Adaptive Coordinate Descent

Ilya Loshchilov
TAO, INRIA Saclay
U. Paris Sud, F-91405 Orsay

Marc Schoenauer
TAO, INRIA Saclay
U. Paris Sud, F-91405 Orsay

Michèle Sebag
CNRS, LRI UMR 8623
U. Paris Sud, F-91405 Orsay

f rstname.lastname@inria.fr

ABSTRACT

Independence from the coordinate system is one source of efficiency and robustness for the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The recently proposed Adaptive Encoding (AE) procedure generalizes CMA-ES adaptive mechanism, and can be used together with any optimization algorithm. Adaptive Encoding gradually builds a transformation of the coordinate system such that the new coordinates are as decorrelated as possible with respect to the objective function. But any optimization algorithm can then be used together with Adaptive Encoding, and this paper proposes to use one of the simplest of all, that uses a dichotomy procedure on each coordinate in turn. The resulting algorithm, termed Adaptive Coordinate Descent (ACiD), is analyzed on the Sphere function, and experimentally validated on BBOB testbench where it is shown to outperform the standard $(1+1)$-CMA-ES, and is found comparable to other state-of-the-art CMA-ES variants.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence Problem Solving, Control Methods, and Search

General Terms

Algorithms

Keywords

Continuous Optimization, Adaptive Encoding, Line Search, Adaptive Coordinate Descent, Covariance Matrix Adaptation

1. INTRODUCTION

Separable continuous optimization problems are problems in which the objective function can be optimized coordinate-wise. Finding the global optimum of a separable function in $\mathbb{R}^d$ amounts to perform d simple *line searches* along each of the d coordinates. Unfortunately, interesting problems are usually not separable. Nevertheless, many optimization methods implicitly assume some form of separability of the objective function, or at least are much more efficient on separable functions as they explicitly use the coordinate system in their search. A well-known exception is the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [7], that performs a rotation-invariant search, and is thus independent of any coordinate system. The basic idea of CMA-ES is to evolve, besides a population of solutions to the optimization problem at hand, a "Covariance Matrix" that can be viewed as a coordinate transform: in case of a quadratic objective function, CMA-ES Covariance Matrix has been empirically demonstrated to gradually converge to the inverse Hessian matrix of the objective function. In the coordinate system defined by this inverse Hessian, the quadratic objective function has become separable, and the optimization problem, trivial. Of course, CMA-ES covariance matrix is only, in the quadratic case, an approximation of the inverse Hessian. And interesting problems are not quadratic indeed (and if they were, they would be easy to solve directly). Nevertheless, twice continuously differentiable objective functions can be viewed as close-to-quadratic around their optima (local or global), and adapting the coordinate system with respect to the "cumulative path" of the search makes it easier and faster to find the optimum.

The basic principles of this adaptive coordinate transformation have been generalized to general search strategies, under the name of *Adaptive Encoding* in [4], and experimented with Cauchy mutations in a stochastic search framework. The resulting optimization method is hence heavily coordinate-dependent, and its results deteriorate when the degree of non-separability of the objective function increases. However, this limitation of Cauchy mutation almost vanishes with Adaptive Encoding, demonstrating the usefulness of a well-designed adaptive coordinate system.

Putting things together, a natural idea is then to couple some simple optimization method, i.e., some successive coordinate-wise line searches, with Adaptive Encoding: coordinate line-searches only work well for separable functions, but Adaptive Encoding should gradually lead the search toward a transformed coordinate system where the objective function resembles more a separable function than in the original system, paving the road for the coordinate line-search. Though the resulting algorithm has little to do with Evolutionary Computation, it heavily relies on Adaptive Encoding, the backbone of CMA-ES algorithms.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for prof t or commercial advantage and that copies bear this notice and the full citation on the f rst page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specif c permission and/or a fee.
GECCO'11, July 12–16, 2011, Dublin, Ireland.
Copyright 2011 ACM 978-1-4503-0557-0/11/07 ...$10.00.

The paper is organized the following way: Section 2 first introduces the algorithmic background, namely Adaptive Encoding and some Coordinate Descent Method, before detailing their coupling into the Adaptive Coordinate Descent algorithm. Section 3 presents the experiments that validate Adaptive Coordinate Descent first on the Sphere function, the well-known separable test function, establishing performance bounds for the proposed approach. Extensive experiments on the BBOB testbench [5] are then presented and discussed. Finally, Section 4 concludes the paper and sketches directions for further researches.

2. ALGORITHMS

2.1 Adaptive Encoding

Though historically introduced as a derandomization of self-adaptive Evolution Strategies (ES) [7], CMA-ES was only recently revisited as a hybrid between some ES with adaptive step-size and some *Adaptive Encoding* (AE) procedure [4]. AE can be applied to any continuous domain search algorithm, in order to make it independent from any given coordinate system. As a result, some search algorithm that performed rather poorly on non-separable functions can be tremendously boosted (e.g., by a factor up to 3 orders of magnitude for Evolution Strategy with Cauchy distribution [4]).

An iteration of CMA-ES, decomposed into Adaptive Encoding and Evolution Strategy with step-size adaptation, is described in Algorithm 1. In standard Evolution Strategy, λ offspring are sampled (line 1) from a normal distribution with step-size σ and mean m, where m is the centroid of best μ individuals of the previous iteration. The λ offspring are evaluated (line 2 with $\mathbf{B} = \boldsymbol{I}$ the identity matrix). Depending on the choice of the step-size adaptation rule, the step-size is then adapted, either by some rule similar to the one-fifth rule [13, 11] (line 4), or using the Cumulative Step-size Adaptation [6] (line 6).

Algorithm 1 CMA-ES = Adaptive Encoding + ES

```
1: $x_i \leftarrow m + \sigma\mathcal{N}_i(0, \boldsymbol{I})$,  for $i = 1 \ldots \lambda$
2: $f_i \leftarrow f(\mathbf{B}x_i)$,  for $i = 1 \ldots \lambda$
3: **if** Evolution Strategy (ES) with 1/5th success rule **then**
4:     $\sigma \leftarrow \sigma\, \boldsymbol{exp}^{\propto(\frac{\texttt{success rate}}{\texttt{expected success rate}} - 1)}$
5: **if** Cumulative Step-Size Adaptation ES (CSA-ES) **then**
6:     $\sigma \leftarrow \sigma\, \boldsymbol{exp}^{\propto(\frac{\|\texttt{evolution path}\|}{\|\texttt{expected evolution path}\|} - 1)}$
7: $\mathbf{B} \leftarrow$AdaptiveEncoding$(\mathbf{B}x_1, \ldots, \mathbf{B}x_\mu)$
```

CMA-ES differs from standard ES on lines 2 and 7, that describe the use of the Adaptive Encoding procedure. CMA-ES maintains a coordinate system transformation matrix $\boldsymbol{B}$, and though it evaluates the individuals in the original coordinate system of $\mathbb{R}^d$ (line 2), it generates the offspring, using some isotropic normal distribution, in some transformed coordinate system (line 1). The $d \times d$ matrix $\boldsymbol{B}$ is the matrix of the transformation. In Algorithm 1, offspring x_i are represented in this transformed coordinate system, and $\boldsymbol{B}x_i$ are their images in the original coordinate system. Matrix $\boldsymbol{B}$ is iteratively adapted by the AE procedure using information from the most successful μ offspring (line 7).

The CMA update rule for $\boldsymbol{B}$, denoted as AE_{CMA}, derived from the original Covariance Adaptation rule of the (μ, λ)-CMA-ES [7], is detailed in Algorithm 2. The covariance matrix update is similar to some Principal Component Analysis (PCA) of the successful search steps. The goal of PCA is to find an orthogonal transformation to convert the set of possibly correlated variables into a set of uncorrelated variables, called principal components, that are the eigenvectors of the covariance matrix of the data. However, while PCA is usually used to reduce the dimensionality of the data by taking into account only the main principal components (corresponding to the largest eigenvalues), CMA retains all principal components. These components are determined at each iteration by the eigendecomposition of the current covariance matrix $\boldsymbol{C}$ (line 15 of Algorithm 2). The transformation matrix $\boldsymbol{B}$ is the square-root of the covariance matrix $\boldsymbol{C}$ (line 16). An illustration of Principal Components Analysis is shown in Fig. 1.(a), where the principal components are depicted as the dotted lines, such that the largest variance by any projection of the data comes to lie on the first principal component, the second largest variance on the second, and so on. The idea of such a transformation is to make the objective function in the transformed space as similar as possible to the Sphere function, which is known to be simple for analysis and optimization.

Algorithm 2 Adaptive Encoding

```
1: Input: $x_1, \ldots, x_\mu$
2: **if** *Initialize* **then**
3:     $w_i \leftarrow \frac{1}{\mu}$ ; $c_p \leftarrow \frac{1}{\sqrt{d}}$; $c_1 \leftarrow \frac{0.5}{d}$ ; $c_\mu \leftarrow \frac{0.5}{d}$
4:     $\mathbf{p} \leftarrow 0$
5:     $\mathbf{C} \leftarrow \boldsymbol{I}$ ; $\mathbf{B} \leftarrow \boldsymbol{I}$
6:     $m \leftarrow \sum_{i=1}^{\mu} x_i w_i$
7:     return.
8: $m^- \leftarrow m$
9: $m \leftarrow \sum_{i=1}^{\mu} x_i w_i$
10: $z_0 \leftarrow \frac{\sqrt{d}}{\|\mathbf{B}^{-1}(m-m^-)\|}(m - m^-)$
11: $z_i \leftarrow \frac{\sqrt{d}}{\|\mathbf{B}^{-1}(x_i-m^-)\|}(x_i - m^-)$
12: $\mathbf{p} \leftarrow (1 - c_p)\mathbf{p} + \sqrt{c_p(2 - c_p)}z_0$
13: $\mathbf{C}_\mu \leftarrow \sum_{i=1}^{\mu} w_i z_i z_i^T$
14: $\mathbf{C} \leftarrow (1 - c_1 - c_\mu)\mathbf{C} + c_1\mathbf{p}\mathbf{p}^T + c_\mu\mathbf{C}_\mu$
15: $\mathbf{B}^\circ\mathbf{D}\mathbf{D}\mathbf{B}^\circ \leftarrow$ eigendecomposition$(\mathbf{C})$
16: $\mathbf{B} \leftarrow \mathbf{B}^\circ\mathbf{D}$
17: Output: $\mathbf{B}$
```

Fig. 1.(b) illustrates an Adaptive Encoding Update iteration, where only the μ (green/bold) best among λ generated offspring are used to compute a partial covariance matrix $\boldsymbol{C}_\mu$ (line 13), which replaces a fraction c_μ of the current covariance matrix $\boldsymbol{C}$ (line 14). Additionally, the path of the mean of distribution (evolution path $\boldsymbol{p}$) is recorded in order to increase the variance of favorable directions (line 12, and bold arrow in Fig. 1.(b)). A fraction c_1 of the current covariance matrix $\boldsymbol{C}$ is also replaced by the rank-one matrix of eigen direction $\boldsymbol{p}$ (line 14). However, the update in line 14 might become instable if $c_\mu + c_1 > 1$. Hence the parameter setting (line 3) needs to be chosen specifically for the algorithm at hand.

It has been shown in [4] that the AE_{CMA}-update applied to Evolution Strategy with Cauchy distribution improves the performance on non-separable functions by a factor of roughly one thousand. These results and the fact that the Sphere-like transformed space is usually simpler to analyse

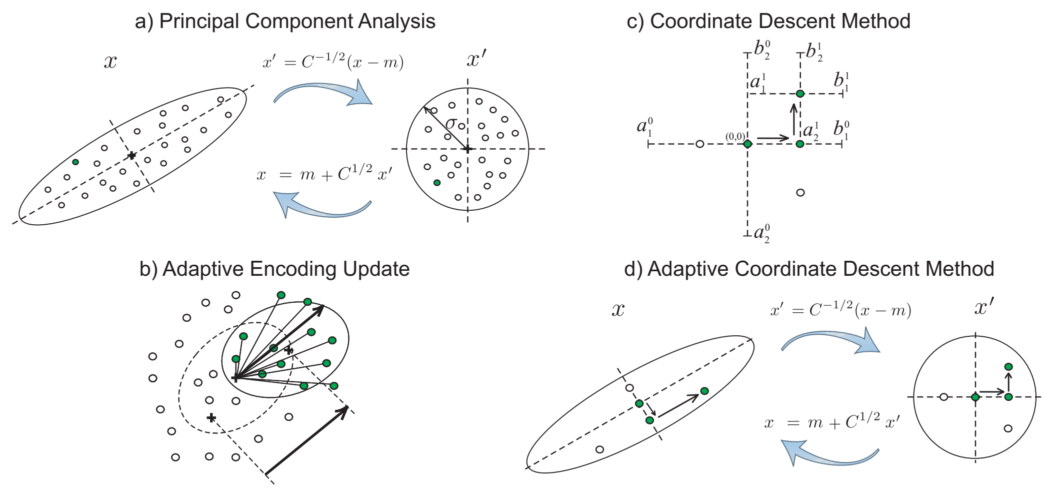


Figure 1: AE$_{\mathrm{CMA}}$-like Adaptive Encoding Update (**b**) mostly based on Principal Component Analysis (**a**) is used to extend some Coordinate Descent method (**c**) to the optimization of non-separable problems (**d**). See text for details.

than the original one, make it reasonable to explicitly exploit this property in search.

2.2 Coordinate Descent by Dichotomy

Coordinate Descent (CD) is probably one of the oldest multidimensional optimization method. It became especially popular in numerical linear algebra under the name of Gauss-Seidel method for solving systems of linear equations. In Evolutionary Computation community, when used for optimization, this method is called Coordinate Strategy [14]. CD is based on the idea that an n-dimensional optimization problem can be decomposed into n one-dimensional sub-problems. Each variable is updated in turn, while all other variables remain fixed, by solving a one-dimension optimization sub-problem using any suitable one-dimension optimization algorithm. Note that CD can be viewed as a special case of Block Coordinate Descent, that partitions the coordinates into N blocks: f is iteratively optimized with respect to one of the coordinate block while other coordinates are fixed [15]. Obviously, it is reasonable to use CD when dealing with unimodal separable problems.

2.2.1 *Adaptive Dichotomy*

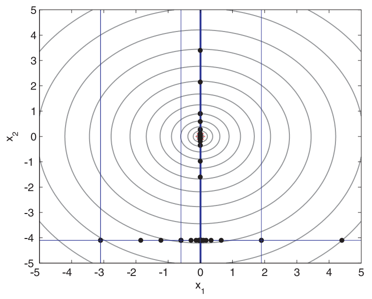
One of the simplest one-dimension optimization algorithm to use is a dichotomy method (inspired by the bisection method to find a zero of a given function). Let us consider an interval $[a, b]$ where the optimum is known to lie, and assume that the value of the objective function f is known at the center $m = \frac{a+b}{2}$ of the interval. Evaluate the two points $X_1 = m - \frac{(b-a)}{4}$ and $X_2 = m + \frac{(b-a)}{4}$, centers of the left and right parts of $[a, b]$. If f is unimodal, only three cases are possible: X_1 is better than m and X_2, X_2 is better than m and X_1, or both X_1 and X_2 are worse than m (if X_1 and X_2 are both better than m, then the problem is multimodal).

If X_1 is better than m and X_2, then the optimum lies in the interval $[a, m)$: replace b with m and m with X_1. Similarly, if X_2 is better than m and X_1, replace a with m and m with X_2. Finally, if X_1 and X_2 are worse than m, then the optimum lies in the interval (X_1, X_2): replace a with X_1 and b with X_2. In all 3 cases, we end up with a new interval $[a, b]$ which contains the optimum, whose length is half that of the original $[a, b]$, and for which we know the value of f at its center.

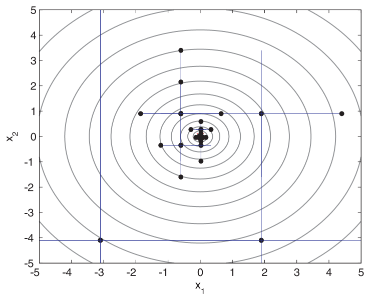
When dealing with multi-dimensional problems, dichotomy steps can be achieved on each coordinate successively: Figure 1.(c) illustrates the 2D-case and displays an example of one dichotomy step in each direction.

Another point of view on the dichotomy method is to consider it as a derandomized $(1 + 2) - ES$ algorithm with step-size adaptation: Assuming the current step-size is σ and current solution is m, the basic step of the dichotomy method described above generates 2 offspring X_1 and X_2 in a deterministic way. The best of m, X_1, and X_2 becomes the next parent, and σ is divided by 2. In the case of one-dimensional unimodal problems, if the initial interval contains the global optimum, this algorithm will find it. Similarly, in the case of multi-dimensional unimodal problems, if the initial rectangle contains the global optimum, the algorithm will find it, either by running the dichotomy method on each coordinate up to a given precision, or by alternating one step of the dichotomy method in each direction in turn. Fig. 2.(a) shows the result of such an optimization of the Sphere function $f(x) = \|x\|^2$ starting from the initial point $X_0 = (-3.1, -4.1)$. Dichotomy proceeds for 20 iterations for the first coordinate and then for 20 iterations for the second coordinate, reaching the target function value 10^{-10} after 80 evaluations. Exactly the same result is obtained by cyclically repeating this procedure over each coordinate in turn, as shown on Fig. 2.(b). The second variant, however, seems to be more robust if the problem is not perfectly separable, exploring a larger region of the search space rather than rapidly reducing one dimension to a single value.

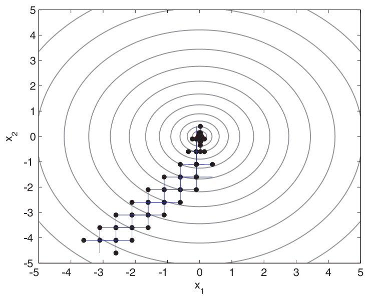
However, if the optimum lies outside the initial interval, or if the interval is somehow transformed after a rotation of the coordinate system (e.g., due to Adaptive Encoding, see Section 2.3), it might be necessary to allow more exploration in case of successful sampling (one offspring was better than the parent m). Such dichotomy method with step-size (interval) adaptation will be called Adaptive Dichotomy (AD), and works as follows: Generate two offspring X_1 and X_2 as



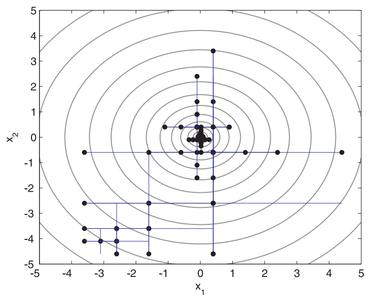
(a) CD k_{succ}=0.5, 80 evaluations.



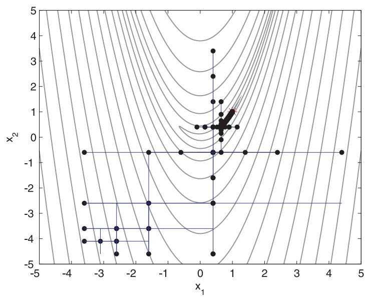
(b) CD k_{succ}=0.5, 80 evaluations.



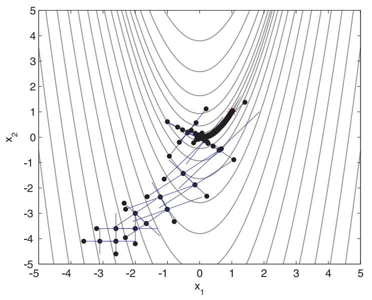
(c) CD k_{succ}=1.0, 117 evaluations.



(d) CD k_{succ}=2.0, 149 evaluations.



(e) CD k_{succ}=2.0, 22231 evaluations.



(f) ACiD k_{succ}=1.2, 325 evaluations.

Figure 2: Coordinate Descent (CD) (**a-e**) and Adaptive Coordinate Descent (ACiD) (**f**) on Sphere (**a-d**) and Rosenbrock (**e,f**) functions. The initial point $\boldsymbol{X}_0$=(-3.1,-4,1) and the target point $\boldsymbol{X}_{target}$ with $f(\boldsymbol{X}_{target}) < 10^{-10}$. While the CD with the dichotomy (**a,b**) performs best on the Sphere function cyclically dividing by two the step size for the corresponding coordinate (depicted as the line), the increasing of the step size by factor k_{succ} in the case of successful sampling leads to a better but still slow convergence on non-separable Rosenbrock function (**e**). The adaptation of the coordinate system allows significantly speed up the search (**f**).

above. If at least one of these two offspring is better than its parent m, then $\sigma \leftarrow \sigma k_{succ}$, otherwise $\sigma \leftarrow \sigma k_{unsucc}$. In the case of standard dichotomy, $k_{succ} = k_{unsucc} = 0.5$, which is suitable for the unimodal separable problems, when initial interval contains the optimum. However, whereas $k_{unsucc} = 0.5$ seems a good choice for all situations, and will be used throughout the end of this paper, $k_{succ} > 0.5$ is mandatory in most cases (e.g., even on the Sphere function, the algorithm will not converge if the initial domain does not contain the optimum). Fig.2.(c) and (d) illustrate runs where the optimum does not lie in the initial rectangle. However, with $k_{succ} = 1.0$ and $k_{succ} = 2.0$ respectively, the Coordinate Descent with Adaptive Dichotomy converges, though at the price of additional functions evaluations.
A more formal description of CD with Adaptive Dichotomy will be given in Section 2.3 (Algorithm 3).

2.2.2 *Convergence Rates*

Before turning to Adaptive Encoding and non-separable functions, let us analyze the convergence rate of CD with Adaptive Dichotomy on the Sphere function, and compare it that of standard Evolution Strategies, whose behavior is well studied in this context.

Linear convergence to the optimal point $\boldsymbol{X}^*$ takes place if there is a constant $c \neq 0$, such that

$$\frac{1}{T_k} \ln \frac{\|\boldsymbol{X}_k - \boldsymbol{X}^*\|}{\|\boldsymbol{X}_0 - \boldsymbol{X}^*\|} \to c, \qquad (1)$$

where $\boldsymbol{X}_0$ is the initial point and $\boldsymbol{X}_k$ the best point found after k iterations for a cost of T_k fitness function evaluations.

The empirical convergence rate on the Sphere function of the proposed CD with $k_{succ} = 0.5$, 1.0 and 2.0, as well as that of two variants of (1+1)-Evolution Strategy, as estimated from the median of 101 independent runs, is shown in Fig. 3. The algorithm denoted as (1+1)-ES corresponds to the (1+1)-Evolution Strategy with the initial step-size $\sigma_0 = 1.8$, while the search interval is $[-3; 3]^d$. The (1+1)-ES opt algorithm is the Evolution Strategy with the scale-invariant step-size: the optimal step-size for ES on Sphere function is proved to be $\sigma = \frac{1.2}{d} \|X - X^*\|$, i.e., proportional to the distance to the optimum.

It is clear that the convergence rate of the CD with (standard) dichotomy ($k_{succ} = 0.5$) is linear with dimension d, and is equal to $-\ln(2)/2d = -0.3465/d$. The rates for CD with Adaptive Dichotomy, with $k_{succ} = 1.0$ and $k_{succ} = 2.0$ are 1.5 and 2.0 times slower, respectively, than with $k_{succ} = 0.5$.

The recently proposed technique of *mirrored sampling* and *sequential selection* for Evolution Strategy [2], can also be used with the CD method proposed here, because the sampled points are symmetric by definition. We hence propose

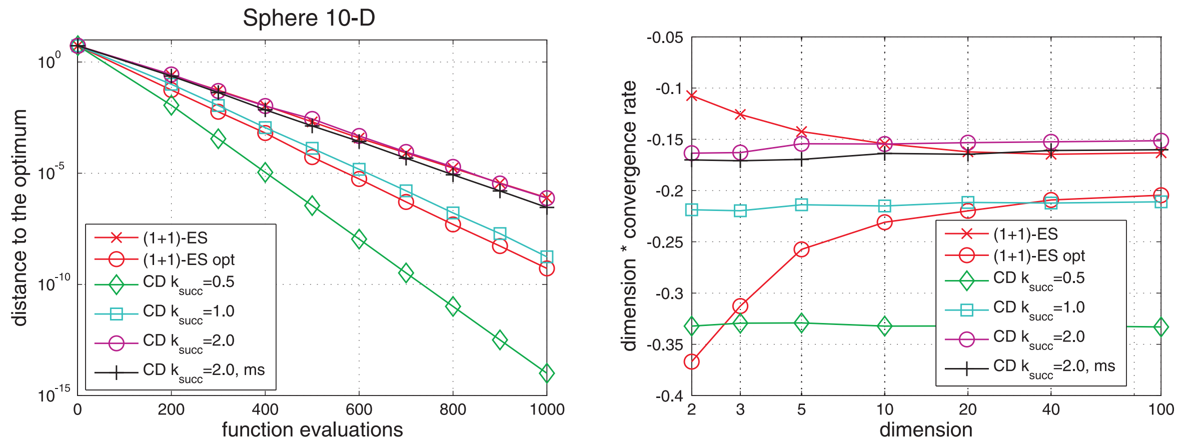


Figure 3: **Left**: Evolution of distance to the optimum versus number of function evaluations for the (1+1)-ES, (1+1)-ES opt, CD k_{succ}=0.5, CD k_{succ}=1.0, CD k_{succ}=2.0 and CD k_{succ}=2.0 ms on $f(x) = \|x\|^2$ in dimension **10**. **Right**: Convergence rate c (the lower the better) multiplied by the dimension d for different algorithms depending on the dimension d. The convergence rates have been estimated for the median of 101 runs.

a modified version of the CD with Adaptive Dichotomy algorithm, called CD k_{succ}=2.0 ms , in which the second offspring is not evaluated if the first one is better than the parent. This does reduce the number of fitness function evaluations, though marginally, as can be seen on Fig. 3 (line with label "ms").

It is important to note that the optimal convergence rate of (1+1)-ES opt is not achievable in practice, because the optimal step-size is unknown for a given black-box function. In the case of CD, parameter k_{succ}, which controls the exploration rate, can be used to implicitly tune the target convergence rate.

A final remark on one-dimensional algorithms: Obviously, any other one-dimensional optimization method could be used instead of the Dichotomy method. The Golden Section method (also called Fibonacci method) is known to have a better convergence rate $c = -\ln\left(\frac{1+\sqrt{5}}{2}\right)/d = -0.4812/d$. However, the Golden Section generates new points with respect to two evaluated points on a line. Therefore, after a change of coordinate (due to Adaptive Encoding, see next Section) will require to recompute the fitness of these rotated point whereas dichotomy only requires the fitness value of the center of the current domain, that is preserved by the change of coordinate.

2.3 Adaptive Coordinate Descent

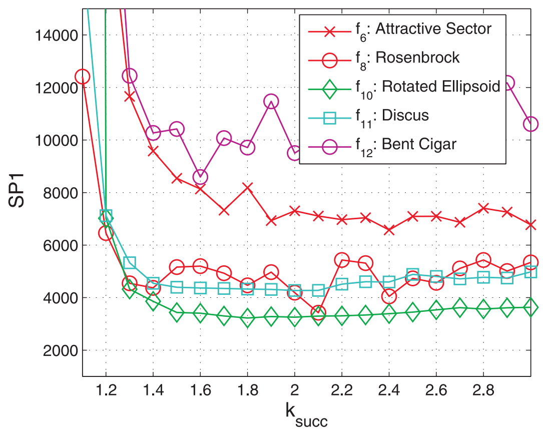
The Adaptive Encoding procedure (Section 2.1) iteratively learns the coordinate systems in which the objective function is "as close to separable as possible". The Coordinate Descent method (Section 2.2) takes advantage of the separability of the problem at hand, iteratively optimizing d independent one-dimensional problems. Combining both approaches leads to propose Adaptive Coordinate Descent (ACiD), which benefits from these two ideas, interleaving CD and AE, learning the same coordinate transform than the original CMA-ES inspired AE, and performing CD steps in the transformed space.

Fig. 1.(d) illustrates how AE acts on the iterations of one-dimensional search steps of Fig. 1.(c). The advantages of ACiD over CD become obvious on non-separable functions. Fig.2.(e) and (f) show sample runs of CD and ACiD respectiveley, on Rosenbrock function in 2-D: ACiD (with $k_{succ} = 2.0$) quickly adapts an appropriate coordinate system and finds the optimum 70 times faster than CD.

Algorithm 3 ACiD

```
1: $m \leftarrow x^{min}_{i:d} + \mathbb{U}_{i:d}(x^{max}_{i:d} - x^{min}_{i:d})$
2: $f_{best} \leftarrow evaluate(m)$
3: $\sigma_{i:d} \leftarrow (x^{max}_{i:d} - x^{min}_{i:d})/4$
4: $\mathbf{B} \leftarrow \boldsymbol{I}$
5: $i_x \leftarrow 0$
6: while NOT Stopping Criterion do
7:     $i_x \leftarrow i_x + 1$ mod. $d$        // Cycling over $[1, d]$
8:     $x'_{1:d} \leftarrow 0$
9:     $x'_{i_x} \leftarrow -\sigma_{i_x}$ ; $x_1 \leftarrow m + \mathbf{B}x'$ ; $f_1 \leftarrow evaluate(x_1)$
10:    $x'_{i_x} \leftarrow +\sigma_{i_x}$ ; $x_2 \leftarrow m + \mathbf{B}x'$ ; $f_2 \leftarrow evaluate(x_2)$
11:    $succ \leftarrow 0$
12:    if $f_1 < f_{best}$ then
13:        $f_{best} \leftarrow f_1$ ; $m \leftarrow x_1$ ; $succ \leftarrow 1$
14:    if $f_2 < f_{best}$ then
15:        $f_{best} \leftarrow f_2$ ; $m \leftarrow x_2$ ; $succ \leftarrow 1$
16:    if $succ = 1$ then
17:        $\sigma_{i_x} \leftarrow k_{succ} \cdot \sigma_{i_x}$
18:    else
19:        $\sigma_{i_x} \leftarrow k_{unsucc} \cdot \sigma_{i_x}$
20:    $x^a_{(2i_x-1)} \leftarrow x_1$ ; $f^a_{(2i_x-1)} \leftarrow f_1$
21:    $x^a_{2i_x} \leftarrow x_2$ ; $f^a_{2i_x} \leftarrow f_2$
22:    if $i_x = d$ then
23:        $x^a \leftarrow \left\{ x^a_{<_{f^a_i}:i} | 1 \le i \le 2d \right\}$
24:        $\mathbf{B} \leftarrow AdaptiveEncoding(x^a_1, \ldots, x^a_\mu)$
```

Algorithm 3 describes the proposed ACiD algorithm. Note that it also describes the non-adaptive CD method by taking $\boldsymbol{B} = \boldsymbol{I}$ and removing the call to AdaptiveEncoding (line 24). Algorithm starts by randomly initializing the current parent m uniformly in the given rectangle domain $\Pi_i[x^{min}_i, x^{max}_i]$, and evaluating it. Initial step-sizes σ_i are set to $\frac{1}{4}$ of the



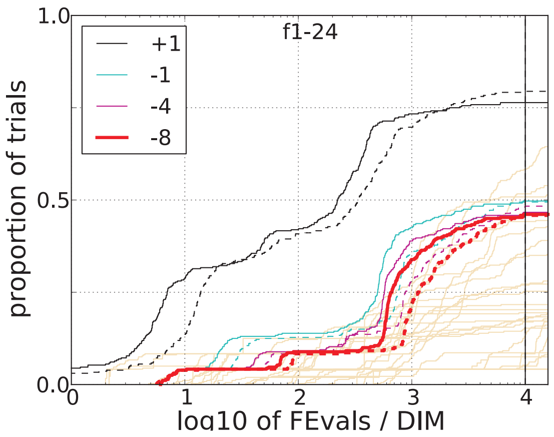


Figure 4: **Left**: The performance of ACiD in **10-D**: BBOB-SP1 (average number of function evaluations to reach target value 10^{-8} divided by success rate) versus step-size multiplier k_{succ}. **Right**: BBOB-like results for noiseless functions in **20-D**: Empirical Cumulative Distribution Function (ECDF), for ACiD (continuous lines) and $(1+1)$-CMA-ES (dashed lines), of the running time (number of function evaluations), normalized by dimension d, needed to reach target precision $f_{\mathrm{opt}} + 10^k$ (for $k = +1, -1, -4, -8$). The vertical black line indicates the maximum number of function evaluations. Light yellow (or gray in b&w) lines in the background show similar ECDFs for target value 10^{-8} of all algorithms benchmarked during BBOB 2009.

corresponding interval length. i_x is the index of the current coordinate, going from 1 to d cyclically (line 7)[1]. The two offspring x_1 and x_2 are generated from m with offset $\pm\sigma_{i_x}$ on coordinate i_x (lines 8-10). If one of them has better fitness value than m (minimization assumed here), m and f_{best} are updated accordingly (line 13 or 15). The i_x step-size σ_{i_x} is then updated multiplicatively depending on the success indicator (line 17 or 19). The coordinates and fitness of both offspring are then stored (lines 20 and 21). At the end of the coordinate loop (i.e., when $i_x = d$, line 23), the Adaptive Encoding procedure is called to update the transformation matrix $\boldsymbol{B}$, using information from the μ best offspring encountered during the d coordinate steps (line 24).

The proposed algorithm is deterministic, therefore the result solution for the noiseless functions only depends on the starting point – and the permutation of variables if any.

3. EXPERIMENTAL VALIDATION

Adaptive Coordinate Descent has been benchmarked on the noiseless Black-Box Optimization Benchmarking (BBOB) testbed [5]. Thanks to the publically available results of many algorithms on the same testbench, and to automatic comparison procedures provided by this framework, ACiD results will be compared to those of the state-of-the-art algorithms: BIPOP-CMA-ES, IPOP-CMA-ES, IPOP-aCMA-ES, (1+1)-CMA-ES and $(1+2^s_m)$-CMA-ES [1].

3.1 Experimental Settings

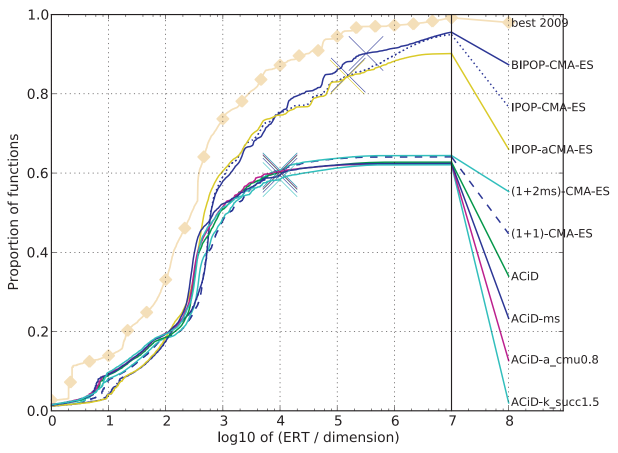
In order to validate ACiD with a robust version, a value of $k_{succ} = 2$ will be used as a baseline (while $k_{unsucc} = 0.5$ as usual). Indeed, while other values for $k_{succ} \in [0.5, 2.0]$ may lead to faster convergence, they also sometimes result in premature convergence on some problems, even on the Sphere function, as demonstrated by preliminary experiments. Also note that the basic rates for the covariance matrix update are set to the simple values $c_1 = \frac{0.5}{d}$ and $c_\mu = \frac{0.5}{d}$ rather than the default ones [4]. However, several other experiments not presented here due to space limitations, demonstrated that slightly different variants of ACiD could improve over the baseline. Hence, a total of 4 variants will be tested in the following: the baseline described above is denoted *ACiD*; *ACiD-k_succ1.5* is the same variant, but with $k_{succ} = 1.5$; *ACiD-ms* uses the value $k_{succ} = 2$, but implements mirrored sampling and sequential selection (Section 2.2.2); finally, *ACiD-a_cmu0.8* uses $c_\mu = \frac{0.5}{d^{0.8}}$ rather than the default value, as some extensive experiments varying all parameters have shown that c_μ is a sensitive parameter indeed, and the value of $\frac{0.5}{d^{0.8}}$ can bring up to 20% improvement on some problems.

Because ACiD can be considered as $(1+2)-ES$ algorithm, a restart procedure is necessary to improve the performance on multi-modal functions. Similarly to the other $(1+1)-ES$ that ACiD will be compared to within BBOB testbench, the algorithm is restarted if the improvements of the best solution is smaller than 10^{-25} during the last $10 + \lfloor 20d^{1.5} \rfloor$ function evaluations. The maximum number of function evaluations is $10^4 d$, and the initial interval is $[-3, 3]^d$. All results are statistics over 15 independent runs.

The performances of all algorithms will be measured using the *expected running time* (ERT), i.e., the expected number of function evaluation to reach a target precision for the first time [5]. The ERT computes to $ERT(f_{target}) = RT_s + \frac{1-p_s}{p_s} RT_{us}$, where the *running times* RT_s and RT_{us} denote the average number of function evaluations for successful and unsuccessful trials respectively, and p_s denotes the fraction of successful trials – a run being successful if it does reach the target precision.

The MatLab source code of ACiD is available online at `http://sites.google.com/site/acdgecco/`.

[1]Random cycles were also tried, but no significant impact on performance was ever observed, so only the cyclic variant is shown here for the sake of simplicity.



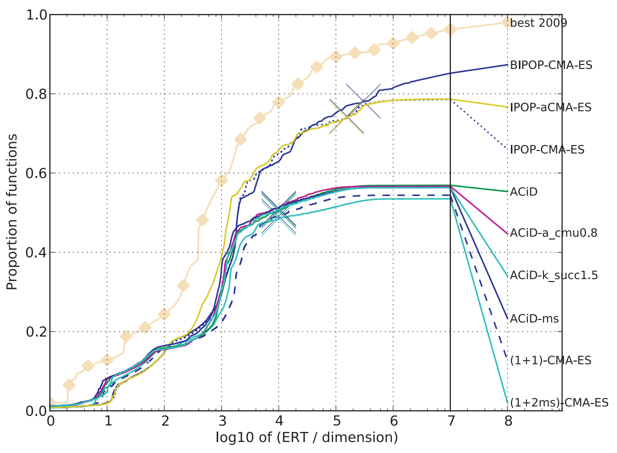


Figure 5: Empirical cumulative distribution of the bootstrapped distribution of ERT vs dimension for 50 targets in $10^{[-8..2]}$ for all functions and subgroups in **10-D** (**Left**) and **40-D** (**Right**). The best ever line corresponds to the best result obtained by at least one algorithm from BBOB 2009 for each of the targets.

3.2 Results and Discussion

The first experiment is concerned with the **sensitivity of parameter** k_{succ}. Fig.4.(Left) shows the performance of ACiD (in terms of BBOB-SP1) depending on k_{succ}, for several problems in 10-D. k_{succ} determines how fast the step-size will increase for a given coordinate if the last step along that coordinate was successful. There is a strong link between k_{succ} and the covariance matrix learning coefficients c_1 and c_μ, since they both determine the comparative impact of the new steps.

The experiments show that ACiD does not converges for $k_{succ} \leq 1$ on non-separable problems, whereas on the Sphere function, ACiD obtains nearly the same results than CD with $0.5 \leq k_{succ} < 1.0$. The reason for this is easy to understand on a small example: if $k_{unsucc} = 0.5$ and $k_{succ} = 1.1$, then in the case of 2 consecutive unsuccessful steps, the step-size σ is divided by 4; so in order to come back to initial step size, 13 consecutive successful steps are needed! The optimal pair k_{succ}, k_{unsucc} of course depends on the problem, and on the other parameters of the ACiD, too. But $k_{succ} = \frac{1}{k_{unsucc}} = 2.0$ seems to be both robust and simple, at least for the given problems.

The BBOB noiseless testbed [5] contains separable, ill-conditioned and multi-modal functions with adequate and weak global structures, with in total 24 functions. Furthermore, standard BBOB outputs include aggregated Empirical Cumulative Distribution Functions (ECDF) that give the proportion of runs that reached a given precision for a given computational effort.

Fig.4.(Right) thus presents the ECDFs of both ACiD and (1+1)-CMA-ES aggregated over the 24 noiseless benchmark problems in 20-D. A closer look at the results (details not shown here) reveals that for 50% of the solved functions, ACiD is about by 40% faster than $(1 + 1)$-CMA-ES. ACiD converges on Attractive Sector function f_6, while $(1 + 1)$-CMA-ES does not. On Rosenbrock f_8 and f_9, Ellipsoid f_2 and f_{10} and Discus f_{11}, some ACiD algorithms are up to 2 times faster than (1+1)-CMA. Furthermore, with a budget of $1000d$ function evaluations, ACiD performs best in 20-D among all algorithms that entered BBOB-2009 competition.

Fig. 5 gives a more general view of similar experiments in 10-D and 40-D – but here experiments with 50 different target values are aggregated. Overall, the functions are not easy to solve. Even the best CMA-ES algorithm, BIPOP-CMA-ES, can solve less than 90% of the problems using maximum number of function evaluations (the large crosses). Furthermore, the elitist algorithms (ACiD, $(1 + 1)$-CMA-ES, $(1 + 2^s_m)$-CMA-ES) are more likely to get stuck in a local optimum than generational algorithms (BIPOP-CMA-ES, IPOP-CMA-ES and IPOP-aCMA-ES), especially in the case where the step-size decreases after each unsuccessful step. The increase of the population size after restart in the case of the premature convergence is the main tool, which leads to a superior performance of the generational CMA-ES algorithms over the single-population algorithms on the multi-modal problems. Indeed, only 10 out of 24 problems are unimodal and this 42% threshold can clearly be seen on the figure. However, the good news is that all ACiD algorithms have at least comparable performance with $(1 + 1)$-CMA-ES and $(1 + 2^s_m)$-CMA-ES: they outperform one another depending on the problem, but the differences are not significant.

The superiority of the ACiD as an absolutely deterministic algorithm was not obvious a priori. These experiments confirm the hypothesis that the efficiency of CMA-ES is mostly due to the Adaptive Encoding procedure, and that the second component of the algorithm (Evolution Strategies, and Gaussian mutations) can be replaced without significant (or even any) loss of performance, at least in the case of the single-individual algorithms. The IPOP-aCMA-ES is only algorithm among the ones presented in this paper that uses all λ offspring in its covariance matrix update. While the best μ points are used to increase the variance along the successful directions, the worst $\lambda - \mu = \mu$ points are used with negative sign to exclude irrelevant directions of search. It is clear that such kind of negative update can be applied to ACiD too. This will be the subject of further work.

4. CONCLUSION AND PERSPECTIVES

The very powerful Covariance Matrix Adaptation part of the state-of-the-art CMA-ES algorithm [7] has been generalized into the so-called Adaptive Encoding (AE) [4] that can be used in conjunction with any optimization algorithm, gradually learning an optimal coordinate system where the objective function at hand is (in the best case, and at least locally) separable. Simple yet powerful algorithms can be used to optimize separable functions, as for instance the Coordinate Descent (CD), that performs up to 2 times faster than the $(1+1)$-ES on the Sphere function. The Adaptive Coordinate Descent algorithm (ACiD), proposed in this paper, uses AE coupled with adaptive CD. ACiD has been shown to be competitive with the CMA-ES algorithms and even up to 2 times faster than $(1+1)$-CMA-ES on several functions of the BBOB test suite. There are, however, a large number of issues that remain open.

Of course, the generational versions of CMA-ES outperform the single-individual ones like $(1+1)$-CMA-ES and ACiD, on most multi-modal problems. But this is essentially due to the restarts with increasing population size. Some further work will be concerned with designing a generational extension of ACiD.

Partial experiments indicate that the off-line tuning of the covariance matrix learning rates c_1 and c_μ can lead to 30-50% speed-up depending on the problem and dimension: more detailed experiments must be made in this direction.

Borrowing ideas from [10], an extension of the Adaptive Encoding procedure to the non-linear case using Kernel Principal Component Analysis (KPCA) [12] is envisioned. Such extension should for instance make it feasible to sample the non-linear distribution along the parabolic shaped optimal valley of the Rosenbrock function.

In ACiD, the evolution path somehow approximates the gradient of the fitness function, and this information is used in the Adaptive Encoding update. However, the line search along the gradient could also be performed explicitly, as in quasi-Newton methods (e.g., BFGS method [3]) and Pattern Search methods (e.g., Hooke and Jeeves method [8]).

We anticipate successful applications of ACiD algorithm to constrained problems. For CMA-ES in large dimensions, resampling the infeasible points does not work, and leads to a rapid decrease of the step-size that further limits the exploration. Within ACiD, the resampling on a line is easy, both in the transformed and in the original spaces.

Another possible extension of ACiD is concerned with surrogate models: the computationally cheap meta-model assisted one-dimensional search becomes favorable even with some budget of 3 to 5 function evaluations, at least for unimodal problems. Furthermore, in order to preserve the invariance properties of the ACiD, comparison-based surrogate models can be used, as advocated in [9].

Finally, the one-dimensional search procedure used in ACiD could be replaced by some k-dimensional search ($k \leq d$). For $k = 2$, the budget is $2k = 4$ function evaluations to find the best of 8 possible states (see Fig.1.(c)). By taking into account all available information, such as the projection of the evolution path on 2-D, we could increase the chances to directly find new best points, for example in the corner. In this case, by simply increasing both step-sizes, the resulting speed-up would increase to 4. We suppose that even such simple strategies, together with sequential selection, can make ACiD significantly faster.

5. ACKNOWLEDGEMENTS

This work is partially funded by FUI of System@tic Paris-Region ICT cluster through contract DGT 117 407 *Complex Systems Design Lab* (CSDL).

6. REFERENCES

[1] A. Auger, S. Finck, N. Hansen, and R. Ros. BBOB 2010: Comparison Tables of All Algorithms on All Noiseless Functions. Technical Report RR-7215, INRIA, 2010.

[2] D. Brockhoff, A. Auger, N. Hansen, D. V. Arnold, and T. Hohm. Mirrored Sampling and Sequential Selection for Evolution Strategies. In R. Schaefer et al., editor, *PPSN XI*, pages 11–20. LNCS 5199, Springer Verlag, 2010.

[3] C. G. Broyden. The Convergence of a Class of Double-rank Minimization Algorithms 1. General Considerations. *IMA J. Appl. Math.*, 6:76–90, 1970.

[4] N. Hansen. Adaptive Encoding: How to Render Search Coordinate System Invariant. In G. Rudolph et al., editor, *PPSN X*, pages 205–214. LNCS 5199, Springer Verlag, 2008.

[5] N. Hansen, S. Finck, R. Ros, and A. Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Technical Report RR-6829, INRIA, 2009-2010.

[6] N. Hansen and A. Ostermeier. Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation. In *International Conference on Evolutionary Computation*, pages 312–317, 1996.

[7] N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evol. Comp.*, 9(2):159–195, 2001.

[8] R. Hooke and T. A. Jeeves. ”Direct Search” Solution of Numerical and Statistical Problems. *J. ACM*, 8:212–229, 1961.

[9] I. Loshchilov, M. Schoenauer, and M. Sebag. Comparison-Based Optimizers Need Comparison-Based Surrogates. In R. Schaefer et al., editor, *PPSN XI*, volume LNCS 6238, pages 364–373. Springer Verlag, 2010.

[10] P. Pošík. Kernel principal components analysis as an efficient crossover operator in real-valued evolutionary algorithms. In *IEEE 4th International Conference on Intelligent Systems Design and Applications 2004*, pages 25–30, 2004.

[11] I. Rechenberg. *Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Hozlboog Verlag, Stuttgart, 1972.

[12] B. Scholkopf, A. Smola, and K.-R. Muller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Technical Report 44, Max Planck Institute for Biological Cybernetics, Tübingen, Germany, 1996.

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

[14] H.-P. P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, Inc., New York, NY, USA, 1993.

[15] P. Tseng. Convergence of Block Coordinate Descent Method for Nondifferentiable Minimization. *J. Optim. Theory Appl.*, 109:475–494, 2001.