An Evolutionary Algorithm for Global Optimization Based on Self-Organizing Maps

Sami Barmada, Marco Raugi, Mauro Tucci

Dept. of Energy, Systems, Territory, and Construction Engineering

University of Pisa

Largo LucioLazzarino, 56122, Pisa, Italy

fax: +39 0502217333

sami.barmada@ing.unipi.it phone: +39 0502217312

marco.raugi@ing.unipi.it phone: +39 0502217325

mauro.tucci@ing.unipi.it phone: +39 0502217348 (corresponding)

Acknowledgement ds

This work was supported by the Italian Ministry of University (MIUR) under a Program for the Development of Research of National Interest (PRIN), grant number 2009TCLKNF.

An Evolutionary Algorithm for Global Optimization Based on Self-Organizing Maps

In this work a new population based algorithm for real-parameter global optimization is presented, which is denoted as self-organizing centroids optimization SOC-opt. The proposed method uses a stochastic approach which is based on the sequential learning paradigm for self-organizing maps (SOM). A modified version of the SOM is proposed where each cell contains an individual, which performs a search for a locally optimal solution and it is affected by the search for a global optimum. The movement of the individuals in the search space is based on a discrete-time dynamical filter, and various choices of this filter are possible to obtain different dynamics of the centroids. In this way a general framework is defined where well known algorithms represent a particular case. The proposed algorithm is validated through a set of problems, which include non-separable problems, and compared with state of the art algorithms for global optimization.

Keywords: self-organizing maps; evolutionary algorithms; global optimization; particle-swarm optimization.

Introduction

A large number of applications make use of global optimization algorithms. Population based stochastic methods allow to carry out difficult search and optimization problems, which often arise in complex applications. Evolutionary algorithms (EAs) are a family of such methods that try to replicate the biological processes present in the natural evolution defining a number of heuristics that exploit the relation between biological evolution and optimization, Back (1996). Evolutionary algorithms and swarm intelligence mainly constitute the field of nature-inspired optimization algorithms.

Among EAs, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) algorithms have shown good results on real-parameters problems, in particular on multi- modal functions. A number of versions of CMA-ES with restarts have been proposed to handle

multi-modal functions: NBIPOPaCMA-ES, Loshchilov (2013), showed the best results together with iCMAES-ILS, Liao (2013), on the continuous optimization benchmark at CEC 2013. In addition, other CMA-ES versions were ranked first on the black-box optimization benchmark (BBOB) in 2009 and 2010 and on the continuous optimization benchmark at CEC 2005, Auger (2005).

In this work a new real-parameter optimization algorithm (based on the work introduced by Barmada et al. (2012)) is presented. The proposed model defines a general framework for defining new algorithms, and a particular configuration of the proposed framework has strong similarities with the classic formulation of the PSO, particle swarm optimization, algorithm. Numerical results are reported showing that the performance of the proposed model can be compared to the performance of the best algorithms of the 2013 CEC competition for real parameter optimization.

The problem is the minimization of an objective function, $F(\mathbf{x})$, $\left\{F:\Omega\subseteq\Re^n\rightarrow\Re\right\}$, $\mathbf{x}\in\Omega$, where Ω is the domain of the search. Some desired requirements of optimization algorithms are the following:

- Simple and straightforward to implement.
- Good performances in comparison with several others in a range of problems.
- The number of control parameters should be low.

The proposed algorithms is based on the paradigm of the sequential learning of self-organizing maps, SOM, Kohonen (2001). The SOM is a popular neural network for unsupervised learning, and it is used in a wide number of applications from clustering to data visualization. The SOM performs a vector quantization of the input distribution, and in general it has a strong explorative power. In the case of the topology preserving SOM,

the final centroids tend to be disposed in a predefined topological order. It has been shown that the topology-preserving feature of the SOM accelerates vector quantization with respect to non topology-preserving methods, de Bodt et al. (2004),

A modified SOM that uses a dynamical filter to implement the movement of the centroids during the learning was proposed by Tucci (2009, 2010, 2011). A central feature is that each centroid moves towards a personal target, while in the classic SOM all centroids move towards the same target point.

In this work a search strategy for optimization that tries to use the explorative power of the modified SOM proposed by Tucci (2011) is presented. In the proposed strategy, the centroid vectors of the SOM represent a population of individuals that undergo local perturbations, and move in the search space as well, in order to perform two tasks simultaneously, a local search and a global search. This is accomplished by moving the centroids towards the combination of two target vectors, a local target, and a global target. The target vectors are obtained as local perturbations of the centroids. The movement of centroids towards the targets takes into account the neighborhood interactions, and it is obtained using a discrete-time dynamical filter. The dynamical filter makes use of the centroid past positions, and its choice is flexible and defined a priori by the user. The use of the dynamical filter is a key factor of the proposed algorithms as it allows defining different dynamics of the centroids. For example using a predictive filter the movement of the centroids exhibits a more active behavior than using a smoothing filter, which causes a more inertial movement. In the following it is shown that these different behaviors give different benefits depending on the characteristics of the optimization problem.

The idea to modify the SOM algorithm to define optimization algorithms can be found in literature, for example Grosan (2012) proposes a hybrid approach that uses

self-organizing neurons and evolutionary algorithms, where the focus is to optimize the SOM neuron weights using EAs. The method proposed in this work is different as each neuron represents an individual of the population for optimization.

In order to analyze the performance of the proposed algorithm, the set of problems proposed for CEC competition on real-parameter optimization problems in 2013 is used. This benchmark is composed by 28 minimization problems and it is widely used in the literature. The results show that the proposed scheme, which is denote as SOC-opt (Self Organizing Centroids-optimization) is competitive with the most efficient optimization algorithms. Advantages and disadvantages are outlined in the following sections.

Review of the modified SOM for supervised learning

In this section we first introduce the classic SOM neural network and then the variant proposed by Tucci (2011), denoted as learning filter SOM. This model is a neural network for unsupervised data analysis, while the next section introduces the new optimization algorithm which is based on the learning filter SOM.

The original SOM consists in a number P of cells arranged in a two dimensional grid with a fixed topology, typically hexagonal, where a distance metric is defined between cells $d(c,i)$. Each cell is associated to a model vector (also called centroid) $\mathbf{q}^i \in \Re^n$ lying in the same space of the input pattern, denoted by the vector $\mathbf{r}$. Once the input vector is presented to the network, the winner node is selected by the following function

$$c = \arg\min_i \left\{ \left\| \mathbf{r}(t) - \mathbf{q}^i(t) \right\| \right\}. \quad (1)$$

Consequently the model vector is updated according to the rule

$$\mathbf{q}^i(t+1) = \mathbf{q}^i(t) + h_{ci}(t)\left(\mathbf{r}(t) - \mathbf{q}^i(t)\right), \quad i = 1 \ldots P, \tag{2}$$

where the discrete time $t = 0,1 \ldots T_{MAX}$ represents the generation step and

$$h_{ci}(t) = \mu(t) e^{-d(c,i)^2/(2\sigma^2(t))} \tag{3}$$

in which $\mu(t)$ and $\sigma(t)$ respectively are the learning factor and the neighborhood functions, both decreasing with time. Heuristically speaking, at larger times nodes far from the winner one are less sensitive to the new inputs; at the same time, at larger times the inertia of the model vectors is higher.

A modification of the SOM based on the implementation of a discrete time dynamical filter has been proposed by Tucci (2009, 2010, 2011). The new approach introduces the concept of memory to be associated to each cell. Each cell of index $i = 1 \ldots P$ contains two sets of N vectors, denoted as memory vectors, $\mathbf{R}^i = \left[\mathbf{r}_1^i, \ldots \mathbf{r}_N^i\right]$ and $\mathbf{Q}^i = \left[\mathbf{q}_1^i, \ldots \mathbf{q}_N^i\right]$, where $\mathbf{q}_j^i \in \Re^n$, $\mathbf{r}_j^i \in \Re^n$. Vectors $\mathbf{q}_k^i$ are related to the last N values of the model vector, while vectors $\mathbf{r}_k^i$ keep trace of the last N values of the input that were able to update the cell state.

For each cell the centroid vector is calculated at each time step by the following combination of the memory vectors:

$$\mathbf{q}^i(t) = \sum_{j=1}^{N} b_j \mathbf{r}_j^i(t) + a_j \mathbf{q}_j^i(t) \tag{4}$$

which can be expressed using matrices as

$$\mathbf{q}^i(t) = \left[\mathbf{R}^i(t) \vdots \mathbf{Q}^i(t)\right] \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix} = \mathbf{M}^i(t) \cdot \mathbf{g} \tag{5}$$

where $\mathbf{g} = \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix}$, and $\mathbf{b} = [b_1, \ldots, b_N]^T$, $\mathbf{a} = [a_1, \ldots, a_N]^T$ are the vectors containing the coefficients b_j, a_j.

When a new input sample $\mathbf{r}(t)$ is presented to the map during training, the winner cell is selected by (1), then the memory vectors of the cells are updated as follows:

$$\mathbf{R}_+^i(t) = \left[\mathbf{r}(t), \mathbf{r}_1^i(t), \ldots \mathbf{r}_{N-1}^i(t)\right], \tag{6a}$$

$$\mathbf{Q}_+^i(t) = \left[\mathbf{q}(t), \mathbf{q}_1^i(t), \ldots \mathbf{q}_{N-1}^i(t)\right], \tag{6b}$$

$$\mathbf{R}^i(t+1) = \mathbf{R}^i(t) + h_{ci}(t)\left[\mathbf{R}_+^i(t) - \mathbf{R}^i(t)\right], \tag{6c}$$

$$\mathbf{Q}^i(t+1) = \mathbf{Q}^i(t) + h_{ci}(t)\left[\mathbf{Q}_+^i(t) - \mathbf{Q}^i(t)\right]. \tag{6d}$$

Equations (6a) and (6b) perform a one step time shift of the memory vectors, while (6c) and (6d) take into account neighborhood collaboration, which means that only the memories of cells near the winning one are substantially updated.

Equation (4) can be interpreted as an implementation of a linear, time invariant, discrete time filter of order N (also called Learning Filter, LN), which has the following transfer function

$$G(z) = \frac{b_1 z^{N-1} + \ldots + b_N}{z^N - a_1 z^{N-1} - a_2 z^{N-2} \ldots - a_N}. \tag{7}$$

The transfer function (7) is defined using the Z-Transform formalism, and the coefficients $\mathbf{b} = [b_1, \ldots, b_N]^T$ and $\mathbf{a} = [a_1, \ldots, a_N]^T$ are fixed and define the particular filtering action between the input sequence and output sequence.

By substituting (6) in (5) the following implicit expression of the update rule is obtained

$$\mathbf{q}^i(t+1) = \mathbf{q}^i(t) + h_{ci}(t)\left(\mathbf{q}^i_+(t) - \mathbf{q}^i(t)\right), \quad i = 1 \ldots P \qquad (8)$$

where $\mathbf{q}^i_+(t) = \left[\mathbf{R}^i_+(t) \vdots \mathbf{Q}^i_+(t)\right]\begin{pmatrix}\mathbf{b}\\ \mathbf{a}\end{pmatrix}$. By comparing (8) with (2) it is evident that in the standard SOM the target vector is the randomly selected input pattern $\mathbf{r}(t)$, while in the modified version, the target is the filter output which is different for each cell. Thinking about the update direction of the model vectors, in the standard SOM it is radial with respect to $\mathbf{r}(t)$, while in the modified SOM it is not.

Optimization Algorithm description: Self Organizing Centroids-optimization

The use of the SOM as an optimization tool needs some changes with respect to the above described procedure. All the modern optimization techniques are based either on the concept of mutation or on the similarity with nature based processes characterized by interaction between different entities.

The heuristic strategy discussion of the algorithm will be extensively done later, but in order to make the algorithm better understandable, the rationale behind it can be easily expressed in the following way: a fitness function (or objective function) needs to be defined, and the winner cell (with respect to an input vector) is selected according to the minimum value of the objective function. Then all the centroids (not only the one of the winner cell) are perturbed, in search of a local minimum of the fitness function. Then the new input to the SOM is defined taking into account the previous winner (actual global minimum) and all the local minima found by the centroids perturbations.

Each centroid represents an individual that moves in the parameters space looking for better fitness values.

Cell model

The cell model is based on the model described in the previous section, with some additional features. From now on the input vectors will be denoted by $\mathbf{x}^i(t)$, the centroids by $\mathbf{c}^i(t)$, while r and q will still be used for the memory vectors.

Given an external input the centroid at each time step is given by the same combination of the memory vectors, $\mathbf{R}^i = \left[\mathbf{r}_1^i, \ldots \mathbf{r}_N^i\right]$ and $\mathbf{Q}^i = \left[\mathbf{q}_1^i, \ldots \mathbf{q}_N^i\right]$ as in (5).

$$\mathbf{c}^i(t) = \left[\mathbf{R}^i(t) \vdots \mathbf{Q}^i(t)\right]\begin{pmatrix}\mathbf{b}\\ \mathbf{a}\end{pmatrix} = \mathbf{M}^i(t)\cdot\mathbf{g} \tag{9}$$

i.e. the centroid moves only when the memory matrix $\mathbf{M}^i(t)$ changes.

Let us define a fitness function F, which must be minimized by the optimization procedure; F can be evaluated for each centroid at each time step as $F\left(\mathbf{c}^i(t)\right)$.

Each cell also contains a target vector $\mathbf{t}^i(t)$ which is the point with best fitness value $F_T^i(t) = F\left(\mathbf{t}^i(t)\right)$.The target vector is generated as a perturbation of $\mathbf{c}^i(t)$, as illustrated in the next section. The goal of the centroid $\mathbf{c}^i(t)$ is to reach the target of the cell $\mathbf{t}^i(t)$.

Another information present in each cell i is the index $neigh(i)$ of one cell belonging to the neighborhood of the cell i (the choice of the neighborhood cell is made a-priori). The distance, calculated in a P -norm, between the centroid $\mathbf{c}^i$ and the

neighbor $\mathbf{c}^{neigh(i)}$ will be used to determine the range of the perturbation of the cell i as it is shown later.

There are also a few global variables, which are known to each cell: one is the index of the winning cell (the cell that contains the global best fitness value) indicated as $gbest$, together with the corresponding fitness value $F_T^{gbest}(t)$ and the corresponding target point $\mathbf{t}^{gbest}(t)$. Also the last worst fitness value of the whole population $F_{\max}(t)$ is a global parameter, which allows the calculation of the deviation

$$e^i(t) = \frac{F_T^i(t) - F_T^{gbest}(t)}{F_{\max}(t) - F_T^{gbest}(t)}, \tag{10}$$

that indicates a relative goodness of the cell's personal target $\mathbf{t}^i(t)$ with respect to the global best point $\mathbf{t}^{gbest}(t)$ (note that $0 \le e^i \le 1$). The deviation $e^i(t)$ will be fundamental for the calculation of the perturbation and the update rate of the cells memory.

Algorithm description

At initialization $t = 0$ all the cells components need to be initialized: the memory vector $\mathbf{M}^i(0)$ of each cell is defined (more details about the memory initialization are given in following sections), and the centroids positions are consequently calculated as $\mathbf{c}^i(0) = \mathbf{M}^i(0) \cdot \mathbf{g}$. At $t = 0$ the targets are coincident with the centroids $\mathbf{t}^i(0) = \mathbf{c}^i(0)$, and their fitness values are straightforwardly calculated as $F_T^i(0) = F\left(\mathbf{c}^i(0)\right)$.

The winner cell is selected according to:

$$gbest(0) = \arg\min_i \left(F_T^i(0)\right), \tag{11}$$

and the best and worst values of the fitness function $F_T^{best}(0) = \min_i \left(F_T^i(0)\right)$ $F_{\max}(0) = \max_i \left(F_T^i(0)\right)$ are calculated. This allows the calculation of the deviation

$$e^i(0) = \frac{F_T^i(0) - F_T^{gbest}(0)}{F_{\max}(0) - F_T^{gbest}(0)}. \tag{12}$$

After the initialization, the algorithm can be divided in four fundamental steps: perturbation, winning cell selection, input vector creation and cells update, new centroid calculation.

perturbation

For each generation $t = 1, 2, \ldots, T_{\max}$, a perturbation is calculated according to

$$\mathbf{p}^i(t) = \mathbf{c}^i(t-1) + \boldsymbol{\delta}^i(t-1) e^i(t-1), \tag{13}$$

in which $\mathbf{p}^i(t)$ is a perturbed centroid, obtained from the centroid at the previous generation with the addition of a random perturbation $\boldsymbol{\delta}^i(t-1)$ whose components are uniformly distributed in $\left[-d_{neigh}, d_{neigh}\right]$, where $d_{neigh} = \left\|\mathbf{c}^i(0) - \mathbf{c}^{neigh(i)}(0)\right\|_p$. If the new point $\mathbf{p}^i(t)$ verifies the given constraints for the problem, i.e. if $\mathbf{x} \in \Omega$, then $\mathbf{p}^i(t)$ is kept, otherwise $\mathbf{p}^i(t) = \mathbf{c}^i(t-1)$. Now the fitness value in each perturbed point can be calculated according to $F_P^i(t) = F\left(\mathbf{p}^i(t)\right)$ and the target vectors of each cell, together with their corresponding fitness values, can be updated if the perturbed point improves the fitness:

$$\begin{cases} F_T^i(t) = F_P^i(t) \\ \mathbf{t}^i(t) = \mathbf{p}^i(t) \end{cases} \text{if } F_P^i(t) \le F_T^i(t-1) \qquad \begin{cases} F_T^i(t) = F_T^i(t-1) \\ \mathbf{t}^i(t) = \mathbf{t}^i(t-1) \end{cases} \text{if } F_P^i(t) > F_T^i(t-1) \quad . \tag{14}$$

Winning cell selection

The winning cell is now selected based on the best value of the fitness function:

$$gbest(t) = \arg\min_i \left(F_T^i(t) \right). \tag{15}$$

Note that due to the selection made in (14) $\min_i \left(F_T^i(t) \right)$ always corresponds to the global best fitness value found in all generations.

Input vector creation and cells update

The input $\mathbf{x}^i(t)$ to each cell i is a convex combination of the cell target $\mathbf{t}^i(t)$ and the best solution $\mathbf{t}^{gbest}(t)$

$$\mathbf{x}^i(t) = \lambda^i(t)\mathbf{t}^{gbest}(t) + \left[1 - \lambda^i(t)\right]\mathbf{t}^i(t), \tag{16}$$

where $\lambda^i(t)$ is a neighborhood function as in the classical SOM algorithm,

$$\lambda^i(t) = \exp\left(-\frac{d\left(gbest(t), i\right)^2}{2\sigma_\lambda^2} \right), \tag{17}$$

and $d\left(gbest, i\right)$ is the distance between the cells in the fixed grid arrangement of the SOM. Equation (16) and (17) show that for the winning cell, $\mathbf{x}^i(t) = \mathbf{t}^{gbest}(t)$, while for cells which are distant from the winning one, $\mathbf{x}^i(t) \to \mathbf{t}^i(t)$. The deviation $e^i(t)$

according to (10) is then calculated for each cell.

At this point also the memory vectors need to be updated. The rule which will be used is very similar to the one defined in equations (6), with the difference that a new function $w^i(t)$ needs to be introduced:

$$w^i(t) = \frac{1}{2}\left(\lambda^i(t) + h^i(t)\right), \tag{18}$$

where $\lambda^i(t)$ is the function defined in (17) and

$$h^i(t) = \exp\left(-\frac{e^i(t)^2}{2\sigma_h^2}\right), \tag{19}$$

is a goodness function which is maximum when the deviation $e^i(t)$ equals to zero, i.e. for the winning cell. It must be noted that $w^i(t)$ is maximum for the winning cell $w^i(t) = 1$, and is high for cells near the winning one (high values of $\lambda^i(t)$) and for cells with a low deviation (good local targets).

The memory vectors are consequently updated as:

$$\begin{cases} \mathbf{R}^i(t) = \mathbf{R}^i(t-1) + w^i(t)\left(\mathbf{R}^i_+(t) - \mathbf{R}^i(t-1)\right) \\ \mathbf{Q}^i(t) = \mathbf{Q}^i(t-1) + w^i(t)\left(\mathbf{Q}^i_+(t) - \mathbf{Q}^i(t-1)\right) \end{cases}, \tag{20}$$

where

$$\begin{cases} \mathbf{R}^i_+(t) = \left[\mathbf{x}^i(t), \mathbf{r}^i_1(t-1), \ldots \mathbf{r}^i_{N-1}(t-1)\right] \\ \mathbf{Q}^i_+(t) = \left[\mathbf{c}(t-1), \mathbf{q}^i_1(t-1), \ldots \mathbf{q}^i_N(t-1)\right] \end{cases}. \tag{21}$$

New centroid calculation

The new centroid is then calculated by using the standard relation

$$\mathbf{c}^i(t) = \left[\mathbf{R}^i(t) \vdots \mathbf{Q}^i(t)\right] \begin{pmatrix} \mathbf{b} \\ \mathbf{a} \end{pmatrix} ..$$

Now the algorithm can proceed with the perturbation step. The generation number can be now increased ($t = t + 1$) and all the steps are repeated until the termination condition is reached, which is typically a maximum number of generations or functions evaluations. The final best solution is represented by the vector $\mathbf{t}^{gbest}$ in the last iteration.

Heuristic strategy discussion

Summarizing the heuristic behind the proposed algorithm, the search strategy is based on a competition between local and global tasks. Each cell has a personal target which is a local best solution, whereas the entire network tracks a global best solution, so the input to each cell is a combination, based on $\lambda^i(t)$, of the local target and the global target. Cells near to the winner cell (high values of $\lambda^i(t)$), are more attracted by the global target than by their personal targets, and they tend to track it quickly because the memory update rate $w^i(t)$ is high as well. Cells far from the winner (low values of $\lambda^i(t)$) only follow their local targets, and their velocity depends on the goodness $h^i(t)$: higher values of the goodness imply higher memory update rate.

Besides following the global target each cell searches in the input space around the centroid for a possible better target, and this is accomplished by the perturbation of the centroid at each iteration. This operation is equivalent to mutation in DE or GA

algorithms, and in general it is not present in PSO algorithms. The strength of the perturbation is affected by two factors: the distance between the centroid and its neighbor, and the deviation of the personal fitness value (which is inversely proportional to the goodness of the personal solution). Each cell knows its centroid distance from a reference neighbor in the input space (d_{neigh}) and use this as the maximum magnitude of the perturbation. In this way the mutation of the centroids tends to adapt to the natural scaling of the problem and a better coverage of the input space is possible. Cells with better personal solution are less perturbed so that they search in a smaller area around the good solution, while cells with relatively bad solutions has major perturbations so that they search far from the centroid.

It is important to point out that the fitness function is calculated in the perturbed points verifying the constraints, hence a maximum of P function evaluations are performed for each generation. If the fitness in the perturbed points is better than the one of the targets, targets are updated (14); this operation is equivalent to the selection in DE or GA algorithms, in fact the fitness of target points $\mathbf{t}^i(t)$ remains the same or gets better according to (14).

The target points $\mathbf{t}^i(t)$ represent the candidate solutions, while the centroids $\mathbf{c}^i(t)$ act as the center of the perturbation and act as a moving agent tracking a combination of the local and global best solutions. This movement is implemented by a discrete filter of order *N*, so that each centroid has a certain inertia, or activity, to track its input, depending on the particular choice of the filter, which can be more or less predictive.

The proposed algorithm is a framework that accepts a number of different configurations. For example the topology between the cells can be created in the input

space following the Neural Gas paradigm instead of the SOM paradigm. Other variants of the proposed method can be defined using a different definition of the random perturbation $\boldsymbol{\delta}^i\left(t-1\right)$ in (13), for instance the random distribution can be chosen as Gaussian, while the amplitude of $\boldsymbol{\delta}^i\left(t-1\right)$ can be chosen as proportional to the distance between the centroid and the global best solution.

Parameters selection

The proposed method is characterized by some parameters to be selected. Regarding the coefficients of the learning filter, some guidelines are given that reduce the potential choices to first or second order discrete filters: this will be discussed in the numerical results section. The other parameters involved in the method are the variances σ_h^2 and σ_λ^2. The parameter σ_λ represents the width of the neighborhood in the fixed grid space, hence it is related to the fixed distances in the spatial arrangement of the cells. If an hexagonal arrangement is used and the distance between two adjacent cells is 1 then σ_λ is typically chosen so that the neighborhood include the first order and second order neighbors. For instance by choosing $\sigma_\lambda = 3$ the neighborhood function $\lambda^i\left(t\right)$ is negligible for third order neighbors. The parameter σ_h^2 determines the relationship between the deviation $e^i\left(t\right)\in\left[0,1\right]$ and the goodness function $h^i\left(t\right)$; by choosing $\sigma_h \in \left[0.1, 0.3\right]$ the goodness function is negligible for maximum deviation. The proposed method has significant different behaviors for different discrete filters.

Regarding space memory required, it increases linearly with the filter order. Acceptable space is required using a discrete filter with low order as 1, 2 or 3. Using a filter of order 1, the memory update in (20) and (21) yields exactly the classic SOM update, as defined by Kohonen (2001). In the context of the classic SOM use, Tucci and

Raugi (2011) have shown that Butterwort filters behave well for static distributions, while in moving environment a second order alpha-beta predictive filter has better tracking capabilities. Butterworth, filters have a smoother behavior, Rabiner (1975), while the alpha-beta filter is a predictive filter and its frequency response amplifies high frequencies. The equations of the alpha-beta filter are the following:

$$\begin{cases} \mathbf{q}(t+1) = \mathbf{q}(t) + \mathbf{v}(t) + \alpha\left[\mathbf{x}(t) - \left(\mathbf{q}(t) + \mathbf{v}(t)\right)\right] \\ \mathbf{v}(t+1) = \mathbf{v}(t) + \beta\left[\mathbf{x}(t) - \left(\mathbf{q}(t) + \mathbf{v}(t)\right)\right] \end{cases}, \tag{22}$$

where $[\alpha, \beta]$ represents the gains of the filter, $\mathbf{q}(t)$ is the centroid position, $\mathbf{v}(t)$ is a velocity variable and $\mathbf{x}(t)$ is the input observed position to track. The discrete time transfer function $G_{\alpha\beta}(z)$ between the measured input observation $\mathbf{x}(t)$ and the predicted output position $\mathbf{q}(t)$, is directly calculated from equations (22) as:

$$G_{\alpha\beta}(z) = \frac{\alpha z + \beta - \alpha}{z^2 + (\beta + \alpha - 2)z + (1 - \alpha)}. \tag{23}$$

For the alpha-beta filter only the normalized cut off frequency has to be chosen, which is included in the interval (0,1).

Memory Initialization

The proposed algorithm has a benefit if the initial arrangement of the centroids in the input space follows the ordered topology of the fixed grid of the cells. This condition can be obtained by calculating the initial distribution training a classical SOM algorithm, by considering as input distribution a uniform distribution covering the entire search domain. In this way the trained centroids will cover the domain of the search maintaining the desired ordered arrangement. If $\mathbf{c}^i(0), i = 1 \ldots P$ are the trained

centroids of the SOM, the memory vectors are initialized as: $\mathbf{r}_j^i(0) = \mathbf{c}^i(0)\ j = 1 \ldots N$, and $\mathbf{q}_j^i(0) = \mathbf{c}^i(0)\ j = 1 \ldots N$. In this way the relation $\mathbf{c}^i(0) = \sum_{j=1}^{N} b_j \mathbf{r}_j^i(0) + a_j \mathbf{q}_j^i(0)$, holds because $\sum_{j=1}^{N} b_j + a_j = 1$ by design.

SOC-opt and Particle Swarm Optimization

The proposed method is based on a search performed at two different levels: each centroid is attracted by a global target (the centroid with the actual lower value of the fitness function) and by a local target (the best value of the fitness function in the neighborhood). This philosophy is somehow similar to the Particle Swarm Optimization (PSO), in which the particles tend to move both towards the global best solution (particle with the actual lower value of the fitness function) and to the local best solution (lower value of the fitness function encountered by each particle).

The standard PSO updating procedure is defined as follows (Schutte and Groenwold 2005)

$$x(t+1) = x(t) + w\left(x(t) - x(t-1)\right) + c_1\phi_1\left(b_L - x(t)\right) + c_2\phi_2\left(b_G - x(t)\right), \quad (24)$$

where *w* is the inertia factor, which keeps the particle in its current trajectory; the last two terms inject deviation according to the distances to the personal b_L and global b_G best location through the cognitive factor c_1 and the social factor c_2, respectively (also called acceleration coefficients); ϕ_1 and ϕ_2 are two random variables distributed in the range $[0,1]$, which inject the unpredictability of the particles' movement.

In the proposed method, the particles' role is taken by the centroids, while b_L and b_G are the actual inputs of (24), whose role is taken by the input defined in (16).In

order to make equation (24) comparable to the SOC opt in terms of symbols, (24) is rewritten by substituting the particles' variable name $x(t)$ with $c(t)$

$$c(t+1)=c(t)+w\big(c(t)-c(t-1)\big)+c_1\phi_1\big(b_L-c(t)\big)+c_2\phi_2\big(b_G-c(t)\big). \quad (25)$$

By substituting (20) in (9), the following holds

$$\mathbf{c}(t+1)=\Big[\mathbf{R}(t)+w(t)\big(\mathbf{R}_+(t)-\mathbf{R}\ (t)\big)\vdots\mathbf{Q}(t)+w(t)\big(\mathbf{Q}_+(t)-\mathbf{Q}\ (t)\big)\Big]\begin{bmatrix}\mathbf{b}\\ \mathbf{a}\end{bmatrix}. \quad (26)$$

By using (9) again, (26) is rewritten as

$$\mathbf{c}(t+1)=\mathbf{c}(t)+w(t)\Big[\big(\mathbf{R}_+(t)-\mathbf{R}\ (t)\big)\vdots\big(\mathbf{Q}_+(t)-\mathbf{Q}\ (t)\big)\Big]\begin{bmatrix}\mathbf{b}\\ \mathbf{a}\end{bmatrix}. \quad (27)$$

By choosing a second order filter in (7), it is defined only by the coefficients b_1, b_2, a_1, a_2, yielding

$$\mathbf{c}(t+1)=\mathbf{c}(t)+w(t)\big[b_1\mathbf{x}(t)+b_2\mathbf{r}_1(t)+a_1\mathbf{c}(t)+a_2\mathbf{q}_1(t)-\mathbf{c}(t)\big],$$

which, regrouped, leads to

$$\mathbf{c}(t+1)=\mathbf{c}(t)+w(t)\big[a_1\mathbf{c}(t)+a_2\mathbf{q}_1(t)\big]+w(t)\big[b_1\mathbf{x}(t)+b_2\mathbf{r}_1(t)-\mathbf{c}(t)\big]. \quad (28)$$

By selecting $a_1=1,\ a_2=-1,\ b_2=0$ and considering that $\mathbf{q}_1(t)=\mathbf{c}(t-1)$, the following is obtained as a final result,

$$\mathbf{c}(t+1)=\mathbf{c}(t)+w(t)\big[\mathbf{c}(t)-\mathbf{c}(t-1)\big]+w(t)\big[b_1\mathbf{x}(t)-\mathbf{c}(t)\big], \quad (29)$$

which is actually very similar to (25), since the last two terms of (25) are related to the

inputs (local and global best) which in (29) are both included in the input $\mathbf{x}(t)$. In addition, the random factors are also included in the input $\mathbf{x}(t)$ according to (16).

The alternative form of the PSO update equation with the constriction factor adds an additional scalar term to the last two terms of (25), increasing the similarity with the proposed method (Schutte and Groenwold, 2005, Eberhart and Shi, 2000, Chatterjee and Siarry, 2006).

It is important to note that the similarity of the two methods is mainly on the particle update step, while the perturbation in the two methods is different.

The transfer function of the second order filter that results from (29) which represents a centroid update similar to PSO in the proposed framework, is given by:

$$G_{PSO}(z) = \frac{b_1 z}{z^2 - z + 1}.$$

It can be observed that this filter is marginally stable as it has two poles on the unit circle. The proposed model gives the flexibility to use different second order filters, and this adds more potential features to the method, and the similarity with the PSO becomes less evident.

Numerical experiments results

In this section, an experimental evaluation of the proposed framework is presented. The CEC 2013 benchmark suite, Liang, et al. (2013), which consists of 28 scalable benchmark functions, is considered. The definition of the 28 functions is provided by Lianget al. (2013). The functions have features that make them difficult minimization problems, such as epistasis, multimodality, noise, rotation and so on. In addition, they have been displaced in order to move the optimum away from the center of the search

space. The 28 functions can be divided into three classes: functions F1 to F5 are unimodal, while the other ones are multimodal. From F6 to F20 are basic multimodal functions, and F21 to F28 are composition functions (obtained by composing different functions amongst the ones previously defined), with a huge number of local minima. The number of variables of all the functions is scalable up to 50. All the problems are constrained, and the constraints are given as max- min bounds on the solution components. The dimensions D=10, 30 and 50 are considered, and a population size $P = 100$ (10x10 grid with hexagonal topology). The stop criterion is given by reaching a function evaluation limit of $10^4 \cdot D$ that is proposed by Lianget al. (2013)

The solution error measure is used to evaluate the performance of the algorithms, which is defined as $F\left(\mathbf{t}^{gbest}\left(T_{\max}\right)\right) - F\left(\mathbf{x}^{opt}\right)$, where $\mathbf{x}^{opt}$ is the known global optimum of the function, and $\mathbf{t}^{gbest}\left(T_{\max}\right)$ is the is the best solution achieved by the algorithm after $10^4 \cdot D$ function evaluations. Each problem is executed independently 51 times, to obtain an estimate of the mean solution error measure and its standard deviation.

Two implementations of the proposed algorithm for two choices of the filter are considered: a first order Butterworth filter and the alpha-beta filter. The configurations of the algorithms are the same for all the problems and dimensions. Table 1 summarizes the parameters selected for the two SOC-opt configurations, a cut-off frequency of 0.6 is selected for all the filters. The reason of this choice is that a normalized cut-off frequency of 0.6 represents a reasonable trade-off between exploitation, that require high values of the cut-off frequency, and exploration, that require low values of the cut-off frequency. The influence of the meta-parameters on the centroids trajectories, and the exploitation/exploration tradeoff are better evidenced in the last experiment in two dimensions. The statistical significance of the observed differences between the

algorithms, for each problem, is evaluated by means of the two-sided Wilcoxon rank sum test, with confidence level fixed to 0.95. This test evidences significant differences between pairs of algorithms. Two pairs of algorithms are considered for the Wilcoxon test, that are two variations of the proposed SOC-opt algorithm against the NBIPOPaCMA, a CMA-ES with restart described in Loshchilov (2013), which resulted to be best performing algorithm of the CEC 2013 benchmark together with the iCMAES-ILS algorithm, described in Liao et al. (2013).

Tables 2, 3 and 4 report the values of the mean and the standard deviation of the solution error measure for the two variants of SOC-opt, the NBIPOPaCMA and the icmaesils respectively for the 10, 30 and 50 variables problems. For the two SOC-opt variants also the results of the Wilcoxon test against the NBIPOPaCMA are shown. The null hypothesis of the test is that the compared values are independent and drawn from identical continuous distributions.

The symbol '=' is used to indicate that the null hypothesis is not rejected, so that the performance difference is not statistically significant. When the null hypothesis is rejected the symbol '+' indicates that the SOC-opt variant exhibits better performance than NBIPOPaCMA, while the symbol '−' is used if the proposed algorithm exhibits inferior performance with respect to NBIPOPaCMA. The last row of the table also shows the total number of occurrences of statistical cases +/ - /=. It is important to point out that the statistical significance has been assessed by considering the final results after 51 runs of the benchmark algorithm NBIPOPaCMA to perform the two-sided Wilcoxon rank sum test against the proposed SOC-opt solutions.

The results show that in the 30 variables case the SOC-opt based on the alpha beta filter is significantly better or equal to NBIPOPaCMA in 22/28 problems (14 better and 8 equal); the other SOC-opt variants (Butterworth and first order filter) are

characterized by reasonably good performances which are comparable to the NBIPOPaCMA for the first order filter and slightly better for the Butterworth filter. This shows that, in general, using the second order filter improves performance of SOC-opt, with respect to first order filter; in particular the SOC-opt with first order filter shows its limit with the composition functions with respect to the NBIPOPaCMA.

Figures 1-6 show the development of the mean error, among the 51 runs for SOC-opt alpha-beta and NBIPOPaCMA, between the best candidate solution and the optimal solution as a function of the number of evaluation, for some of the problems (10 dimensions case). It can be observed that the SOC-opt alpha-beta performance is in general comparable to NBIPOPaCMA.

Figures 7-10 show the movement of one centroid during the optimization process, for problem 1 in 2 dimensions. From figures 7 and 8 it can be observed that for fixed filter bandwidth and increasing the BW filter order the dynamics of the trajectory is smoother. Figures 9-10 show the centroid trajectory for fixed filter order (a second order alpha-beta filter), and changing the filter bandwidth. The trajectories for higher bandwidth values exhibit a faster movement. Figure 11 also shows the points obtained as centroids perturbations (13), where the objective function is actually calculated, for the case of Fig 9 that only shows the centroids. From Fig 11 it is evident that a slow trajectory allows the centroid to better explore the search space nearby, improving exploration, while a fast trajectory will converge faster to the best solutions found by the population, improving exploitation. We believe that the proposed model defines a tool where the tradeoff between exploration and exploitation, which is a fundamental paradigm of global optimization, can be easily controlled by the user, as they are directly related to the meta-parameters, represented by the filter order and filter bandwidth.

Conclusions

A new algorithm for optimization has been proposed, that defines a new strategy based on mutation and selection, where the individuals represent moving agents that track a personal target solution and are affected by a global dynamic through a neighborhood interaction. The main strength of the proposed algorithm is its flexibility to different competitive paradigms, which can be topology preserving like SOM or not like vector quantization or Neural Gas, and the possibility to design different centroids dynamics by selecting the low pass filter.

A drawback of the proposed algorithm is the memory requirement, as each cell contains $2N+1$ vectors of the dimension of the parameter vector, where N is the order of the filter. Anyway, good performance is obtained when the filter order is low, so memory requirement is not critical.

References

Auger, A., and Hansen, N. 2005."A restart CMA evolution strategy with increasing population size". In *Evolutionary Computation, IEEE International Conference on*, 1769–1776.

Back, T. 1996.*Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York: Oxford University Press.

Barmada, S.,Raugi, M., and Tucci, M. 2012."Global optimization algorithm based on self-organizing centroids," in *Evolutionary Computation (CEC), 2012 Congress on IEEE*,1-6.

Chatterjee, A., andSiarry, P. 2006. "Nonlinear Inertia Weight Variation for Dynamic Adaptation in Particle Swarm Optimization." *Computers & Operations Research* 33: 859–871.

De Bodt, E., Cottrell, M., Letremy, P., and Verleysen, M. 2004. "On the use of self-organizing maps to accelerate vector quantization".*Neurocomputing* 56: 187-203

Eberhart, R. C., and Shi, Y. 2000. "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization". In. *Evolutionary Computation, IEEE International Conference on*, 84 – 88.

Grosan, C., Hassanien, A.E., 2012. "Hybrid self organizing neurons and evolutionary algorithms for global optimization.*"Journal of Computational and Theoretical Nanoscience*9: 304-309.

Kohonen, T. 2001.*Self-Organizing Maps, Third Edition*. Berlin: Springer.

Liang, J. J., Qu, B. Y., Suganthan, P. N., and Hernandez – Diaz, A. G. 2012. *Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization.*Technical Report 201212, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China 01/2013.

Liao, T. and Stutzle T. 2013. "Benchmark Results for a Simple Hybrid Algorithm on the CEC 2013 Benchmark Set for Real-parameter Optimization". In *Evolutionary Computation, IEEE Congress on,* 1938 – 1944.

Loshchilov, I. 2013. "CMA – ES with Restarts for Solving CEC 2013 Benchmark Problems". In *Evolutionary Computation, IEEE Congress on,* 369 – 376.

Rabiner, L. R., & Gold, B. (1975). Theory and application of digital signal processing. *Englewood Cliffs, NJ, Prentice-Hall, Inc., 1975. 777 p., 1*.

Schutte, J. F., and Groenwold, A.A. 2005. "A Study of Global Optimization Using Particle Swarms".*Journal of Global Optimization* 31: 93–108.

Tucci, M., and Raugi, M. 2009."A Learning Algorithm for Self-Organizing Maps Based on a Low-Pass Filter Scheme." In *International Conference on Adaptive and Intelligent Systems ICAIS 2009*, 31-36.

Tucci, M., and Raugi, M. 2010."Adaptive FIR Neural Model for Centroid Learning in Self-Organizing Maps." *IEEE Transactions on Neural Networks* 21: 948-960.

Tucci, M., and Raugi, M. 2011."A filter based neuron model for adaptive incremental learning of self-organizing maps". *Neurocomputing*: 74, 1815-1822.

Table 1. Parameters of the SOC-opt algorithm

SOC-opt first order filter			**SOC-opt alpha-beta**		
Filter	σ_λ	σ_h	***Filter***	σ_λ	σ_h
$a_1 = 0.549$ $b_1 = 0.451$	3	0.3	$\alpha = 0.669$ $\beta = 0.360$	3	0.3

Table 2. Mean values and standard deviations of the solution error measure, 10 variables.

	SOC-opt First order filter			**SOC-opt Alpha beta**			**NBIPOP-aCMA**			**iCMAES-ILS**	
	Mean	***St.D***		***Mean***	***St.D***		***Mean***	***St.D***		***Mean***	***St.D***
*F*1	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*2	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*3	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*4	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*5	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*6	3.73	4.303	-	0.00	0.00	=	0.00	0.00		3.89	4.80
*F*7	8.65e-02	9.30e-01	-	4.21e-06	9.02e-06	-	0.00	0.00		4.91e-06	1.34e-05
*F*8	2.00e+01	1.14e-01	=	2.01e+01	9.00e-02	=	2.03e+01	9.00e-02		2.04e+01	7.61e-02
*F*9	2.81e-01	5.22e-01	-	2.18e-01	4.33e-01	=	2.32e-01	4.40e-01		2.86e-01	5.38e-01
*F*10	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.000	0.000
F11	9.95e-01	5.34e-01	-	3.48	3.91e-01	+	3.64e-01	5.06e-01		4.77e-01	5.71e-01
F12	2.48e-01	5.95e-01	=	2.35e-01	5.30e-01	=	2.38e-01	5.42e-01		2.34e-01	4.26e-01
F13	5.02e-01	7.49e-01	=	4.24e-01	5.31e-01	+	4.84e-01	6.76e-01		3.33e-01	4.73e-01
F14	1.69e+02	1.80e+02	-	7.50e+01	1.04e+02	+	1.15e+02	9.24e+01		5.08e+01	9.99e+01
F15	1.59e+02	1.52e+02	=	1.38e+02	1.01e+02	+	1.58e+02	1.17e+02		4.42e+01	1.02e+02
F16	2.89e-02	3.25e-01	-	4.30e-02	2.19e-02	+	1.20e-01	2.63e-01		3.73e-01	3.00e-01
F17	1.36e+01	5.80e-01	-	1.13e+01	5.28e-01	=	1.13e+01	5.45e-01		1.12e+01	5.08e-01
F18	1.44e+01	5.74	-	1.30e+01	5.03	-	1.13e+01	1.28		1.12e+01	5.01e-01
F19	4.07e-01	1.12e-01	+	4.80e-01	1.28e-01	+	5.25e-01	1.39e-01		6.98e-01	1.50e-01
F20	2.34	5.49e-01	+	2.05	3.86e-01	+	2.73	6.50e-01		2.72	5.24e-01
F21	4.00e+02	0.00	-	4.00e+02	0.00	-	1.53e+02	5.04e+01		2.18e+02	1.11e+02
F22	1.97e+02	1.38e+02	-	1.91e+02	1.16e+02	-	1.75e+02	1.15e+02		1.66e+02	8.10e+01
F23	1.56e+02	1.19e+02	+	1.60e+02	1.28e+02	+	1.74e+02	1.23e+02		4.08e+01	2.08e+01
F24	1.11e+02	3.01e+01	=	1.17e+02	3.10e+01	=	1.20e+02	3.22e+01		1.32e+02	3.25e+01
F25	1.81e+02	2.82e+01	=	1.60e+02	2.65e+01	+	1.77e+02	3.99e+01		1.92e+02	2.47e+01
F26	1.04e+02	1.44e+01	+	1.03e+02	7.55	+	1.11e+02	2.50e+01		1.18e+02	1.30e+01
F27	3.20e+02	3.12e+01	=	4.00e+02	0.00	-	3.17e+02	2.96e+01		3.25e+02	4.20e+01
F28	2.50e+02	1.07e+02	=	2.44e+02	9.66e+01	=	2.49e+02	8.80e+01		2.24e+02	1.01e+02
	+/-/=	4/10/14		+/-/=	10/5/13						

Table 3.Mean values and standard deviations of the solution error measure, 30 variables.

	SOC-opt First order filter			**SOC-opt Alpha beta**			**NBIPOP-aCMA**			**iCMAES-ILS**	
	Mean	***St.D***		***Mean***	***St.D***		***Mean***	***St.D***		***Mean***	***St.D***
*F*1	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*2	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*3	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*4	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*5	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*6	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*7	2.30	5.41	+	2.28	3.36	+	2.31e+00	6.05		7.01e-02	1.56e-01
*F*8	2.28e+01	5.6e-02	-	2.14e+01	0.12	-	2.09e+01	0.05		2.09e+01	6.23e-02
*F*9	3.00	1.29	+	3.13	1.28	+	3.30	1.38		4.34	1.72
*F*10	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
F11	4.27	2.00	-	3.88	1.39	-	3.04	1.41		2.25	1.05
F12	2.37	1.18	+	2.70	1.38	+	2.91	1.38		1.72	1.23
F13	2.69	1.23	+	2.28	1.49	+	2.78	1.45		2.16	1.30
F14	8.00e+02	5.42e+02	-	6.58e+02	2.37e+02	+	8.10e+02	3.60e+02		7.08e+02	2.94e+02
F15	7.42e+02	2.09e+02	+	4.57e+02	2.64e+02	+	7.65e+02	2.95e+02		2.59e+02	1.18e+02
F16	6.54e-01	9.69e-01	-	8.97e-01	1.29	-	4.40e-01	0.93		3.75e-01	2.65e-01
F17	3.28e+01	1.33	+	2.75e+01	1.32	+	3.44e+01	1.87		3.43+01	1.86
F18	6.55e+01	4.77e+01	=	6.23e+01	4.50e+01	=	6.23e+01	4.56e+01		4.01e+01	1.87e01
F19	6.83	4.65e-01	-	2.85	0.44	-	2.23	3.41e-01		2.24	5.66e-01
F20	9.76	7.26e-01	+	9.27	0.33	+	1.29e+01	5.98e-01		1.44e+01	7.38e-01
F21	1.92e+02	3.10e+01	=	2.15e+02	31.84	-	1.92e+02	2.72e+01		1.88e+01	3.25e+01
F22	9.76e+02	4.33e+02	-	4.36e+02	2.35e+02	+	8.38e+02	4.60e+02		5.33e+02	3.63e+02
F23	6.58e+02	2.69e+02	+	3.51e+02	1.34e+02	+	6.67e+02	2.90e+02		2.69e+02	1.41e+02
F24	1.61e+02	291.632	=	8.63e+01	2.35e+01	+	1.62e+02	3.00e+01		2.00e+02	6.16e-04
F25	2.58e+02	1.20e+01	-	3.04e+02	1.58e+01	-	2.20e+02	1.11e+01		2.40e+02	5.12
F26	1.34e+02	1.74e+01	+	9.74e+01	7.32	+	1.58e+02	3.00e+01		2.16e+02	3.67e+01
F27	4.79e+02	7.50e+01	-	2.65e+02	4.33e+01	+	4.69e+02	7.38e+01		3.00e+02	9.34e-03
F28	3.88e+02	8.00e+01	-	2.01e+02	3.24e+01	+	2.69e+02	7.35e+01		2.45e+02	9.01e+01
	+/-/=	9/9/10		+/-/=	14/6/8						

Table 4.Mean values and standard deviations of the solution error measure, 50 variables.

	SOC-opt First order filter			**SOC-opt Alpha beta**			**NBIPOP-aCMA**			**iCMAES-ILS**	
	Mean	***St.D***		***Mean***	***St.D***		***Mean***	***St.D***		***Mean***	***St.D***
*F*1	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*2	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*3	2.14e+01	1.41e+02	-	1.93e+01	1.40e+02	-	1.82e+01	1.21e+02		2.01e-02	1.08e-1
*F*4	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
*F*5	0.00	0.00	=	0.00	0.00	=	0.00	0.00		1.52e-08	0.00
*F*6	0.00	0.00	=	0.00	0.00	=	0.00	0.00		4.19e+01	7.82
*F*7	4.64	5.39	+	3.42	4.33	+	4.97	5.72		5.44e-01	1.10
*F*8	1.98e+01	3.9e-02	+	1.5e+014	4.00e-02	+	21.1	4.5e-02		2.11e+01	6.29e-02
*F*9	1.40e+01	2.81	-	4.32	1.48	-	7.22	2.29		8.18	3.28
*F*10	0.00	0.00	=	0.00	0.00	=	0.00	0.00		0.00	0.00
F11	8.73	3.00	-	5.00	1.36	-	5.51	2.96		5.94	1.98
F12	5.03	1.80	+	3.55	1.37	+	5.37	2.54		5.77	1.81
F13	1.68e+01	5.98	-	8.94	6.43	-	7.56	5.47		5.73	2.64
F14	1.23e+03	5.43e+02	+	6.77e+02	4.37e+02	+	1.38e+03	5.67e+02		8.59e+02	7.10e+02
F15	2.01e+03	5.53e+02	-	9.82e+02	0.27e+02	+	1.55e+03	5.48e+02		6.42e+02	2.97e+02
F16	1.21	9.98e-01	-	8.43e-01	1.45	=	8.78e-01	1.44		6.28e-01	3.53e-01
F17	1.03e+02	6.94	-	9.84e+01	7.87	-	5.74e+01	2.73		5.75e+01	1,57
F18	1.29e+02	9.88e+01	+	1.00e+02	8.99e+01	+	1.34e+02	1.00e+02		6.43e+01	8.39
F19	4.00	4.72e-01	+	4.41	6.21e-01	=	4.46	5.93e-01		3.62	8.79e-01
F20	2.03e+01	1.00	+	1.84e+01	9.99e-01	+	2.25e+01	1.18		2.44e+01	4.52e-01
F21	3.16e+02	2.29e+01	-	2.20e+02	1.69e+01	-	1.98e+02	1.40e+01		2.00e+02	0.000
F22	1.22e+03	5.87e+02	+	7.85e+02	5.43e+02	+	1.45e+03	6.01e+02		5.87e+02	5.62e+02
F23	2.51e+03	8.84e+02	-	2.01e+03	9.00e+02	-	1.71e+03	8.09e+02		5.57e+02	3.26e+02
F24	3.21e+02	2.20e+01	-	1.99e+02	2.28e+01	-	2.40e+02	2.04e+01		2.00e+02	5.39e-02
F25	4.89e+02	1.64e+01	-	3.13e+02	1.29e+01	-	2.48e+02	5.06		2.74e+02	6.24
F26	3.22e+02	2.24e+01	-	1.99e+02	13.953	=	1.96e+02	1.43e+01		2.41e+02	4.97e+01
F27	5.98e+02	1.33e+02	+	4.32e+02	1.05e+02	+	7.28e+02	1.44e+02		3.02e+02	1.49e+01
F28	4.00e+02	0.00	=	4.00e+02	0.00	=	4.00e+02	0.00		4.00e+02	0.000
	+/-/=	9/12/7		+/-/=	12/6/10						

Figure 1. Mean error versus number of function evaluations for problem 23

Figure 2. Mean error versus number of function evaluations for problem 19

Figure 3. Mean error versus number of function evaluations for problem 18

Figure 4. Mean error versus number of function evaluations for problem 17

Figure 5. Mean error versus number of function evaluations for problem 16

Figure 6. Mean error versus number of function evaluations for problem 15

Figure 7. Centroids movement in the search space for problem 1 in two dimensions, the filter order changes while the band of the filter is fixed.

Figure 8. Centroids movement in the search space for problem 1 in two dimensions, the filter order changes while the band of the filter is fixed. Zoomed area around the final (optimal) point

Figure 9 Centroids movement in the search space for problem 1 in two dimensions, the filter order is fixed while the band of the filter changes

Figure 10 Centroids movement in the search space for problem 1 in two dimensions, the filter order is fixed while the band of the filter changes. Zoomed area around the final (optimal) point

Figure 11 Centroids and perturbation points in the search space for problem 1 in two dimensions, the filter order is fixed while the band of the filter changes.