# Comparison-Based Complexity of Multiobjective Optimization

Olivier Teytaud
Tao Inria, Lri
Umr Cnrs 8623
Paris-Sud F-91405 Orsay Cedex
olivier.teytaud@inria.fr

## ABSTRACT

Several comparison-based complexity results have been published recently, including multi-objective optimization. However, these results are, in the multiobjective case, quite pessimistic, due to the huge family of fitness functions considered. Combining assumptions on fitness functions and traditional comparison-based assumptions, we get more realistic bounds emphasizing the importance of reducing the number of conflicting objectives for reducing the runtime of multiobjective optimization. The approach can in particular predict lower bounds on the computation time, depending on the type of requested convergence: pointwise, or to the whole Pareto set. Also, a new (untested yet) algorithm is proposed for approximating the whole Pareto set.

## Categories and Subject Descriptors

G.1.6 [**Optimization**]: Global Optimization

## General Terms

Theory

## 1. INTRODUCTION

The comparison-based model[22, 7] is a recent tool for modeling evolutionary computation. Most evolutionary algorithms are comparison-based, and it is known that comparison-based black-box complexity is equivalent to black-box complexity for some invariant families of fitness functions[8].

[20] has shown lower complexity bounds for multiobjective optimization. Unfortunately, the results therein, based on a worst case analysis on very wide families of objective functions, are quite pessimistic and exponential in the number of objectives. [18] has shown a runtime analysis for multiobjective optimization, using the results in [24] on the mono-objective case, with much more realistic rates, thanks to assumptions on fitness functions, supposed to be quadratic. We will here keep their assumptions on families of functions; basically, their model is a natural extension of the classical sphere function to the multiobjective case.

This paper extends results from [18], limited to small values of the offspring size $\lambda$ and the number $d$ of objective functions, by using comparison-based complexity results from [22]. Importantly, the results emphasize the importance of the number of conflicting objectives in the runtime analysis.

Section 2 introduces comparison-based complexity. Section 3 recalls fundamental lower bounds in this setting (basically, the convergence is at best linear in log-scale, and the constant decays linearly as a function of the dimension). Section 4 provides lower bounds for the multiobjective case: essentially, there is, for convergence to the whole Pareto set, an additional factor linear as a function of the number of conflicting objectives. Section 5 shows corresponding upper bounds on the comparison-based complexity. Section 6 summarizes the paper and discusses limitations and further work.

## 2. COMPARISON-BASED ALGORITHMS OF TYPE $(\mu \dagger \lambda)$

This section is devoted to a formal definition of comparison-based multi-objective optimization algorithms. The maths are strongly based on [22, 21, 7], but the application to multi-objective optimization is new and provides interesting specific hints.

We here follow the lines of [7] for defining four classes of evolutionary algorithms: Selection-Based Non-Elitist Strategies, Selection-Based Elitist Strategies, Full-ranking Non-Elitist Strategies, Full-Ranking Elitist Strategies. Let $\lambda$ and $\mu$ be two integers. The first case is the case of Selection Based evolution strategies (SB-$\mu$, $\lambda$-ES). In this case, there is a set $\mathcal{I}$ of internal states. The algorithm starts in the initial state $I_0$ returned by the function *initial_state*. At each iteration, the algorithm follows these three successive steps. First generate a set of $\lambda$ points, called the *offspring*. Then select only $\mu$ of these points, by any procedure, as a function of the fitness values. Finally, the internal state is updated. In the mono-objective case, this includes for example cumulative step-length adaptation (CSA) by [9], covariance matrix adaptation (CMA) by [9], mutative self-adaptation (SA) by [17, 19], covariance matrix adaptation by SA (CMSA) by [3], the one-fifth rule for step-size adaptation [17, 2]. In the multi-objective case, it includes NSGA-II[6], SPEA2[25]. Techniques which use gradients or surrogate models are not concerned.

SB-$(\mu + \lambda)$-ES (the second case, termed elitist) selects $\mu$


*GECCO'11,* July 12–16, 2011, Dublin, Ireland.

points among the offspring *and* the points selected at the previous step.

Please notice that the framework is very general: it includes adaptation by cross-entropy methods (CEM by [5]) or Estimation of Distribution Algorithms like e.g. UMDA [15], Compact Genetic Algorithm [10], Population-Based Incremental Learning [1], Relative Entropy [14] and Estimation of Multivariate Normal Algorithms (EMNA by [13]), Direct Search Methods[4], the Nelder-Mead algorithm[16], the Hooke&Jeeves algorithm[11] - and MOO techniques based on these algorithms, without surrogate model learning. This points out the links between all these algorithms.

**Algorithm 1** Selection Based $(\mu, \lambda)$-ES (resp. Selection Based $(\mu+\lambda)$-ES). Framework for evolution strategies based on selection, working on a fitness function $f$. The random variable $\omega$ is a random seed. An algorithm matching this framework is obtained by specifying the distribution of $\omega$, the space of states, and the functions *initial_state*, *generate*, *update* and *proposal*. $x^{(f)}_{\omega,n}$ is the approximation of a solution proposed by the algorithm after $n$ iterations.

**Initialization:** $I_0 \leftarrow \text{initial\_state}(\omega)$; $S_0 \leftarrow \emptyset$; $n \leftarrow 0$
**while** true **do**
  $n \leftarrow n+1$
  **Generation step** (generate an offspring $O_n$ of $\lambda$ distinct points)**:** $O_n \leftarrow \text{generate}(I_{n-1})$
  $E_n \leftarrow O_n$ (resp. $E_n \leftarrow O_n \oplus S_{n-1}$ for $(\mu+\lambda)$-ES)
  $\ell \leftarrow \min(\mu, |E_n|)$
  **Selection step:** $v_n \leftarrow \text{select}(E_n, f)$, i.e.:
    The vector $v_n = (i_1, \ldots, i_\ell)$ is the vector of indices of selected points.
    (i.e. an increasing sequence of length $\ell$ in $[[1, \min(\mu, |E_n|)]]$
  )
  **Update the internal state:** $I_n \leftarrow \text{update}(I_{n-1}, E_n, v_n)$
  $S_n \leftarrow (E_{n,i_1}, \ldots, E_{n,i_\ell})$
  $x^{(f)}_{\omega,n} \leftarrow \text{proposal}(I_n)$
**end while**

We present SB-$(\mu, \lambda)$-algorithms (resp. SB-$(\mu + \lambda)$-algorithms) in Algorithm 1. In this algorithm (also in Algorithm 2), the concatenation of the two vectors $x = (x_1, \ldots, x_k)$ and $x' = (x'_1, \ldots, x'_\ell)$ is denoted by $x \oplus x' = (x_1, \ldots, x_k, x'_1, \ldots, x'_\ell)$; we also use the shortcut $v \in (x_1, \ldots, x_k)$ to express that there exists $i \in [[1, k]]$ such that $x_i = v$.

Notice that the length of the vector $v_n$ is equal to $\mu$ except maybe during the first iterations of the algorithm in the case $\lambda < \mu$: this explains the use of the auxiliary variable $\ell$ in Algorithm 1 for the $(\mu + \lambda)$-case.

Finally, we would like to explain a generalization of SB-$(\mu \dagger \lambda)$-ES, called Full Ranking $(\mu \dagger \lambda)$-ES (FR-$(\mu \dagger \lambda)$-ES). Instead of just giving the best $\mu$ points (i.e., the $\mu$ points with the lowest fitness values), we can consider a selection procedure which returns the best $\mu$ points *ordered with respect to their fitness*.

The outline of these algorithms is given in Algorithm 2. More precisely, the selection step described in this algorithm works as follows. Given the vector of points $E_n = (z_1, \ldots, z_p)$ considered at step $n$, the function *select* returns a vector of $\mu$ distinct integers $v_n = (i_1, \ldots, i_\ell)$, but as the full ranking matters, there's no constraint on $v_n$; it's just a vector of $\ell$ distinct points in $[[1, p]]$ (Once again, the length of the vector $v_n$ may not be equal to $\mu$ at the beginning of the algorithm in the "+" case).

**Algorithm 2** Full Ranking $(\mu, \lambda)$-ES (resp. Full Ranking $(\mu + \lambda)$-ES). Framework for evolution strategies based on full ranking, working on a fitness function $f$. The random variable $\omega$ is a random seed. Compared to Algorithm 1, the vector of integers $v_n$ obtained at the selection step is now ordered with respect to the fitness values of points from $E_n$; this framework is thus more general.

**Initialization:** $I_0 \leftarrow \text{initial\_state}(\omega)$; $S_0 \leftarrow \emptyset$; $n \leftarrow 0$
**while** true **do**
  $n \leftarrow n+1$
  **Generation step** (generate an offspring $O_n$ of $\lambda$ distinct points)**:** $O_n \leftarrow \text{generate}(I_{n-1})$
  $E_n \leftarrow O_n$ (resp. $E_n \leftarrow O_n \oplus S_{n-1}$)
  $\ell \leftarrow \min(\mu, |E_n|)$
  **Selection step:** $v_n \leftarrow \text{select}(E_n, f)$, i.e.:
    The vector $v_n = (i_1, \ldots, i_\ell)$ is the vector of indices of selected points
    (i.e. an increasing sequence of length $\ell$ in $[[1, \min(\mu, |E_n|)]]$
  **Update the internal state:** $I_n \leftarrow \text{update}(I_{n-1}, E_n, v_n)$
  $S_n \leftarrow (E_{n,i_1}, \ldots, E_{n,i_\ell})$
  $x^{(f)}_{\omega,n} \leftarrow \text{proposal}(I_n)$
**end while**

Note that both Algorithms 1 and 2 define a class of algorithms: in order to obtain an algorithm, one has to specify the distribution of $\omega$, how the offspring is generated (function *generate*), the space of states $\mathcal{I}$ as well as the functions *initial_state* and *update*, and finally the function *proposal*. Throughout the chapter, we assume that all functions involved in these algorithms are measurable.

Importantly, in spite of the formulation in Algorithms 1 and 2, the formalization is not restricted to deterministic algorithms. In Algorithms 1 and 2, all steps are deterministic, but there is an initial source of randomization, $\omega$, which can be as large as required, e.g. it might be an infinite sequence of realizations of a random Gaussian variables or random uniform variables. So this is not a restriction, the algorithm must just be rephrased accordingly, so that $\omega$ is reported in the internal state and used in *e.g.* the "proposal" function.

# 3. COMPLEXITY LOWER BOUNDS

We consider a bounded domain $E$ and a distance $dist : E \times E \to \mathbb{R}$ on $E$ or on a superset of $E$. For $\varepsilon > 0$, we define $N_E(\varepsilon)$ to be the maximum integer $n$ such that there exist $n$ distinct points $x_1, \ldots, x_n \in D$ with $dist(x_i, x_j) \geqslant 2\varepsilon$ for all $i \neq j$. $N_E(\varepsilon)$ is termed the packing number, and quantifies the "size" and "dimensionality" of the domain as explained below. In particular, $N_E(\varepsilon) = |E|$ when $\varepsilon$ is small enough in the case of a finite domain $D$, and $\log N_E(\varepsilon) \sim z \log(1/\varepsilon)$ when $\varepsilon \to 0$ if the domain $E$ is bounded with non-empty interior [12] of dimension $z$. Please notice that these definitions are consistent both for continuous domain $E \subset \mathbb{R}^z$ and discrete domain $E \subset \mathbb{R}^z$.

We denote by $x^*(f) \subset D$ the set of optima. It can be a point, in particular in the mono-objective case, or a multi-objective algorithm in which the Pareto set is restricted to one point; but, in the general case, we get a set of optima, a Pareto set of dimension typically equal to the number of objectives minus one. For keeping generality in our results, we will need a metric on subsets of $D$. Typically, $x^{(f)}_{\omega,n}$ is the proposal by the multi-objective algorithm - it can be an approximation of the target Pareto set, or it can be just a

point, if the algorithm is intended to propose a pointwise convergence. Then, $dist(x^{(f)}_{\omega,n}, x^*(f))$ can be:

- $\inf_{x\in x^*(f)} ||x - x^{(f)}_{\omega,n}||$ for a pointwise convergence (i.e. $x^{(f)}_{\omega,n}$ is a point).
- The Hausdorff distance between $x^{(f)}_{\omega,n}$ and $x^*(f)$ (i.e. $x^{(f)}_{\omega,n}$ is a set).

The precise definition of the metric does not matter for the moment, and we'll discuss these examples later.

For any given optimization algorithm as in Algorithm 2, and for $\varepsilon > 0$ and $\delta > 0$, we define $n_{\varepsilon,\delta}$ be the minimum number $n$ of iterations such that with probability at least $1 - \delta$, an optimum is found at the $n$-th iteration within distance $\varepsilon$; i.e., $n_{\varepsilon,\delta}$ is minimal such that for all $n \geqslant n_{\varepsilon,\delta}$ and for all $f \in \mathcal{F}$,

$$Pr_\omega[dist(x^{(f)}_{\omega,n}, x^*(f)) < \varepsilon] \geqslant 1 - \delta.$$

A crucial notion from [23] is the *branching factor* $K$ of algorithms as defined above.

$$K = \sup_E |\{\text{select}(E, f) \ : \ f \in \mathcal{F}\}|,$$

where the supremum holds for:

- $E$ any set of size at most $\lambda$ in the case of SB-$(\mu, \lambda)$-ES or Full Ranking $(\mu, \lambda)$-ES;
- $E$ any set of size at most $\mu+\lambda$ in the case of SB-$(\mu+\lambda)$-ES or Full Ranking $(\mu + \lambda)$-ES.

Various bounds on $K$ can be derived[22]:

$$K \leqslant \binom{\mu}{\lambda} \leqslant 2^\lambda/\sqrt{2\pi\lambda} \text{ for } SB-(\mu,\lambda)-ES; \quad (1)$$

$$K \leqslant \binom{\mu}{(\lambda+\mu)} \text{ for } SB-(\mu+\lambda)-ES; \quad (2)$$

$$K \leqslant \lambda!/(\lambda-\mu)! \text{ for } FR-(\mu,\lambda)-ES; \quad (3)$$

$$K \leqslant (\lambda+\mu)!/\lambda! \text{ for } FR-(\mu+\lambda)-ES. \quad (4)$$

THEOREM 1 (LOWER BOUND HITTING TIME FOR $(\mu \dot{+} \lambda)$-ES.). *Consider a $(\mu, \lambda)$-ES or $(\mu + \lambda)$-ES, as defined in Algorithm 2, and a set $\mathcal{F}$ of fitness functions on domain $D$, i.e. $\mathcal{F}$ is a set of functions from $D$ to $\mathbb{R}^d$, i.e. $\mathcal{F} \subset (\mathbb{R}^d)^D$, and define $x^*(f)$ the optimum of $f$ (this is a set, possibly restricted to a point), and such that $\{x^*(f) \ : \ f \in \mathcal{F}\} = D$. Let $\varepsilon > 0$ and $\delta \in ]0, 1[$. In particular, if $K$ denotes the branching factor of the algorithm, then*

$$n_{\varepsilon,\delta} \geqslant \frac{\log(1-\delta)}{\log K} + \frac{\log N(\varepsilon)}{\log K}.$$

In the next section, we will see the consequences of this theorem in multi-objective optimization.

# 4. RESULTS IN MULTIOBJECTIVE OPTIMIZATION

The section above, based on earlier results in comparison-based optimization and inspired by communication complexity, provided general results which can be used in many frameworks around comparison-based algorithms, and also around iterative algorithms with bounded branching factor. We here specialize the analysis in the case of multi-objective optimization.

Section 4.1 will provide necessary tools in terms of packing numbers for later sections. Section 4.2 will provide general results in terms of computation time before reaching a given precision, and section 4.3 will provide the main results, namely the number of fitness evaluations before reaching a given precision.

## 4.1 Packing numbers

Packing numbers go back to [12]. A basic case is $N_E(\varepsilon) = \Theta(1/\varepsilon^z)$ for $E$ an open bounded subset of $\mathbb{R}^z$; many cases are derived from this one. We assume in the rest of this paper that $N \geqslant d$, and we work on a search space $D = [0, 1]^N$.

**Standard case.** If $E$ is equal to $[0, 1]^N$ and the distance is the standard euclidean metric, then we get $N_E(\varepsilon) = \Theta(1/\varepsilon^N)$. This is widely used in monoobjective optimization[22, 20]. We now switch to more important cases (more important for the multiobjective case), namely cases involving sets (which will be, for us, Pareto sets).

**Sphere functions.** If $E$ is the set of possible Pareto sets in $[0, 1]^N$ defined by $d$ spherical objective functions with optima in $[0, 1]^N$, i.e.

$$E = \{convexHull(x_1, \dots, x_d); (x_1, \dots, x_d) \in \left([0, 1]^N\right)^d\} \quad (5)$$

and if $E$ is equipped with the Hausdorff metric, then

$$N_E(\varepsilon) = \Theta(1/\varepsilon^{Nd}). \quad (6)$$

This is proved by considering the convex hulls of sets of size at most $d$ (which are exactly the Pareto sets corresponding to $d$ sphere functions).

**Sphere functions, pointwise convergence.** If $E$ is the set of possible Pareto sets in $[0, 1]^N$ defined by $d$ spherical objective functions and equipped with

$$d(a, b) = \inf_{y\in b} ||a - y|| \quad (pointwise\ metric); \quad (7)$$

i.e. we have the same set of possible Pareto sets as above, but not the same distance), then the situation is more complicated: the algorithm is only allowed to provide a point, whereas the target is a set (the Pareto set); this dissymetry implies, for a branching factor theorem, that we need a slightly adapted definition of $N_E(\varepsilon)$, namely $N_E(\varepsilon)$ is the minimum number of points $x_1, \dots, x_{N(\varepsilon)}$ in $D$ such that for all $e \in E$, there is $i$ such that $d(x_i, e) \leqslant \varepsilon$. This is a covering number; covering numbers can be used as well as packing numbers in branching factor theorems.

Furthermore, we have to distinguish different cases. Converging to just one point of the Pareto set can be made by optimizing just one objective function (provided that there is an objective function different from all others), and we get the same convergence as for mono-objective functions. A more interesting question is what happens when the objective functions are constrained to be different. We therefore define the set $E$ as follows

$$E = \{intersections\ of\ affine\ subspaces\ of\ \mathbb{R}^N\ with\ [0, 1]^N\} \quad (8)$$

which means that the optima of the $d$ sphererical objective functions are sufficiently far away (outside the domain possibly - but obviously we request convergence only to the Pareto set in the domain), so that the Pareto set has its frontiers on the frontiers of the domain $D = [0, 1]^d$. We point out that considering smaller affine subspaces (e.g. included in $[a, b]^N$) would be sufficient for our results (both

upper bounds and lower bounds, in Table 1); we use $E$ as above for simplicity.

Then, we claim:

**Lemma 2.** *Consider $E$ as defined in Eq. 8. Then, for the pointwise metric (Eq. 7),*

$$N_E(\varepsilon) = \Omega(1/\varepsilon^{N+1-d}). \quad (9)$$

**Proof:** Consider the set

$$E' = \{(x_1, \ldots, x_{d-1}, 0, \ldots, 0); (x_1, \ldots, x_{d-1}) \in \mathbb{R}^{d-1}\}$$

$e'$ is isomorphic to $[0,1]^{N+1-d}$ and therefore $N_{E'}(\varepsilon) = \Omega(\frac{1}{\varepsilon}^{N+1-d})$. As $E' \subset E$, $N_{E'}(\varepsilon) \leqslant N_E(\varepsilon)$ and therefore $N_E(\varepsilon) = \Omega(1/\varepsilon^{N+1-d})$. □

## 4.2 Lower bound on the number of comparisons

The number of comparisons performed during an optimization run is always a lower bound on the computational cost; if the cost of the fitness function is low and if the internal cost of the optimization algorithm is low, then this becomes the main computational cost. There are other cases in which the number of comparisons is a very natural criterion: when it is the only available information, as *e.g.* when parametrizing a strategy for two-players games. The following result is an immediate consequence of Theorem 1; note that we have a 3 in the equations because the branching factor, for each comparison, is 3, as two points might lead to the dominance of the first by the second, or a dominance of the second by the first, or they might be uncomparable; there are 3 cases.

THEOREM 2 (W.R.T NB OF COMPARISONS). *Consider assumptions of Theorem 1 with $K = 3$ corresponding to the result of a comparison between the fitnesses of two visited points (the three outcomes are uncomparable, better than, worst than; we are in a multiobjective setting).*

*Then, with $\log_3(x) = \log(x)/\log(3)$, the number of comparisons $n_c$ required for ensuring with probability $1-\delta$ a precision $\varepsilon$ is $n_c \geqslant \log_3(1-\delta) + \log_3(N(\varepsilon))$. I.e., formally, $P(\|x_{n_c} - x^*(f)\| < \varepsilon) \geqslant 1-\delta \Rightarrow n_c \geqslant \log_3(1-\delta) + \log_3(N(\varepsilon))$.*

This very general result is relevant when the cost of the comparisons is not negligible. In many cases, the cost of the comparisons is negligible in front of the fitness evaluations - in next section, we will switch to complexity measures based on the number of fitness evaluations.

## 4.3 Lower bound on the number of function evaluations

We can now derive the main bounds. Table 1 applies Theorem 1 using Eq. 9 for the first row, and Eq. 6 for the second row, and Eqs 1-4 for bounding $K$.

# 5. UPPER BOUNDS ON THE COMPARISON-BASED COMPLEXITY

This section is devoted to the tightness of some dependencies in the bounds above.

We have shown (Theorem 1 and Eq. 6) that the runtime is lower bounded by $\Theta(Nd\log(1/\varepsilon))$ for convergence to the whole Pareto set for sphere functions. We have the following upper bound counterpart:

| Category of ES | SB- $(\mu,\lambda)$ | SB- $(\mu+\lambda)$ | FR- $(\mu,\lambda)$ | FR- $(\mu+\lambda)$ |
|---|---|---|---|---|
| Pointwise convergence to the Pareto set | $\frac{N+1-d}{\log\left(\frac{\mu}{\lambda}\right)}$ | $\frac{N+1-d}{\log\left(\frac{\mu}{\lambda+\mu}\right)}$ | $\frac{N+1-d}{\log\left(\frac{\lambda!}{(\lambda-\mu)!}\right)}$ | $\frac{N+1-d}{\log\left(\frac{(\lambda+\mu)!}{\lambda!}\right)}$ |
| Hausdorff convergence to the Pareto set | $\frac{Nd}{\log\left(\frac{\mu}{\lambda}\right)}$ | $\frac{Nd}{\log\left(\frac{\mu}{\lambda+\mu}\right)}$ | $\frac{Nd}{\log\left(\frac{\lambda!}{(\lambda-\mu)!}\right)}$ | $\frac{Nd}{\log\left(\frac{(\lambda+\mu)!}{\lambda!}\right)}$ |

**Table 1: Each equation is the coefficient of $\log(\varepsilon)$, within constant multiplicative factor, in the lower bound on the runtime; see Th. 1 for the complete lower bound (e.g. for pointwise convergence in the SB-$(\mu,\lambda)$-ES case, we get a lower bound $n_{\varepsilon,\delta} = \Omega\left(\frac{\log(1-\delta)}{\log K} + \frac{N+1-d}{\log\left(\frac{\mu}{\lambda}\right)}\log\varepsilon\right)$). $N$ is the dimension of the search space and $d$ is the number of objective functions; we assume $N \geqslant d$.**

**Proposition 1:** *For any set of $d$ spherical objective functions as defined in Eq. 5 and Hausdorff metric, algorithm 3 outputs an approximate Pareto set with precision $\varepsilon$ (for the Hausdorff metric) within runtime $O\left(\left(Nd\log(\frac{1}{\varepsilon})\right)\right)$ of the Pareto set.*

**Remark:** The constant in the $O(.)$ does not depend on the chosen spherical functions.

**Algorithm 3** Algorithm for approximating the whole Pareto set in the case of $d$ spherical objective functions. We assume that each $e_i$ is an instance of a comparison-based algorithm with runtime $\leqslant N\log(1/\varepsilon)$ in the monoobjective case (e.g. the Hooke& Jeeves pattern search algorithm), i.e. the number of fitness evaluations is $O(N\log(1/\varepsilon))$ for reaching precision $\varepsilon$. The convergence properties of variants of the Hooke&Jeeves algorithm can be found in Fournier et al.

Inputs: $f_1, \ldots, f_d$, which are $d$ spherical objective functions on $[0,1]^N$.
Initialize $e_1, \ldots, e_d$, some linearly converging comparison-based algorithms for mono-objective optimization.
**while** (true) **do**
  **for** $i \in [[1,d]]$ **do**
    Perform one iteration of algorithm $e_i$ on objective function $f_i$.
    Let $\hat{x}_i$ be the approximation of $\arg\min f_i$ provided by $e_i$.
  **end for**
  The approximate Pareto set is the convex hull of $\{\hat{x}_1, \ldots, \hat{x}_d\}$.
**end while**

**Proof:** There are monoobjective optimization algorithms with runtime bounded by $O(N\log(1/\varepsilon))$; see e.g. the proof in [7] for a variant of the Hoock&Jeeves algorithm.

Algorithm 3 applies such an algorithm to each of the $d$ objective functions. After $O(dN\log(1/\varepsilon))$ evaluations, we

get $d$ distinct optima within precision $\varepsilon$. The convex hull of these $d$ points is, within precision $O(\varepsilon)$, the Pareto set. □

**Proposition 2:** *For any set of $d$ spherical objective functions as defined in Eq. 8 and the metric defined in Eq. 8, algorithm 4 outputs a point at Euclidean distance at most $\varepsilon$ of the Pareto set within runtime $O\left((N+1-d)\log(\frac{1}{\varepsilon})\right)$.*

**Proof:** Algorithm 4 proceeds as follows:

- It spends $N+1$ function evaluations for finding a linear subspace of $\mathbb{R}^N$ generated by $N+1-d$ linearly independent vectors which are not parallel to the Pareto set.
- Then, it performs an optimization in this affine subspace by any comparison-based algorithm with runtime $O\left(dim(B)\log\frac{1}{\varepsilon}\right) = O\left((N+1-d)\log\frac{1}{\varepsilon}\right)$. □

**Algorithm 4** Algorithm for finding one point close to the Pareto set.

Inputs: $f_1, \ldots, f_d$, which are $d$ spherical objective functions on $[0,1]^N$.
Output: at each iteration, a point $\hat{x}$ hopefully converging to the Pareto set.
Perform a design of experiments as follows:
  Evaluate the fitness values at all points of the form $(0,0,\ldots,0,1,0,\ldots,0)$.
  If $f(0,0,\ldots,0,1,0,\ldots,0) \neq f(0,\ldots,0)$, then $(0,0,\ldots,0,1,0,\ldots,0)$ is not parallel to the Pareto set.
  Pick $N+1-d$ such vectors (which are not parallel to the Pareto set).
  Let $B$ be the vector space generated by these $N+1-d$ vectors.
**while** (true) **do**
  Perform one iteration of (e.g.) the Hoock&Jeeves algorithm in the vector space $B$ on fitness function $f_1$.
  Let $\hat{x}$ be the current iterate of this algorithm.
**end while**

# 6. CONCLUSIONS AND DISCUSSION

Our results show that for convergence to the whole Pareto set, the number of conflicting objectives has a (multiplicative) linear impact on the runtime, as well as the dimension of the search space (Proposition 1 and complexity bounds in Table 1, second row). On the other hand, for pointwise convergence, the runtime is linear in $N+1-d$; a high number of conflicting objectives makes the problem easier. This can be used for choosing between pointwise convergence and global convergence.

These intuitively satisfactory results are essentially a consequence of the realistic model used in [18].

There are several possible extensions:

- Further analysis could include an iterative model: monoobjective optimization for each optimum separately, and then successive choices by the user of weights of each fitness function.
- Extend the results to quadratic fitness functions and not only spherical fitness functions.
- Apply the VC-type results as in [7] for improving the results in the case of $\lambda$ large.

Also, whenever Algorithm 3 has been proposed for theoretical purposes (proving bounds), the idea of finding good points and then extrapolating them into an approximate Pareto set might be a good idea in the real world as well, probably with more than $d$ points however - experiment this is the most immediate further work.


### *Acknowledgments*

O. Teytaud is grateful to the LISC laboratory, where he wrote a part of this document. We are also grateful to Gunter Rudolph, Dimo Brockhoff, Marc Schoenauer and Dagstuhl's seminar 10361 for fruitful discussions.


# 7. REFERENCES


[1] S. Baluja. Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Technical Report CMU-CS-94-163, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1994.

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

[3] H.-G. Beyer and B. Sendhoff. Covariance matrix adaptation revisited - the CMSA evolution strategy. In G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni, and N. Beume, editors, *Proceedings of PPSN*, pages 123–132, 2008.

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

[5] P.-T. de Boer, D. Kroese, S. Mannor, and R. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, January 2005.

[6] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 849–858, Berlin, 2000. Springer.

[7] H. Fournier and O. Teytaud. Lower bounds for comparison based evolution strategies using VC-dimension and sign patterns. pages 1–22.

[8] S. Gelly, S. Ruette, and O. Teytaud. Comparison-based algorithms are robust and randomized algorithms are anytime. *Evolutionary Computation Journal (MIT Press), Special issue on bridging Theory and Practice*, 15(4):411–434, 2007.

[9] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[10] G. R. Harik, F. G. Lobo, and D. E. Goldberg. The compact genetic algorithm. *IEEE Trans. on Evolutionary Computation*, 3(4):287, November 1999.

[11] R. Hooke and T. A. Jeeves. "Direct search" solution of numerical and statistical problems. *Journal of the ACM*, 8(2):212–229, 1961.

[12] A.-N. Kolmogorov and V.-M. Tikhomirov. $\varepsilon$-entropy and $\varepsilon$-capacity of sets in functional spaces. *Amer. Math. Soc. Transl. 17, pp 277-364*, 1961.

[13] P. Larranaga and J. A. Lozano. *Estimation of*

*Distribution Algorithms. A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, 2001.

[14] H. Mühlenbein and R. Höns. The estimation of distributions and the minimum relative entropy principle. *Evolutionary Computation*, 13(1):1–27, 2005.

[15] H. Mühlenbein and T. Mahnig. Evolutionary computation and Wright's equation. *Theoretical Computer Science*, 287(1):145–165, 2002.

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

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

[18] G. Rudolph and N. Beume. Convergence rates of sms-emoa on continuous bi-objective problem classes. In *Foundations of Genetic Algorithms XI (FOGA)*, 2010.

[19] H.-P. Schwefel. Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit. Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik Re 215/3, Technische Universität Berlin, Juli 1974.

[20] O. Teytaud. On the hardness of offline multi-objective optimization. *Evolutionary Computation*, 15(4):475–491, 2007.

[21] O. Teytaud and H. Fournier. Lower bounds for evolution strategies using vc-dimension. In G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni, and N. Beume, editors, *PPSN*, volume 5199 of *Lecture Notes in Computer Science*, pages 102–111. Springer, 2008.

[22] O. Teytaud and S. Gelly. General lower bounds for evolutionary algorithms. In *Proceedings of PPSN*, volume 4193 of *Lecture Notes in Computer Science*, pages 21–31. Springer, 2006.

[23] O. Teytaud and S. Gelly. General lower bounds for evolutionary computation. In *proceedings of PPSN*, 2006.

[24] C. Witt and J. Jägersküpper. Rigorous runtime analysis of a (mu+1) es for the sphere function. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pages 849–856. ACM, 2005.

[25] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. Technical Report 103, Gloriastrasse 35, CH-8092 Zurich, Switzerland, 2001.