Self-Organizing Nets for Optimization

Michele Milano, Petros Koumoutsakos, Jürgen Schmidhuber

M. Milano is with the Graduate Aeronautics Laboratories, California Institute of Technology, Pasadena CA 91125, U.S.A. E-mail: milano@galcit.caltech.edu

P. Koumoutsakos is with the Institute of Computational Science (ICoS), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland. E-mail: petros@inf.ethz.ch

J. Schmidhuber is with IDSIA, Galleria 2 Manno (Lugano), CH-6928, Switzerland. E-mail: juergen@idsia.ch

Abstract

Given some optimization problem and a series of typically expensive trials of solution candidates sampled from a search space, how can we efficiently select the next candidate? We address this fundamental problem by embedding simple optimization strategies in learning algorithms inspired by Kohonen's self-organizing maps and neural gas networks. Our adaptive nets or grids are used to identify and exploit search space regions that maximize the probability of generating points closer to the optima. Net nodes are attracted by candidates that lead to improved evaluations, thus quickly biasing the active data selection process towards promising regions, without loss of ability to escape from local optima. On standard benchmark functions our techniques perform more reliably than the widely used covariance matrix adaptation evolution strategy. The proposed algorithm is also applied to the problem of drag reduction in a flow past an actively controlled circular cylinder, leading to unprecedented drag reduction.

Keywords

Neural Networks, Self Organization, Stochastic Optimization

I. Introduction

A central problem of optimization is to select promising solution candidates without wasting too much time on others. This problem of efficient active data selection is an essential motivation of a wide variety of optimization techniques including genetic algorithms, evolution strategies, reinforcement learning algorithms, tabu search, etc.

To illustrate the problem we focus on stochastic optimization techniques such as evolution strategies (ES) [3], which are standard tools used for the optimization of multivariate, possibly non-continuous functions.

Given an n-dimensional search space, ES uses a population of points (n-dimensional "parents") to generate offspring at random from gaussian distributions with the parents as mean, and a standard deviation (step size) that is continually adapted based on information about past successes. This so-called *mutation* process provides ES with the ability to escape from local minima.

To improve efficiency in terms of convergence speed, however, adaptation of the step size during the optimization process is of crucial importance [3] [4]. The set of all mutation steps yielding improvements is called the *evolution path* of the ES [5]. Clearly, it makes sense to exploit the information embedded in the evolution path to accelerate convergence.

A widely used technique called *covariance matrix adaptation evolution strategy* (CMA-ES) embeds the information about the evolution path in a covariance matrix describing correlations between previous successful mutation steps. Subsequent mutation steps are forced to be uncorrelated with previous ones, thus optimizing step size.

One drawback of this approach is that information is acquired by means of a *local* process which does not improve the global performance properties of the ES [5]. Here we use various adaptive grids or self-organizing maps (SOMs) [1], [2] to trace the evolution path. We define SOM-based mutation operators to generate new offspring; the SOM is continuously trained on successful offspring only, thus reflecting the evolution path in a way that allows for selection of further successful candidates with high probability. We will see that this results in a global rather than local search method, thus overcoming problems of local optimizers such as CMA-ES.

Section II will describe the new optimization method in detail focusing on the modification of a Kohonen SOM (KSOM); section III will overcome certain drawbacks of the KSOM optimizer by introducing a modification of the neural gas (NG) learning algorithm (NGSOM); section IV will briefly review CMA-ES and present comparative results on benchmark functions; section V will apply KSOM and the NGSOM optimizers to a flow control application; section VI will conclude with an outlook.

II. The KSOM-ES algorithm

We start by introducing relevant definitions.

1. $f(.)$ is the scalar valued objective function to be optimized.
2. $\vec{x} \in \mathbf{R^n}$ denotes a parameter vector in which $f(.)$ is evaluated.
3. $\vec{x}_{best}$ is a time-varying, variable parameter denoting the $\vec{x}$ corresponding to the best currently known $f(\vec{x})$.
4. $F_{best} = f(\vec{x}_{best})$; T is a threshold value for F_{best} used to decide if satisfactory convergence has been attained.
5. $\vec{X_i} \in \mathbf{R^n}$, $i \in \{1, \ldots, m\}$ are the m SOM codebook vectors or nodes of the adaptive grid.

The KSOM has n-dimensional connectivity. Its nodes or codebook vectors partition the search space into simplexes; nodes on the search space boundary are fixed. In Fig.1 a

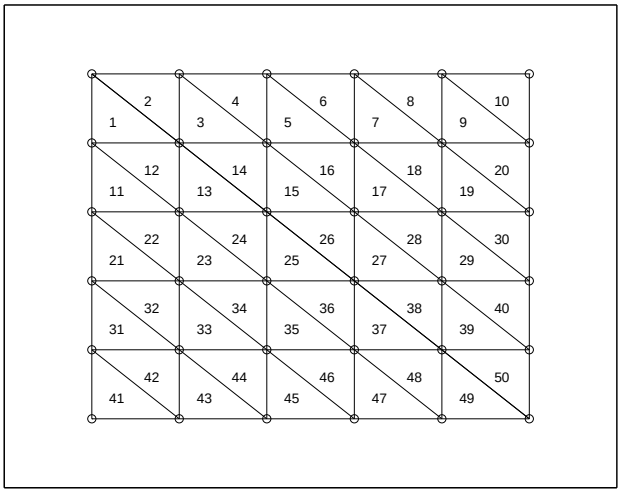

Fig. 1. Subdivision of a 2-dimensional KSOM into simplexes. Circles denote codebook vectors; numbered triangles denote simplexes.

subdivision for $n = 2, m = 36$ is shown.

We are now ready to define a mutation operator $\mathbf{M}$ as follows:

1. Select a simplex at random in the KSOM.
2. Generate a random point $\vec{x}$ uniformly distributed within the simplex.

Given an optimization problem, the KSOM-ES algorithm can be outlined as follows:

1. Initialization:

(a) Distribute the codebook vectors such that they are roughly equally spaced in the search volume.

(b) Use $\mathbf{M}$ to generate a first point $\vec{x}_{best}$, and set $F_{best} = f(\vec{x}_{best})$.

2. Use $\mathbf{M}$ to generate a new $\vec{x}$ and compute $f(\vec{x})$.
3. If $f(\vec{x}) < F_{best}$ then:

(a) $\vec{x}_{best} = \vec{x}$; $F_{best} = f(\vec{x})$

(b) Perform a Kohonen node adaptation step [1] on the KSOM with $\vec{x}_{best}$ as input, but restrict nodes on the boundary to move on the boundary only (in the experiments we actually fix the boundary nodes).

4. If $F_{best} > T$ then go to 2.

The Kohonen training step is defined in the standard way:

$$\vec{X}_i^{new} = \vec{X}_i + \eta \cdot h(i,w) \cdot \left(\vec{x} - \vec{X}_i\right), \quad i \in \{1,\ldots,m\} \tag{1}$$

where w is the index of the codebook vector nearest to $\vec{x}$ (the "winning" vector); $h(i,w)$ is the neighborhood function; η is the learning rate.

For the SOM adaptation we use an n-dimensional lattice connection topology; in example in Fig.1, the horizontal and vertical lines represent the connection topology between two-dimensional SOM neurons. Also, for each neuron w of the network, we define the set of its nearest neighbors $\mathbf{N_w}$ as the set of directly connected neurons. The corresponding neighborhood function is:

$$h(i,w) = \begin{cases} 1, & \mathrm{i} \in \mathbf{N_w} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

The learning parameters are kept constant during learning, unlike the traditional KSOM in which the learning step and the neighborhood size are gradually decreased as the learning proceeds. This is not done here, because the KSOM is used to track the probability distribution of trial points capable to improve the value F_{best}; this distribution is inferred from the set $\{x : F(x) < F_{best}\}$, which varies with F_{best}, thus it is not stationary with the number of iterations performed. For this reason any varying schedule for the KSOM parameters should depend on the current F_{best} and on the objective function $f(\vec{x})$ landscape, so we decided to leave the learning parameters fixed in this preliminary study. Adaptation of the learning parameters during the course of the optimization is the subject of ongoing research.

Without step 3b, the KSOM-ES algorithm degenerates into pure random search (approximated by the initial optimization stages). Since the KSOM is trained only on offspring yielding an improvement of F_{best}, however, allocation of successive offspring through subsequent mutations tends to focus on points in the vicinity of the most successful points encountered so far. This feedback mechanism usually yields a natural evolution path without many unnecessary deviations, as will be illustrated on the standard test functions in the following section. But since the KSOM-defined simplexes always cover the entire search space, there always will be a nonvanishing probability of selecting points other than

those near successful previous points — this prevents the method from getting stuck in local optima, and makes it a *global* method.

Unlike in the traditional KSOM applications we are not so much interested in the nodes but in the space between them. The KSOM adapted by our optimization algorithm can be viewed as approximating the probability distribution of successful trial points. This approximation is piecewise constant inside the KSOM simplexes, because new trial points are uniformly distributed within each simplex.

We also point out that the proposed method is **not** population-based: the SOM network is used to generate trial points, but no function values are computed for the SOM nodes: instead, using the SOM nodes one trial point per iteration is generated, and the function is computed for the generated trial points only. The SOM is used only to track the probability distribution of trial points improving the current best known function value.

III. The neural-gas optimization algorithm (NG-ES)

A major drawback of KSOM-ES is its bad scaling: the number of nodes in the adaptive KSOM grid increases exponentially with the number of function parameters n. To see this, consider the minimum number of nodes needed by KSOM to cover an n-dimensional space; neglecting the boundary nodes, for each dimension at least two nodes are needed to set up the initial grid: this yields a minimum of 2^n nodes needed by the KSOM.

This problem is due to the SOM's n-dimensional connectivity. To overcome the difficulty, we now derive an alternative optimization algorithm based on neural gas (NG) networks [2]. In analogy to the KSOM case we modify standard NG by introducing a mechanism for generating trial points. Premature convergence is avoided through introduction of an additional adaptation term.

To generate trial points we define the following mutation operator M_{NG}, acting on a n-dimensional NG network with m nodes:

1. Select a NG node $\vec{X}^*$ at random;

2. draw a random point $\vec{x}$ from an n-dimensional spherical Gaussian distribution, of mean $\vec{X}^*$ and standard deviation $\sigma = min(||\vec{X}^* - \vec{X}_i||)$, $i = 1, ..., m$, where $||\vec{X}^* - \vec{X}_i||$ is the Euclidean distance.

Here we use the notation introduced for the KSOM. The standard deviation σ is cho-

sen so as to guarantee that the gaussian kernels always overlap. The NG optimization algorithm proceeds as follows:

1. Initialization:
   (a) Distribute the m NG nodes uniformly randomly in the search volume.
   (b) Use M_{NG} to generate a first point $\vec{x}_{best}$, and set $F_{best} = f(\vec{x}_{best})$.
2. Use M_{NG} to generate a new $\vec{x}$ and compute $f(\vec{x})$
3. If $f(\vec{x}) < F_{best}$ then:
   (a) $\vec{x}_{best} = \vec{x}$; $F_{best} = f(\vec{x})$.
   (b) Rank the NG nodes based on $\vec{x}_{best}$, sorting them in increasing order of distance from $\vec{x}_{best}$; store the indices of the sorted nodes in a vector $\vec{i^{rank}}$.
   (c) Update the NG: $\vec{X_i^{new}} = \vec{X}_i + \alpha \cdot h_N(i) \cdot (\vec{x}_{best} - \vec{X}_i), \quad i \in \vec{i^{rank}}$.
4. If $F_{best} > T$ then go to 2.

Here $\vec{i^{rank}}$ is a vector containing the closeness ranking of the NG nodes with respect to the winner node [2], α is the adaptation step and $h_N(i)$ is the NG neighborhood function, defined as:

$$h_N(i) = e^{-\frac{i}{\lambda}} \tag{3}$$

where λ is a parameter determining the size of the adaptation neighborhood, and i is the ith element of $\vec{i^{rank}}$. The number of nodes m is no longer dependent on the dimensionality n of the search space, since the NG network does not require the definition of an explicit connection topology. This fact overcomes the "curse of dimensionality" problem encountered in the definition of the KSOM ES.

Unlike the KSOM optimizer, however, the NG optimizer defined as above might converge prematurely and become a local search method. To see this, consider step 3c: the NG nodes could be attracted towards a region containing many successful points, thereby reducing the average kernel width σ. Depending on the initial distribution of the nodes on the function landscape, this process could get trapped in a very small region, with σ continuing to grow smaller and smaller. To avoid this problem we introduce a "repulsive" term acting on the NG nodes:

$$C_i = \sum_{j=1, j\neq i}^{m} \frac{K}{||\vec{X}_i - \vec{X}_j||^2}(\vec{X}_i - \vec{X}_j) \tag{4}$$

where K is a parameter determining the strength of the interaction. With 4 the updating equation for the NG in step 3b becomes:

$$X_i^{\vec{new}} = \vec{X}_i + \alpha \cdot h_N(i) \cdot (\vec{x}_{best} - \vec{X}_i) + C_i, \quad i \in i^{\vec{rank}}. \tag{5}$$

Setting K to zero recovers the original update rule; when the adaptation process clusters the nodes, the C_i increase with the inverse of the inter-node distances, thus counteracting excessive clustering. This, however, introduces new parameters, i.e., the form of the repulsion term and the parameter K.

IV. Results on test functions and comparison with CMA-ES

Here we compare the performance of our SOM-based optimization algorithms on standard test functions with the covariance matrix adaptation evolution strategy (CMA-ES) [5]. The CMA-ES is a widely used optimization algorithm that approximates an evolution path by exploiting information conveyed by sequences of successful mutations. Its fundamental mutation operator for generating new offspring is:

$$\vec{x}^{new} = \vec{x}^{old} + \delta \cdot \mathbf{B} \cdot \vec{\mathbf{z}} \tag{6}$$

where δ is a global step size; $\vec{z} \sim N(\vec{0}; \vec{I})$ is a random vector drawn from a normal distribution; the columns of the matrix $\mathbf{B}$ are the eigenvectors of the covariance matrix of the distribution of mutation points. The matrix $\mathbf{B}$ may be viewed as a rotation matrix that allows for optimizing the direction in which to generate a new point; all information needed for the calculation of $\mathbf{B}$ is gathered from points generated during the optimization process itself. The step size δ is also adaptive; together with 6, the equations describing the full CMA adaptation process are:

$$\mathbf{C^{new}} = (\mathbf{1} - \mathbf{c_{cov}})\mathbf{C^{old}} + \mathbf{c_{cov}}\mathbf{s}_\delta^{\vec{\mathbf{new}}} \tag{7}$$

$$s_\delta^{\vec{new}} = (1 - c)s_\delta^{\vec{old}} + c_u \mathbf{B} \cdot \vec{\mathbf{z}} \tag{8}$$

$$\delta^{new} = \delta^{old} exp\left(\beta(||s_{\delta}^{\vec{new}}|| - \hat{\chi})\right) \tag{9}$$

where $\mathbf{C}$ is the current covariance matrix, whose eigenvectors are stored in $\mathbf{B}$; $\vec{s_\delta}$ is a vector cumulating the step size variations (initialized by $\vec{0}$ in the first step); $\hat{\chi} = \sqrt{n}E\left(||\vec{z}||\right)$; c determines the accumulation time for $\vec{s_\delta}$, c_{cov} determines the cumulation time for the estimated covariance matrix, and $c_u = \sqrt{c(2-c)}$; the parameter β damps the step size variation between successive generations. We refer the reader to [5], [6] for full details on the CMA algorithm and its parameters. Here we fixed the values for the CMA external parameters to the suggested values: $c = 1/\sqrt{n}$, $\beta = 1/n$, and $c_{cov} = \frac{2}{n^2}$, with n being the number of variables of the objective function.

To begin an experimental analysis of the various optimization algorithms we consider three 2-dimensional test functions [7]:

1. Modified Rosenbrock function:

$$f(x,y) = 74 + 100 \cdot \left(y - x^2\right)^2 + (1-x)^2 - 400 * e^{-\left(\frac{(x+1)^2+(y+1)^2}{0.1}\right)} \tag{10}$$

$$(x,y) \in [-2,2] \times [-2,2]$$

2. Griewangk's function:

$$f(x,y) = 1 + \frac{1}{200}\left(x^2 + y^2\right) - cos(x) \cdot cos\left(\frac{y}{\sqrt{(2)}}\right) \tag{11}$$

$$(x,y) \in [-100,100] \times [-100,100]$$

3. Rastrigin's function:

$$f(x,y) = 20 + \left(x^2 - 10 \cdot cos(2\pi x)\right) + \left(y^2 - 10 \cdot cos(2\pi y)\right) \tag{12}$$

$$(x,y) \in [-5.12,5.12] \times [-5.12,5.12]$$

The global minimum of test functions 2 and 3 is $(0,0)$; function 1 is classical Rosenbrock with minimum in $(1,1)$ plus a gaussian bump in $(-1,-1)$. This modification causes a local minimum in $(1,1)$ and a global minimum in $(-1,-1)$, which makes function 1 difficult to optimize, because the local minimum basin is larger than the global minimum basin. Therefore an algorithm exploiting only local information is very likely to be driven into the local minimum.

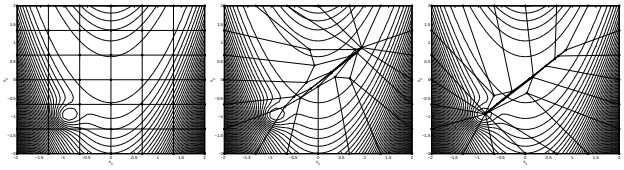

Fig. 2. Evolving KSOM superimposed on contour plots of function 1. From left to right: situation after initialization, after 60 iterations, and after 100 iterations. Only neighborhood connections are shown.

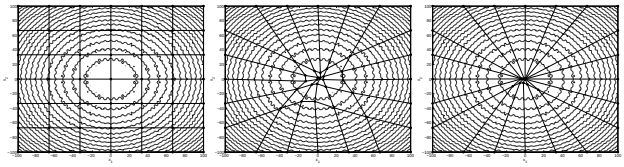

Fig. 3. Evolving KSOM superimposed on contour plots of function 2. From left to right: situation after initialization, after 300 iterations and after 700 iterations.

In all tests we used two KSOMs, with grids of 5×5 and 7×7 codebook vectors, respectively, and two NGSOMs, with 10 and 20 nodes respectively. The learning parameter for the KSOM was always fixed to $\eta = 0.2$. For the NG the learning parameter $\alpha = 0.2$, $\lambda = N/3$ with N being the number of nodes in the NG, and the repulsion parameter $K = 0.001$. The CMA algorithm used for comparison is a $(1, 10)$ strategy.

Data points passed to all the SOM adjusters were normalized between $[-1, 1]$, thus making the learning parameter choices robust with respect to changes in the search domain size.

In our study the SOM parameters were kept constant during the optimization process, in contrast to what is usually done by the more sophisticated standard training algorithms [1], [2]. As explained before, the reason is that the distribution of optimal trial points is not stationary during the optimization process, but depends on the environment embodied by the shape of the function being optimized. Improper annealing of the SOM parameters would jeopardize the network's ability to track this nonstationary distribution.

Figures 2-4 trace the evolution of the grid used by 7×7 KSOM-ES on the three test functions considered. We observe that the KSOM codebook vectors quickly focus the search on areas that contain the most promising points.

Figures 5 and 6 show the evolution of the NGSOM optimizer on the test function 1, by

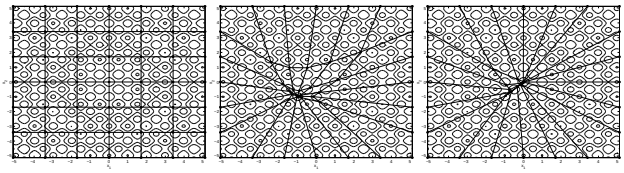

Fig. 4. Evolving KSOM superimposed on contour plots of function 3. From left to right: situation after initialization, after 100 iterations and after 200 iterations.

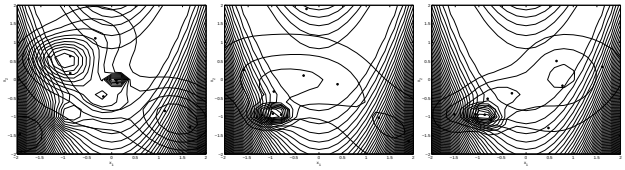

Fig. 5. Evolving NGSOM superimposed on contour plots of function 1. From left to right: situation after initialization, after 50 iterations and after 100 iterations.

plotting the probability density of trial points superimposed to the contours of the test function (figure 5), and a mesh plot of the probability density in the search volume (figure 6). The NG-ES efficiently selects promising regions during the optimization even though the nodes do not form too compact clusters, due to the repulsion term.

To perform a fair comparison with CMA-ES we used the same convergence criterion for all the methods. The thresholds for satisfactory convergence were fixed to 40 for function 1, and to 10^{-3} for functions 2 and 3. There was a maximum of 5000 function evaluations per run.

To obtain statistically significant results we performed a total of 10000 optimization runs per test function and method, each run with a different initial point for CMA-ES (unnecessary for the SOM-ES optimizers).

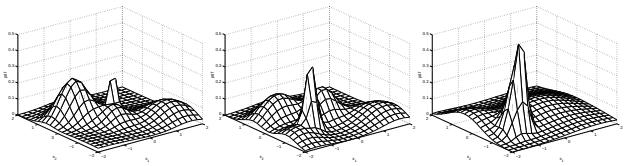

Fig. 6. Evolving NGSOM probability density function, when optimizing function 1. From left to right: situation after initialization, after 50 iterations and after 100 iterations.

The results in table I compare the average number of function evaluations to convergence of the algorithms tested, and the average percent converged runs over 100 optimizations, also reporting standard deviations for both quantities.

In these tests the SOM-ES optimizers outperform CMA-ES in terms of convergence frequency; on the other hand, when CMA-ES happens to be successful the average number of function evaluations required to achieve convergence is smaller; this is not surprising as CMA-ES is a local method devised for optimal exploitation of local information.

The advantages of the SOM-ES optimizers are most evident in case of the most difficult function 1 whose difficulty stems from the additional gaussian bump containing the global minimum: the local minimum basin is much larger than the global minimum basin, and the global minimum is not centered in the search domain (a centered global optimum with large basin facilitates the task of many optimization algorithms [8]).

TABLE I

PERFORMANCE COMPARISON: KSOM-ES AND NG-ES VS CMA-ES ON 3 TEST FUNCTIONS.

	Test 1		Test 2		Test 3	
Method	Av. Evals.	% Succ.	Av. Evals.	% Succ.	Av. Evals.	% Succ.
5×5 SOM-ES	1600 ± 200	$(70 \pm 8)\%$	130 ± 40	$(90 \pm 7)\%$	180 ± 50	$(90 \pm 7)\%$
7×7 SOM-ES	750 ± 90	$(90 \pm 5)\%$	100 ± 40	$(90 \pm 8)\%$	200 ± 50	$(90 \pm 8)\%$
NG-ES (m=10)	1700 ± 200	$(90 \pm 7)\%$	180 ± 50	$(90 \pm 10)\%$	210 ± 50	$(90 \pm 8)\%$
NG-ES (m=20)	780 ± 80	$(90 \pm 9)\%$	150 ± 40	$(90 \pm 8)\%$	180 ± 40	$(90 \pm 7)\%$
CMA-ES	70 ± 40	$(30 \pm 10)\%$	210 ± 50	$(40 \pm 10)\%$	100 ± 40	$(60 \pm 10)\%$

Of course, the somewhat artificial time limit is the only reason why the SOM optimizers did not always achieve 100% success — they are global optimizers; thus they will eventually always find the global optimum.

Comparing the performance of the SOM optimizers, within the time limit the NG-ES finds the global optima of the test functions as frequently as the KSOM-ES, but it requires a larger number of function evaluations to do so. This is probably due to the different mechanism used by each algorithm to encourage exploration: fixed boundary nodes for

the KSOM appear to be more efficient than repulsion terms for the NG in this respect.

However, due to the practical scaling problems detailed before, KSOM-ES optimizers are preferable for problems with few dimensions, while NG-ES becomes more convenient for higher dimensions.

Does a global method such as NG-ES lose much when applied to a unimodal problem where CMA-ES should excel? We tested this on the 10-dimensional Rosenbrock function:

$$f(\mathbf{x}) = \sum_{\mathbf{i=1}}^{\mathbf{9}} \mathbf{100}\left(\mathbf{x_{i+1}} - \mathbf{x_i^2}\right)^{\mathbf{2}} + (\mathbf{1} - \mathbf{x_i})^{\mathbf{2}} \tag{13}$$
$$x_i \in [-2, 2], \quad i = 1, ..., 10.$$

This test function attains its minimum value of zero in $x_i = 1, \quad i = 1, ..., 10$. Since it is unimodal, it is easily optimizable by local search methods such as CMA-ES.

We performed 1000 optimization runs on this function, using a $(1, 10)$ CMA-ES and a NG-ES with $N = 100$ nodes, $\alpha = 0.1$, $\lambda = N/3$, and $K = 10^{-5}$. The average number of function evaluations to convergence is 4800 evaluations for CMA-ES, and 7500 evaluations for NG-ES. As expected, CMA-ES does perform a bit better than NG-ES, because CMA-ES focuses on local information to converge as quickly as possible, while NG-ES requires additional overhead for exploration, avoiding excessive clustering of its neurons. In the case of unimodal functions this is a waste of time, nonetheless NG-ES performs only slightly worse than CMA-ES.

This little test demonstrates that the exploration capability which makes our method global rather than local does not necessarily greatly affect its convergence speed.

V. Application to Fluid Dynamics

We apply the method to a challenging application, namely the problem of drag reduction in flow past an actively controlled circular cylinder at moderate Reynolds numbers.

The flow past a circular cylinder is a well established prototypical configuration of bluff body flows [12]. Here we use the KSOM-ES algorithm in conjunction with computational fluid dynamics (CFD) to achieve drag reduction by active flow control through well-studied rotary oscillations [9], [10] of the cylinder surface.

Governing equations are the Navier-Stokes equations in the velocity-pressure formulation [13]:

$$\frac{d\mathbf{v}}{dt} + (\mathbf{v}\cdot\nabla)\mathbf{v} = -\frac{\mathbf{1}}{\rho}\nabla\mathbf{P} \; + \; \nu\,\nabla^{\mathbf{2}}\mathbf{v} \tag{14}$$

$$\nabla\cdot\mathbf{v} = \mathbf{0} \tag{15}$$

where $\mathbf{v}$ is the velocity vector, P, ρ is the pressure and density of the flow and ν denotes the kinematic viscosity. The boundary conditions are defined as:

$$\mathbf{v}(\mathbf{x},\mathbf{t}) = \mathbf{V_{ext}} \qquad \text{on the cylinder surface} \tag{16}$$

$$\mathbf{v}(\mathbf{x},\mathbf{t}) \; = \; \mathbf{U}_\infty\,\mathbf{e_x} \qquad \text{as } |\mathbf{x}| \to \infty \tag{17}$$

where $\mathbf{V_{ext}}$ is the surface velocity induced by the actuators, and $\mathbf{e_x}$ is the unit vector in the stream-wise direction. U_∞ denotes the free-stream velocity. The Reynolds and Strouhal numbers of the flow (Re and St respectively) are defined as:

$$Re = \frac{U_\infty\,D}{\nu} \qquad\qquad St \; = \; \frac{f\,D}{U_\infty}$$

where $D \; = \; 2\,R$ is the diameter of the cylinder and f the shedding frequency of the flow. We also define a normalized time, by scaling time as follows:

$$t^* = \frac{t}{t_a} = \frac{t}{(D/U_\infty)}$$

where t_a is the time taken by a fluid particle in the free stream to be advected past the cylinder. The definition of such normalized quantities allows a ready generalization of the results to differently scaled geometries. We conduct two-dimensional simulations for Reynolds number $Re = 500$ (Re indicates the degree of flow turbulence), considering two-dimensional incompressible viscous flow past a circular cylinder.

We focus on the traditional and central problem of reducing the drag, represented by the drag coefficient C_D calculated from quantities measured on the cylinder surface as:

$$C_D \; = \; \frac{2}{\rho U_\infty^2 D}\int_{cyl}(p\mathbf{n}_{\mathrm{x}} - \tau_{ix}\mathbf{n}_{\mathrm{i}})dl \tag{18}$$

where p is the pressure and τ_{ix} the viscous stress tensor on the surface of the cylinder and $\mathbf{n}_{\mathrm{i}}, i = x, y$ denotes the components of the unit normal to the cylinder surface [13]. The rotational oscillations imposed to the cylinder are expressed by the following equation:

$$\omega(t^*) = a_1 sin(\omega_0 t^*) + a_2 sin(2\omega_0 t^* + \phi) \tag{19}$$

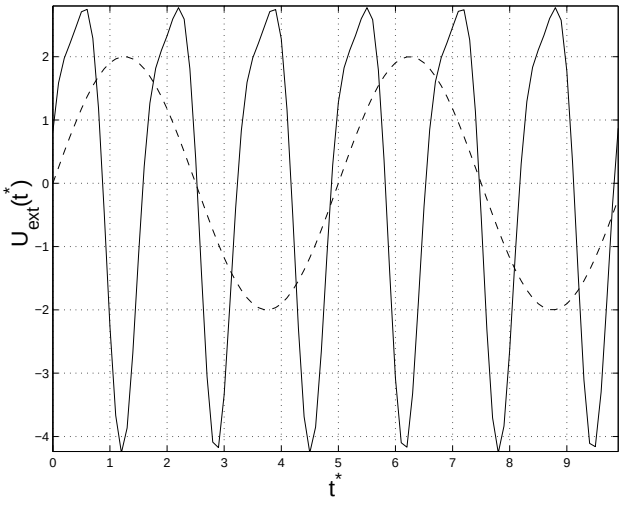

Fig. 7. Optimized rotary oscillations: angular velocity as a function of time, for the hand-tuned solution (dashed line), and for the KSOM-ES optimized solution (continuous line).

where ω is the instantaneous rotational frequency, $\{a_1, a_2, \omega_0, \phi\}$ are the parameters to be optimized in order to minimize the drag. These are the first 2 harmonics of a Fourier series representing the periodic excitation searched.

We optimized the following function:

$$J(a_1, a_2, \omega_0, \phi) = \int_0^T C_D(a_1, a_2, \omega_0, \phi) \mathrm{dt}^* \tag{20}$$

where T is the time horizon considered, that we fixed at four times the uncontrolled shedding frequency.

The search space limits were $\{[0, 2], [0, 2], [0, 20], [-\pi, \pi]\}$ for the parameters $\{a_1, a_2, \omega_0, \phi\}$, respectively. We used a 4-dimensional KSOM with 5 nodes per dimension, resulting in $5^4 = 625$ nodes. The first and last node of each dimension were kept fixed at the search space boundaries, like in the 2D experiments above.

After 200 iterations the KSOM-ES converged to the solution reported in Fig.7.

An hand-tuned solution was obtained by performing purely sinusoidal rotary oscillations at a frequency of two times the shedding frequency. In this way each shedding period produces two counterrotating vortices, which tend to cancel each other thereby reducing the drag.

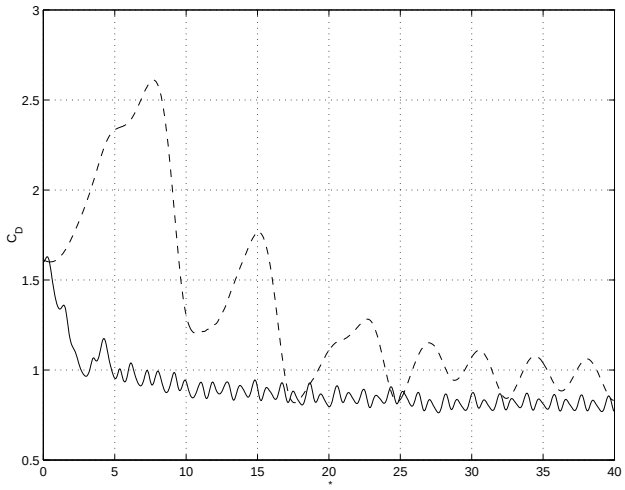

Fig. 8. Dashed line: drag for best hand-tuned input; cont. line: drag for SOM-ES optimized input.

For this problem, however, there is no reasonable "starting guess" for the possible solution in the case of a double harmonic oscillation. Therefore it would be impractical to apply the CMA algorithm, or any other algorithm requiring a reasonable starting point. In this case an appropriate strategy to find the optimum for CMA would be to perform several different optimization runs starting from different initial points, and then pick the best solution found. This is unnecessary for the SOM optimizer, because it starts the search process with a uniform random distribution, automatically focusing on interesting points as the search proceeds. An alternative approach would be to take a point close to the optimal point found by SOM-ES as a suitable guess for the CMA algorithm, but then the comparison of CMA with SOM-ES would be unfair. For these reasons, the CMA algorithm was not applied to this problem.

Fig.8 plots the cylinder drag, comparing it to the best previous solution obtained by hand-tuning one harmonic only. It can be seen that the solution found by the SOM-ES has a faster transient time and narrower oscillations.

The difference between the two drag curves in the transient phase depends on the fact that the initial phase of the shedding may actually increase a vortex that was shedding from the uncontrolled body surface just before control is initiated. This is indeed what

is observed in the case of hand-tuned oscillation. In order to eliminate this effect, either control must be started at a proper initial time, or an additional phase term can be introduced in the oscillations: the initial phase ϕ in equation 19 may be interpreted as a degree of freedom used up by taking this initial phase into account. Furthermore, due to the form chosen for the fitness function, the area under C_D^2 is minimized: this choice has the obvious side effect of minimizing the transient time too.

VI. Conclusions

We introduced a simple yet new approach to active data selection, based on self-organizing maps trained to reflect relevant structure of an error or fitness landscape revealed by successful candidates encountered during an optimization process.

Our optimizers based on Kohonen self-organizing maps and neural gas networks do not require ad-hoc initialization. Although the methods tend to focus computational resources on the neighborhood of previously observed successful candidates, they never lose the ability to discover global optima. On standard test functions they consistently outperform the widely used CMA-ES in terms of reliability, without necessity for parameter tuning.

Our adaptive grid-based optimizers also successfully solved a prototypical problem of flow control, i.e., drag reduction through rotary oscillations of a cylinder.

In our present implementations the training parameters do not change over time. In ongoing work we are experimenting with adaptation rules for those parameters, and we intend to apply SOM-ES optimizers to additional complex optimization problems.

References

[1] T. Kohonen, *Self-Organizing maps*, Springer Verlag 1995

[2] Martinetz, T., Schulten, K. "A 'Neural-Gas' Network learns Topologies", in: Kohonen et al. (Eds), Artificial Neural Networks, Elsevier, North-Holland, pp.397-402

[3] H. P. Schwefel, *Evolution and Optimum Seeking*,Wiley 1995

[4] T. Bäck, U. Hammel, H. P. Schwefel, "Evolutionary Computation. Comments on the History and Current State", IEEE Trans. on Evolutionary Computation, vol. 1, n. 1, 1997, pp. 3-17

[5] N. Hansen, A. Ostermeier, "Adapting Arbitrary Normal Mutation Distributions in Evolution Strategies: The Covariance Matrix Adaptation", IEEE Intern. Conf. on Evolutionary Computation (ICEC) Proceedings, 1996, pp. 312-317

[6] N. Hansen, A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolutionary Strategies", Evolutionary Computation, vol. 9, n. 2, 2001, pp. 159-195

[7] M. Milano, J.Schmidhuber, P.Koumoutsakos, "Active Learning with Adaptive Grids",International Conference on Artificial Neural Networks (ICANN), Vienna, Austria, August 2001

[8] D. Whitley, K. Mathias, S. Rana, J. Dzubera, "Building Better Test Functions", Proc. of the 6th Int. Conf. on GAs, Morgan Kaufmann, 1995, pp. 239-246

[9] P.T. Tokumaru and P.E. Dimotakis, "Rotary oscillation control of a cylinder wake",J. Fluid Mech., v. 224, 1991,p. 77

[10] Y.M. Chen and Y.R. Ou and A.J. Pearlstein, "Development of the wake behind a circular cylinder impulsively started into rotatory and rectilinear motion: Intermediate rotation rates",J. Fluid Mech., v. 253, 1993, p. 449

[11] R. Mittal, "Large-eddy simulation of flow past a circular cylinder", Center for Turbulence Research annual research briefs, Stanford, 1995, p. 107

[12] A. Roshko, "On the wake and drag of bluff bodies",J. Aerosp. Sci., 1955, v. 22, p. 124

[13] R. L. Panton, *Incompressible flow*, Wiley, 1996, p. 387