset [z_\infty + \epsilon, z_M]$. Therefore g_n, defined on $[z_\infty, z_M]$, is continuous and converges pointwise to some g, and

$$\forall k \geq n_\epsilon, \forall z \in [z_{n_\epsilon}, z_M], g_k(z) = g(z) \tag{3}$$

$$\textit{where } n_\epsilon \to \infty \textit{ as } \epsilon \to 0 \tag{4}$$

$$\textit{and } z_{n_\epsilon} \leq z_\infty + \epsilon \tag{5}$$

Equation 3 implies that g is increasing on $[z_\infty, z_M]$. We are going to show that g_n uniformly converges to g. We need to prove the following fact:

$$y_n \to y_\infty \textit{ as } n \to \infty \tag{6}$$

Let's prove equation 6. If $\exists n_0, \forall n \geq n_0, z_n = z_\infty$, the result is immediate. Therefore, we assume without loss of generality that

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

Before showing equation 6, we will show equation 8:

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

If $\forall n, z_n \neq z_\infty$, then by definition, equation 8 holds. So, we will assume in the sequel of the proof of equation 8 that $\exists n_0; z_{n_0} = z_\infty$.

Then, thanks to equation 7, we can define by induction an increasing sequence $(n_i)_{i\geq 0}$ (n_0 is defined above) such that n_i is minimal under constraint

$$\forall i \geq 2, z_\infty < z_{n_i} < z_{n_{i-1}} \tag{9}$$

Then by definition, $y_{n_i} = \frac{y_{n_{i-1}}+y_\infty}{2}$. Then, equation 8 holds with $m = n_i$ for i sufficiently large.

We now have to show that equation 8 implies equation 6.

As $z_m > z_\infty$ and $z_n \to z_\infty$, $\exists p; \forall n \geq p, z_n \leq z_m$. So, $\forall n \geq p, y_n \leq y_m \leq y_\infty + \epsilon$. Therefore, equation 6 is proved.

Let's now show that $g_n \to g$ uniformly.

Since $n_\epsilon \to \infty$ as $\epsilon \to 0$ (equation 3), and according to equation 6,

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

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

$$\forall z \geq z_{n_\epsilon}, g_n(z) = g(z) \textit{ by equation } 3$$
$$\textit{and } \forall z \leq z_{n_\epsilon}, g_n(z) \leq g_n(z_{n_\epsilon}) = y_{n_\epsilon} \leq y_\infty + \epsilon'$$

This concludes the proof of $g_n \to g$ uniformly on $[z_\infty, z_M]$.

This implies that g is continuous as each g_i is continuous. Also we have shown that g is increasing. Therefore, g is in G^0. By assumption (c), $g \circ fitness$ is in F too. Therefore, assumption (a) shows that Opt reaches the convergence rate given by the ϵ_N on $g \circ fitness$, and Opt' therefore has the same property on $fitness$.$\square$

Remark: *In corollary above, assumption b) can be replaced by $\lim fitness(x_n) = \inf fitness(x_n)$, but the proof does not work if this assumption is removed. For example, suppose that $z_1 = fitness(x_1) = 0$ and $z_n = fitness(x_n) = 1 + 1/n$ for all $n \geq 2$. Then $z_n \to 1$, $y_1 = 0$ and $y_n = 1/2^n$ for all $n \geq 2$. Any non decreasing function g such that $g(z_n) = y_n$ is equal to 0 on $[0, 1]$, and so is not increasing.*

2.2 Bayesian optimization and one-step optimal algorithms for the worst case among increasing transformations of the fitness functions

It has been emphasize in NFL-theorems ([18]) that unless a priori information is available, there's no good or bad algorithms. We here investigate the effect of an a priori distribution of fitness-functions: can in that case optimal algorithms be designed ? We will see in the section below that the answer is yes. Then, in the spirit of section 2, we will focus on robustness with respect to increasing transformations of the fitness-functions, in a simplified (greedy) framework. We will see that this leads to reasonnably tractable algorithms that can be implemented with billiards (2.3).

An optimal algorithm for a given distribution of fitness functions Consider a family of fitness-functions $f(.,\theta)$ on a same domain D, depending on a random parameter θ and that each $f(.,\theta)$ has one and only one minimum $x^*(\theta)$. Define x_1,..., x_N the N iterates of an optimization algorithm:

- $x_1 = Opt()$;
- for $n \in \{2,\ldots,N\}$,
  $x_n = Opt(x_1, x_2, x_3, \ldots, x_{n-1}, f(x_1,\theta), f(x_2,\theta), \ldots, f(x_{n-1},\theta))$.

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*[**?**], which states that the following Opt is optimal:

Note $V(x_1,\ldots,x_i,y_1,\ldots,y_i) = \inf_{Opt_{|i}} E_{\theta;\forall j\leq i, y_j=f(x_j,\theta)}||x_N - x^*(\theta)||^2$. Bellman's optimality principle states that the following function Opt is optimal, i.e. minimizes $E||x_N - x^*(\theta)||^2$ [1]:

$$Opt(x_1,\ldots,x_{n-1},y_1,\ldots,y_{n-1}) \in \\ \arg\min_x E_{\theta;\forall i\leq n-1, y_i=f(x_i,\theta)} V(x_1,\ldots,x_{n-1},x,y_1,\ldots,y_{n-1},f(x,w)) \quad (10)$$

where V is computed by backward induction as follows:

$$V(x_1,\ldots,x_N,y_1,\ldots,y_N) = E_{\theta;\forall i, f(x_i,\theta)=y_i}||x_N - x^*(\theta)||^2 \\ V(x_1,\ldots,x_{n-1},y_1,\ldots,y_{n-1}) = \inf_x E_y V(x_1,\ldots,x_{n-1},x,y_1,\ldots,y_{n-1},y) \quad (11)$$

where y is distributed as $f(x,\theta)$, with θ following its probability distribution conditionally to $\forall i \leq n-1, f(x_i,\theta) = y_i$.

A faster version in a robust and greedy framework As the algorithm above is very complicated and beyond the scope of this paper, it will be the object of a further work. We will here introduce (i) a greedy-version of optimality (ii) a robust framework as in section 2. This leads to a framework classicaly termed design of experiments. In static design of experiments, n points are sampled on the domain, and the corresponding fitness-values are computed. Thereafter, these points and their fitnesses are used to estimate the model, and possibly the $\arg\min$ of the fitness. So, we here study this approach in the case (i) of an a

[1] We here study the mean square distance to the optimum after N iterates; other criterions could be considered as well.

priori distribution of probability on fitnesses in the spirit of bayesian statistics (ii) in a robust framework w.r.t compositions of fitnesses with $g \in G$.

Consider $x_1, \dots, x_n$ n points in $\mathbb{R}^d$. Consider a probability distribution $P(w)$. For some $g \in G$, consider $y = g(f(x,w))$ and $\forall i, y_i = g(f(x_i, w))$ (implicitly, y and y_i depend on g).

We consider below the optimal estimator of $\arg\min g \circ f(., w)$ for the squared distance for the worst case of $g \in G$.

Theorem 2. *Assume that any w is such that $f(., w)$ has one and only one minimum value $x^*(w)$. Then, for any function $Opt(x_1, \dots, x_n, y_1, \dots, y_n)$ with values in $\mathbb{R}^d$,*

$$E_w \sup_{g \in G} ||Opt(x_1, \dots, y_n) - x^*(w)||^2 \geq E_w \sup_{g\ \in G} ||x^\sigma - x^*(w)||^2 \qquad (12)$$

where $x^\sigma = E_w x^*(w)|sign(f(x_i, w) - f(x_j, w))$ (i.e., x^σ is the expectation of $x^*(w)$ conditionally to the values of the $sign(y_i - y_j)$ for any $i, j \in \{1, \dots, n\}$).

Moreover, equality in equation 12 only holds if $Opt(x_1, \dots, y_n) = x^\sigma$.

Proof:

Consider some function Opt. Consider Ω the set of w leading to some (arbitrary) fixed ranking of $x_1, \dots, x_n$.

Assume without loss of generality that there is at least one w in Ω such that for some g, $Opt(x_1, \dots, y_n) \neq x^\sigma$ (otherwise, if for any Ω, this does not occur, then Opt chooses x^σ in all cases and we are in the equality case).

Note x'_g the value of $Opt(x_1, \dots, y_n)$ (which depends on g as the y_i depend on g). x'_g is a random variable as it depends on w.

The main argument now is that, as we only consider $w \in \Omega$, i.e. w leading to one ranking of the $f(x_i, w)$, one can define g_w so that the y_i are fixed, independently of $w \in \Omega$, to some value such that $Opt(x_1, \dots, y_n) = x' \neq x^\sigma$. Then, x'_{g_w} is constant and equal to x' (we have no idea of its value, we only require that it is constant and different from x^σ).

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

$$\begin{aligned} E_{w\in\Omega} \sup_g ||x'_g - x^*(w)||^2 &\geq E_{w\in\Omega} ||x' - x^*(w)||^2 \\ \geq E_w || \underbrace{E_w x^*(w)}_{x^\sigma} - x^*(w)||^2 &+ ||x' - \underbrace{E_w x^*(w)}_{x^\sigma}||^2 \\ &\geq E_{w\in\Omega} || \underbrace{E_{w\in\Omega} x^*(w)}_{x^\sigma} - x^*(w)||^2 \end{aligned} \qquad (13)$$

with equality in eq. 13 if and only if $x' = x^\sigma$. This is the expected result. $\square$

We can also derive an equivalent of theorem 2 for generations of λ-points:

Theorem 2'. *Assume that for any w, $f(., w)$ has one and only one minimum $x^*(w)$ in domain D. Consider Opt a map with values in D^λ and note σ the ranking of the $f(x_i, w)$ for $i \in [[1, m]]$. Then, for any $x_1, \dots, x_m$, and for $y_i = g \circ f(x_i, w)$, for any σ_0,*

$$E_w \sup_g d(Opt(x_1, y_1, \dots, x_m, y_m), x^*(w))^2 | \sigma = \sigma_0$$

$$\geq \inf_{x\in D^\lambda} E_w d\left(x, x^*(w)\right)^2 |\sigma = \sigma_0 \text{ (intra-class variance)} \tag{14}$$

with equality if and only if for any $y_1, \ldots, y_m$ realizing σ_0 the family $Opt(x_1, \ldots, y_m)$ realizes the minimum of equation 14.

TODO manips 2'

Proof: *The proof is very similar to the proof of theorem 2. Note $u_g = Opt(x_1, \ldots, y_m)$.*

If for some w such that $\sigma = \sigma_0$, and for some $g \in G$, u_g is a family u that does not minimize equation 14, then for any w such that $\sigma = \sigma_0$ we can define g_w such that $u_{g_w} = u$. Then, with $\Omega = \{w; \sigma = \sigma_0\}$,

$$E_{w\in\Omega} \sup_g d(u_g, x^*(w))^2 \geq E_{w\in\Omega} d(u, x^*(w))^2 > \inf_{x\in D^\lambda} E_{w\in\Omega} d\left(x, x^*(w)\right)^2.$$

This concludes the proof. □

Theorem 2 shows how to reach optimality conditionally to $x_1, \ldots, x_n, y_1, \ldots, y_n$, for the worst case among increasing transformations g. Precisely, we can derive the following corollary:

Corollary of theorem 2. *Any algorithm of the form*

$$x_n = Opt(x_1, y_1, \ldots, x_{n-1}, y_{n-1})$$

$$y_n = fitness(x_n)$$

such that

$$\forall n \in \mathbb{N}; E \sup_g ||x_n - x^*(w)||^2 |s \leq Var(x^*(w)|s + \epsilon \tag{15}$$

where s is $x_1, \ldots, x_{n-1}$ and the ranking of $y_1, \ldots, y_{n-1}$, verifies $E||x_n - x^\sigma||^2 \leq \epsilon$.

This criterion of optimality is a greedy criterion. We only use this greedy criterion as more relevant criterions lead to difficultly tractable algorithms. Another drawback of this criterion is that the optimal algorithm, which reaches $\epsilon = 0$, has the following very inappropriate behavior due to the fact that at the first epoch, the distribution conditionally to σ is not modified by fitness results: $\forall n, x_n = Ex^*(w)$.

We believe that two solutions are available to avoid this trouble: randomly generate the m first points, or add noise. The latter preserves the optimality criterion in the corollary above; therefore we choose this solution. Therefore, a possible algorithm, satisfying equation 15, and depending on some arbitrary ϵ, is as follows:

Algorithm A(ϵ).

- consider ϵ a noise level.
- for $n \geq 1$,
  - define $x_n = x^\sigma + \epsilon'$ where σ is the ordering of the $(y_i = f(x_i, w); i < n)$ and ϵ' is a random gaussian isotropic noise of variance ϵ.
  - compute $y_n = fitness(x_n)$.

We note this algorithm BEDA (Billiard-Estimation-of-Distribution-Algorithm) when a billiard is used for computing x^σ. BEDA is well-defined for any distribution of fitnesses (but some distributions might be much harder than others for a real implementation). In that case, we can omit the noise with variance ϵ: the finiteness of the billiard trajectory is sufficient to introduce some noise. In order to ensure that no pathological behavior appears, the $d+1$ first points are randomly drawn on the unit sphere. We note BREDA, for Billiard-Random-Estimation-Of-Distribution-Algorithms, the variant in which x_n is generated according to the distribution of $x^*(w)$, conditionally to σ (instead of choosing its expectation).

We can also consider a parallel version of our algorithm based on theorem 2' instead of theorem 2, in which instead of 1 point, λ points are generated at each epoch. This leads to the following algorithm:

Algorithm B.

– for $n \geq 1$,
  - **Intra-class-variance-minimization:** define $x_n^1, \ldots, x_n^\lambda$ a family of λ points minimizing $E_w(\inf_i ||x^*(w) - x_n^i||^2|\sigma)$ where σ is the ranking of all previously visited points.
  - compute $y_n^i = fitness(x_n^i)$.

As pointed out above, with $A(0)$, for some domain and some prior distribution, we might get $x_n = x_{n-1} = \cdots = x_0$. With B, we believe that no pathological behavior arises. Therefore, we consider B and not some noisy $B(\epsilon)$.

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

The expectation (w.r.t random variable w, conditionally to the ranking of previously visited points) defining x^σ in algorithm A, that has been shown an optimal estimate of the arg min for the worst case on g, can be computed by *ergodic billiard*. Ergodic billiard can be used as in bayesian inference (see e.g. [8]). Consider a uniform prior probability for w on some set E. E is defined by a set of constraints.

– find one point w^0 that satisfies the ranking of $x_1, \ldots, x_n$.
– choose randomly one direction d^0 in the unit sphere of $\mathbb{R}^d$.
– for $n = 0, \ldots, \infty$
  - generate $w^{n+1} = w^n + \lambda d^n$ with the smallest $\lambda > 0$ such that at least one constraint among the followings becomes active:
    - ∗ constraints ensuring that w^{n+1} is such that $sign(f(x_i, w^{n+1}) - f(x_j, w^{n+1})) = sign(y_i - y_j)$;
    - ∗ constraints ensuring that $w^{n+1} \in E$.
  - get d^{n+1} by symetrization of the direction d^n w.r.t active constraints.
– output the weighted average x^σ of the $x^*([w^n, w^{n+1}])$, weighted by the length of $[w^n, w^{n+1}]$.

If the billiard is ergodic, what is not proved but conjectured at least for constraints in general position, the sequence of segments $[w^n, w^{n+1}]$ weighted by the prior probability on w approximate the posterior probability conditionally to the constraints. We simply take the average value of the $x^*(w)$ associated to this ergodic billiard as a mean-square approximation x^σ of the optimum.

We can also consider the case of λ points generated simultaneously (algorithm B, corresponding to theorem 2'). As the ergodic billiard provides points uniformly distributed in the domain, k-means can be used to find the x^i_n by minimization of the intra-class-variance. 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 our algorithm for a "greedy" criterion, in which the ultimate goal is always the next iterate. This might be very far from the optimality for, e.g. 100 iterates. This problem will be discussed in the experimental section only.

3 Experiments

We present experiments with the algorithms BEDA and BREDA defined in 2.2.

Combining results from [17] and results above, we can conclude 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 most $1 - O(1/d)$ with respect to the number of comparisons;
- for some simple fitness functions, this convergence rate is achieved uniformly on compositions with increasing functions, and, at the worst case among such compositions, comparison-based methods are indeed optimal;
- for the greedy criterion defined in section 2.2, bayesian optimization conditionally to ranks is optimal.

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

Theorems 1 and 2 have shown how to build, for any fitness-based optimization-algorithm and any fitness-function, a fitness that is the image of this fitness by some $g \in G$, and for which the algorithm can not be better than some comparison-based algorithm. Section 3.1 shows that robustness with respect to the worst case of $g \in G^0$ is sufficiently well approximated by the construction of g as in the proof of theorem 1 to strongly disturb some standard non-comparison-based algorithms. Section 3.2 then experiments the efficiency of our billiard based algorithm in front of some other algorithms.

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

We below consider optimization of a fitness $g \circ fitness$ with g defined as in the proof of theorem 1, where $fitness$ is one of the fitness functions in [16]. Each optimizer works on $g \circ fitness$, but for the sake of comparison, as g is dependent of the optimizer, the result reported below is the best value of $fitness$ on points visited by the algorithm.

Precisely, the experimental setup is as follows:

– consider *Opt* an optimizer and $fitness$ a fitness function.
– use *Opt* on $g \circ fitness$, where g is built as in theorem 1.
– the result is $r = fitness(x)$, where x is the best visited point for $fitness$ (or equivalently, the best visited point for $g \circ fitness$).

The expectation of r (which is random if $fitness$ is random), is therefore the expectation of the result for $g \circ fitness$, where g is built as in theorem 1; this is a lower bound on the expectation of r associated to $g \circ fitness$ for the worst case on increasing transformations of g. The results show that this approximation is sufficient to strongly modify the relative efficiency of algorithms.

LBFGSB is the Limited-memory Box-constrained BFGS from [20]. Random is the naive random search. GAO is the simple genetic algorithm defined in [5]. HJ is the Hooke&Jeeves algorithm ([10, 12, 19], implementation available at `http://www.ici.ro/camo/unconstr/hooke.htm`). CMAES is the covariance-matrix-adaptation algorithm from [7, 4] (Beagle version 3.0.1), with $\lambda = 2\lfloor(4. + \lfloor 3. * ln(dimension)\rfloor)/2\rfloor$. LBFGSB here uses finite differences and is the only algorithm that does not only depend 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 0 is $x \mapsto ||x-w||^{1/4}$ with w uniformly distributed in the unit ball, functions 1 to 6 are the unimodal functions in the Cec05 benchmarks. The number presented is the average fitness after 256 fitness-evaluations, averaged over 33 runs. As LBFGSB does not only depend 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 (this transformation is suitable in the proof of theorem 1 because the number of time steps is finite, but not for the corollary about convergence rates). The comparisons for fitnesses 1-6 are moderately significant, as the deterministic algorithm BFGS can not provide standard deviations for deterministic fitnesses 1-6 and standard deviations for stochastic algorithms are moderately informative for deterministic fitness functions, but the conclusion according to which when g is applied LBFGSB is outperformed by GAO, HJ and CMAES for fitnesses f0 is significant, and the fact that this is reproduced for each fitness among f1,f2,f3,f4,f5 is significant. Therefore we conclude from these experiments, and in accordance with theory above, (i) that the robustness w.r.t. g is not verified by BFGS even in practice, (ii) that worst-case on g make even very easy functions untractable by non-rank-invariant algorithms, (iii) that the procedure defined in the proof of theorem 1 is efficient for finding hard-fitnesses.

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

	LBFGSB	Random	GAO	HJ	CMAES
f0	0.266 / 0.524	0.562	0.366	0.179	0.367
	± 0.075 /0.045	± 0.042	± 0.068	± 0.055	± 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 fitnesses 0, 3 and 6, the best with ex-aequo for fitnesses 1,2, 5, and outperformed by comparison-based algorithms only for fitness 4. When g is applied, LBFGSB is worse than random search for fitnesses 0,1,2,4,5, and worse than GAO or CMAES for all fitnesses. We see that the non-differentiability for fitness 0, which comes from the application of $x \mapsto x^{1/8}$ to a differentiable fitness, is not a big trouble for LBFGSB, whereas function g built from theorem 1 is much harder.

	LBFGSB	Random	GAO	HJ	CMAES
f0	0.199 / 1.095	1.037	0.709	0.255	0.816
	± 0.020 /0.010	± 0.002	± 0.007	± 0.003	± 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 fitnesses 0, 1, 2, 3, and also outperforms CMAES for fitness 6. 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 fitnesses 0, 1, 2, 4, 5.

	LBFGSB	Random	GAO	HJ	CMAES
f0	1.095 / 1.357	1.346	1.214	0.614	1.300
	$\pm$ 0.009 / 0.003	$\pm$ 0.000	$\pm$ 0.001	$\pm$ 0.003	$\pm$ 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 f1 function. 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, can not work efficiently with very moderate number of fitness-evaluations (as the gradient is not available, finite differences are applied, therefore one iterate costs 51 fitness-evaluations). When a transformation g as 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).

3.2 Results on the sphere function

We compare (i) our algorithm with billiard (ii) our algorithm without billiard, only using a point provided by the stochastic gradient algorithm for finding w satisfying constraints (iii) an algorithm using billiard in order to generate one point according to the distribution-probability of $\arg\min f(., w)$. Results are presented in figures 1 and 2.

Results show that (i) the algorithm is much more efficient than various existing algorithms; (ii) choosing always x^σ as next point is not a good solution. Randomly choosing a possible candidate, according to the posterior probability, is better. This is in favor of random-diversification.

4 Conclusion

It has been shown in [17] that comparison-based algorithms are slow. We here show (theorem 1 and corollaries, section 2.1) that comparison-based algorithms are however as fast as any fitness-based method for the worst case among C^∞-increasing transformations of the fitness. We also show (section 2.2) that an optimal algorithm can be designed for a distribution of fitness functions on the worst case among increasing transformations of the 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-Free-Lunch theorems

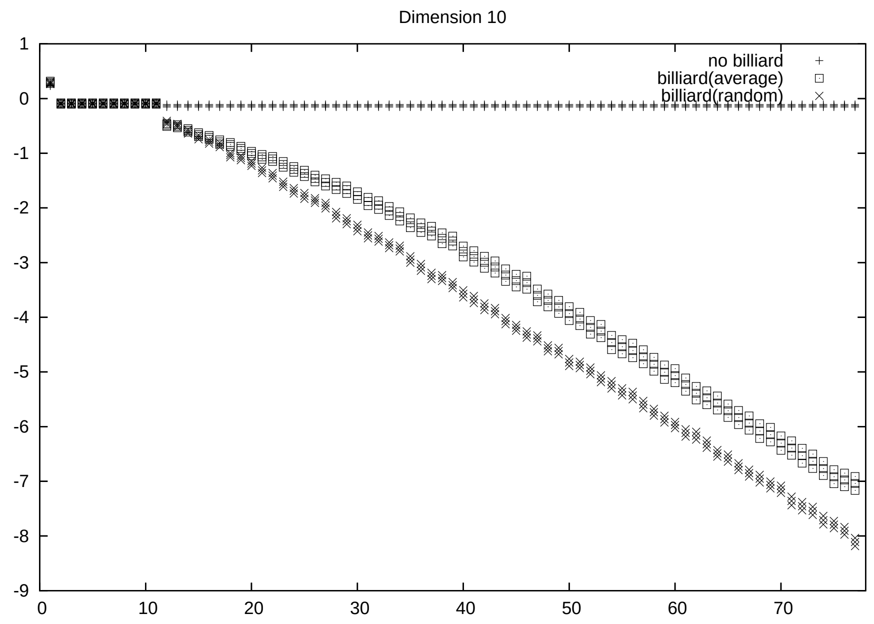

Fig. 1. Results in dimension 10: we present the ln (natural logarithm) of the distance to optimum versus the number of function-evaluations, $\pm$ standard deviation. The case in which the first point consistent with the ranking of previous points found by the stochastic is used as next iterate does not converge. The random generation in the domain is better than the average solution, in spite of the fact that the latter is optimal in a greedy sense. With 256 function-evaluations, Hooke&Jeeves and CMAES reach respectively -5.532 ± 0.78 and -1.692 ± 0.75 (-1.736 ± 0.61 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, is a comparison-based algorithm much better with 70 fitness-evaluations than Hooke&Jeeves and CMAES (parametrized as explained in the text) with 256 fitness-evaluations in dimension 10.

that show that priors are necessary for proving that an algorithm is better than another.

Interestingly, we show in this framework, that informations beyond ranking information *must* be ignored by an optimal optimization algorithm (theorems 2 and 2'). Section 2.2 also presents intra-class-minimization tools as an optimal paradigm for generating a population when a distribution of possible fitness-functions is available (this is not restricted to comparison-based methods, but to the general case of Estimation-Of-Distribution-algorithms). Section 2.3 shows that ergodic billiard can be used to implement the proposed algorithm, together with k-means when generating λ-points at once. The two resulting algorithms are not fully theoretically analyzed, as only "greedy" properties have been shown, but experiments show the relevance of the approach for frugal optimization frameworks. We conclude (i) that comparison-based algorithms are

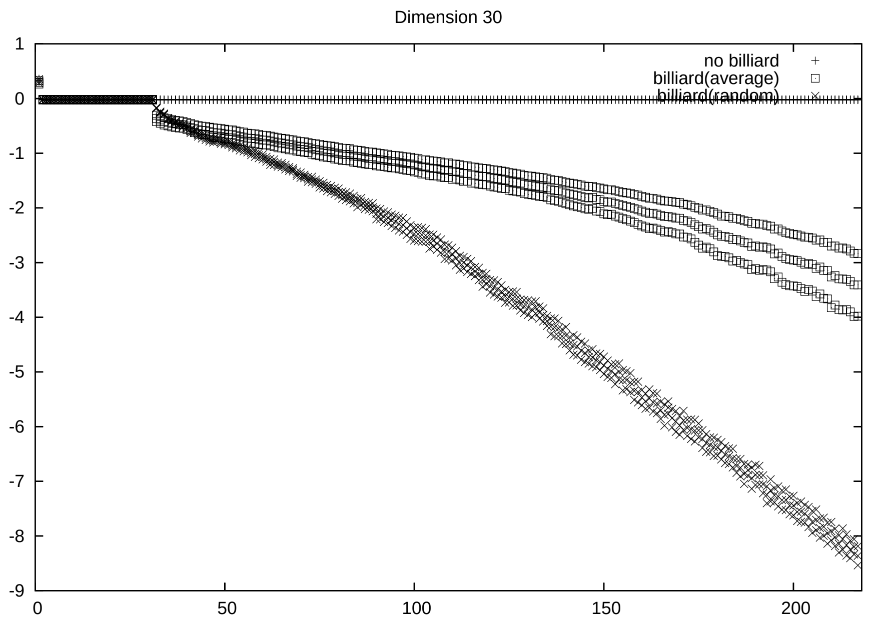

Fig. 2. Results in dimension 30. With 256 function-evaluations, Hooke&Jeeves and CMAES reach respectively -1.944 ± 0.39166 and 0.556 ± 0.17. The billiard approaches BEDA and BREDA clearly outperforms other algorithms; in particular, the difference with existing algorithms is much larger than in lower dimension. BREDA is more efficient than BEDA.

optimal in some robust frameworks (ii) that greedy-optimal algorithms can be associated to bayesian priors through an estimation of distribution of optima (iii) that greedy-optimal algorithms for comparison-based methods can be implemented efficiently with ergodic billiards.

As a conclusion, we summarize the practical implications of our work as follows:

– optimal algorithms, for a given prior, can be defined (section 2.2). We believe that these optimal algorithms are practical in the case of very expensive fitness-functions, and that their approximation by faster algorithms is an important research area for the future.
– also, an a priori on the distribution of fitnesses leads, in a simplified greedy framework, to optimal algorithms that are tractable (via billiards techniques) at least for moderate number of fitness-evaluations; TODO tester ca sur une fitness dure j'y crois dur comme fer
– as already shown in [17], comparison-based methods are slow when the dimension is large, at least when the computation-time of comparisons is not negligible in front of the computation-time of fitnesses;
– however, they are optimal in a natural robustness-framework;

– families of fitnesses are an interesting framework for optimization; whereas optimizing on a finite family of fitnesses can lead to very specific algorithms that typically only sample the finite set of possible minima, optimizing in front of a distribution of fitnesses is non-trivial and optimality (for a given number of iterates) can be properly defined. Therefore, we prefer stochastic families of fitnesses (possibly simply by a random rotation and/or translation), without a previously defined set of random seeds. A finite set (or a deterministic sequence) of random seeds is usefull as pairing improves the significance of comparisons, but it can introduce biases. Some cases of random landscapes can be found in TODO refs dont ppsn
– theorem 2' suggests that generations of λ points in EDA should be done in a derandomized manner; in theorem 2', it is shown that generating points according to an inter-class variance criterion is optimal in a greedy sense. Minimizing the inter-class variance is computationnaly hard, but a reasonnable approximation is the use of quasi-random numbers. This has been successfully tested in [13] in moderate dimension TODO verifier, and until dimension 50 (without decay of results beyond dimension 50) in `http://www.lri.fr/~teytaud/resultsDCMA.pdf` with scrambled quasi-random sequences (quasi-random sequences that are enhanced by some moderate randomization) ; see also [2].

Further works include (i) the possible use of Markov-Chain-Monte-Carlo (instead of ergodic billiards) for optimal optimization algorithms if we are not in the comparison-based case (ii) the design of better optimality criterions and their implementation through stochastic dynamic programming techniques (as defined in section 2.2) (iii) other approximations of such algorithms. Also, we did not succeded in generalizing theorem 1 to randomized algorithms; investigations are possible in this direction.

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. A. Auger. Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible markov chains. *Theoretical Computer Science*, 334:35–69, 2005.
2. A. Auger, M. Jebalia, and O. Teytaud. Xse: quasi-random mutations for evolution strategies. In *Proceedings of Evolutionary Algorithms, 12 pages*, 2005.
3. S. Droste. 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, 2005.
4. C. Gagné and M. Parizeau. Open BEAGLE: A new versatile C++ framework for evolutionary computations. In *Late breaking papers of the GECCO 2002*, pages 161–168, July 2002.

5. S. Gelly, O. Teytaud, and C. Gagné. Resource-aware parameterizations of EDA. In *Proc. of the 2006 IEEE Congress on Evolutionary Computation (IEEE-CEC 2006)*, July 16-21 2006. http://www.lri.fr/∼teytaud/tsm2.pdf.
6. D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning.* Addison Wesley, 1989.
7. N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 11(1), 2003.
8. R. Herbrich, T. Graepel, and C. Campbell. Bayes point machines. *Journal of Machine Learning Research*, 1:245–279, 2001.
9. J. H. Holland. *Adaptation in natural and artificial systems.* University of Michigan Press, Ann Arbor, 1975.
10. R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM, Vol. 8, pp. 212-229*, 1961.
11. J. Jagerskupper and C. Witt. Runtime analysis of a (mu+1)es for the sphere function. Technical report, 2005.
12. A. F. Kaupe. Algorithm 178: direct search. *Commun. ACM*, 6(6):313–314, 1963.
13. S. Kimura and K. Matsumura. Genetic algorithms using low-discrepancy sequences. In *GECCO*, pages 1341–1346, 2005.
14. Y. Nikulin. Robustness in combinatorial optimization and scheduling theory: An annotated bibliography. *Optimization online*, 2004. http://www.optimization-online.org/DB_HTML/2004/11/995.html.
15. G. Rudolph. Convergence rates of evolutionary algorithms for a class of convex objective functions. *Control and Cybernetics*, 26(3):375–390, 1997.
16. P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report AND KanGAL Report #2005005, IIT Kanpur, India, 2005.
17. O. Teytaud and S. Gelly. General lower bounds for evolutionary algorithms. In 10th *International Conference on Parallel Problem Solving from Nature (PPSN 2006)*, 2006.
18. D. Wolpert and W. Macready. No free lunch theorems for search. Technical report, Santa Fe Institute, 1995.
19. M. Wright. 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, 1995. http://citeseer.ist.psu.edu/wright95direct.html.
20. C. Zhu, R. Byrd, P.Lu, and J. Nocedal. L-BFGS-B: a limited memory FORTRAN code for solving bound constrained optimization problems. *Technical Report, EECS Department, Northwestern University*, 1994.