INFORMATICA, 2016 Vol. 0, No. 0, 1–0

CoRSO (Collaborative Reactive Search Optimization): Blending Combinatorial and Continuous Local Search

Mauro BRUNATO, Roberto BATTITI

Dept. of Information Engineering and Computer Science, University of Trento, Italy
E-mail: {mauro.brunato,roberto.battiti}@unitn.it

Abstract.

We propose a heuristic global optimization technique which combines combinatorial and continuous local search. The combinatorial component, based on Reactive Search Optimization, generates a trajectory of binary strings describing search districts. Each district is evaluated by random sampling and by selective runs of continuous local search. A reactive prohibition mechanisms guarantees that the search is not stuck at locally optimal districts.

The continuous stochastic local search is based on the *Inertial Shaker* method: candidate points are generated in an adaptive search box and a moving average of the steps filters out evaluation noise and high-frequency oscillations.

The overall subdivision of the input space in a tree of non-overlapping search districts is adaptive, with a finer subdivision in the more interesting input zones, potentially leading to lower local minima.

Finally, a portfolio of independent *CoRSO* search streams (*P-CoRSO*) is proposed to increase the robustness of the algorithm.

An extensive experimental comparison with Genetic Algorithms and Particle Swarm demonstrates that *CoRSO* and *P-CoRSO* reach results which are fully competitive and in some cases significantly more robust.

1. Introduction. Global optimization, in particular with stochasticity (Zhigljavsky and Žilinskas, 2007), presents a suite of techniques and theoretical results for solving optimization problems. Now, it is generally assumed that competitive results for a specific technique can be obtained only for selected classes of functions, in particular for high-dimensional problems. While this paper has no space for an extensive review of methods assuming specific properties of the functions to be optimized, let's mention a couple of notable examples.

A first context studied by many researchers assumes that functions satisfy the Lipschitz condition (Sergeyev and Kvasov, 2015). If $f(x)$ satisfies the Lipschitz condition over the search hyper-interval with an unknown Lipschitz constant K, a deterministic Divide-the-Best algorithm based on efficient diagonal partitions of the search domain and smooth auxiliary functions is proposed in (Sergeyev and Kvasov, 2015). The method adaptively estimates the unknown Lipschitz constant

K and the objective function and its gradient are evaluated only at two vertices corresponding to the main diagonal of the generated hyperintervals.

A second well-known case assumes that the functions satisfies some statistical model, so that theoretically justified methods can be developed, in the framework of rational decision making under uncertainty, to generate new sample points based on information derived from previous samples, and to study convergence properties. The book (Zhigljavsky and Žilinskas, 2007) contains an in-depth presentation of the topic. Let's note that statistics is an essential part of machine learning: techniques developed by different communities should be considered jointly in the bag of tools to address challenging optimization instances. In particular, the models of functions can be random processes in the case of functions of one variable, and random fields in the case of functions of many variables. As an example, for a Wiener process $\xi(x)$ in one dimension (the limit of a random walk), one can derive an analytical formula for the probability distribution of the minimum given the previously evaluated sample points, and use it for a rational stopping condition with a predefined tolerance. In multiple dimensions, random fields are a possible model but with a high inherent computational complexity motivating the study of simpler statistical models. The P-algorithm (Žilinskas, 1985) generates the next point to be evaluated as the one maximizing the probability to improve the current record, given the previously observed samples. In multiple dimensions, if y_{on} is the current record value and (x_i, y_i) are the previous evaluated points and corresponding values, the next $(n+1)$-th optimization step is defined as:

$$x_{n+1} = \operatorname*{argmax}_x \Pr\Big\{\xi(x) \leqslant (y_{on} - \epsilon) \mid \xi(x_1) = y_1, ..., \xi(x_n) = y_n\Big\}.$$

The P-algorithm with simplicial partitioning is proposed in (Žilinskas and Žilinskas, 2002) to obtain a practical algorithm. The observation points coincide with the vertices of the simplices and different strategies for defining an initial covering and subdividing the interesting simplices are proposed and considered. The P^*-algorithm, combining the P-algorithm with local search, is related to the algorithm presented in this paper, which is also based on a combination of global models and efficient local searches when the current area is deemed sufficiently interesting.

Response surface methods substitute the real function with a surrogate model learnt by means of the previously evaluated points. The surrogate model is then optimized at each step in place of the original function, which can be very computationally efficient if evaluating f is very costly (like in engineering simulations) and if the surrogate model is sufficiently fast to be optimized. For example, a response surface built with radial basis functions centered on the evaluated points is considered in (Gutmann, 2001).

In practical applications, verifying that a specific function obeys a given model can be difficult, in particular if few instances are of interest, and not an infinite collection of them. In this case, the use of theoretically justified methods is of course possible, in some cases highly desirable, but at the price of abandoning the comfort zone of guarantees derived from the statistical analysis.

If the function does not possess a rich ("non-random") structure, convergence to the global optimum can become painfully slow and smart techniques become comparable to, or even worse than, pure random search. As an example of the "curse of dimensionality", to reach an approximation of the global optimizer with a specific probability, pure random search requires a number of iterations which increases exponentially as the input dimension d increases. No structure, no hope of efficient optimization!

On the other hand, most real-world problems have a "black box" nature. One can evaluate the output given the inputs (for example by engineering simulators), but no analytic form and no hints about the function structure are given *a priori*. Furthermore, in many cases just a single relevant instance has to be solved at a given time (like designing a new airplane wing to reduce fuel consumption). An effective strategy in these cases consists of adopting machine learning methods to learn some elements of the instance structure in an online (reactive) manner so that the proper solution technique can be chosen or adapted (Battiti *et al.*, 2008).

In this paper we design two reactive (adaptive) techniques, called *CoRSO* and *P-CoRSO*, which integrate these elements: local-search building blocks scouting for local minima (working with continuous variables) and a discrete (combinatorial) prohibition-based search acting on search boxes identified by binary strings. The method is related to the "branch and probability bound" technique described in (Zhigljavsky and Žilinskas, 2007), being based on (i) branching the optimisation set into a tree of subsets of the input space, (ii) making decisions about the perspectiveness of the subsets for further search, and (iii) selecting the subsets that are recognized as perspective for further branching.

CoRSO is based on C-RTS (Battiti and Tecchiolli, 1996), but focuses on the Inertial Shaker method for continuous stochastic local search instead of the more complex and less scalable Reactive Affine Shaker. *P-CoRSO* builds an additional coordination level among a portfolio of independent *CoRSO* runs to increase the robustness for some deceptive functions.

The specific proposal is to build a *trajectory of subsets* via local search (LS) acting on binary strings, plus a prohibition mechanism for diversification which ensures that LS is not stuck at locally optimal subsets. In addition, all previously evaluated points are saved in memory and used for simple statistical inference about the number of local minima present in a subset and to evaluate the perspectiveness of the subsets.

Local search on binary strings is an effective building block for solving complex combinatorial optimization problems, and the local minima traps can be cured by Reactive Search Optimization (Battiti *et al.*, 2008). *CoRSO* extends RSO to the case of *continuous optimization* problems, with input variables consisting of real numbers, solved by a team of local searchers. *CoRSO* uses a framework for solving continuous optimization problems by a strategic use of memory and *cooperation* among a team of self-adaptive local searchers.

The three pillars of *CoRSO* are: multiple local searchers in charge of districts (portions of input space), mutual coordination, and continuous "reactive" learning and adaptation.

CoRSO adopts a *sociological/political paradigm.* Each local searcher takes care of a *district* (an input area), generates samples and decides when to fire local search in coordination with the other members. The *organized subdivision of the configuration space* is adapted in an online manner to the characteristics of the problem instance. *Coordination by use of globally collected information* is crucial to identify promising areas in configuration space and allocate search effort.

2. Intelligent coordination of local search processes. To fix the notation and the direction, let's assume that we aim at *minimizing* a function $f(x)$ defined over a set of continuous variables x. No constraints are present apart from simple bounds on each input variable. Models of cultural evolution inspire a set of powerful optimization techniques known as *Memetic Algorithms (MAs).* According to a seminal paper (Moscato, 1989), memetic algorithms are population-based approaches that combine a fast heuristic to improve a solution (and even reach a local minimum) with a recombination mechanism that creates new individuals.

The fast heuristic to improve a solution is some form of *local search.* LS generates a search trajectory in configuration space $X^{(t)}$, depending on the iteration counter t, so that the next point $X^{(t+1)}$ is selected from a set of neighbors, with a bias towards points with lower function values. The motivation for the effectiveness of stochastic local search for many real-world optimization tasks lies in the *correlation between function values at nearby points*: The probability to find points with lower values is larger for neighbors of points which are *already* at low function values.

In many cases a given optimization instance is characterized by *structure at different levels*, as explained with the big valley property (more details in (Battiti *et al.*, 2008)). If we reduce the initial search space to a set of attractors (the local minima), again it may be the case that nearby attractors —having an attraction basin close to each other— tend to have correlated values. This means that *knowledge* of previously found local optima can be used to direct the future investigation efforts. Starting from initial points close to promising attractors favors the discovery of other good quality local optima, provided that a sufficient diversification mechanism avoids falling back to previously visited ones.

In sequential local search the knowledge accumulated about the fitness surface flows from past to future searches, while in parallel processes with more local searchers active at the same time, knowledge is transferred by mutual sharing of partial results. We argue that the relevant subdivision is not between sequential and parallel processes (one can easily simulate a parallel process on a sequential machine) but between different ways of *using the knowledge accumulated by set* *of local search streams to influence the strategic allocation of computing resources* to the different LS streams, which will be activated, terminated, or modified depending on a shared knowledge base, either accumulated in a central storage, or in a distributed form but with a periodic exchange of information.

MAs fit in this picture, a set of individuals described by genes and subjected to genetic evolution scouts the fitness surface to search for successful initial points, while LS mechanisms (analogous to life-time learning) lead selected individuals to express their full potential by reaching local optima through local search. The *Genetic Algorithms* used in standard MAs follow the biological paradigms of selection/reproduction, cross-over and mutation. While GAs are popular for many applications, there is actually no guarantee that specific biologically-motivated genetic operators must be superior to *human-made direct mechanisms to share the knowledge accumulated* about the fitness surface by a set of parallel search streams (a.k.a. population). Alternative coordination mechanisms have been proposed for example in (Törn and Viitanen, 1992) with the name of "topographical" global optimization, based on "clustering" methods (Törn and Žilinskas, 1989). The idea is to identify possible attraction basins by first sampling points, then defining a directed graph (each point is connected to a neighbor with higher function value, for k nearest neighbors), and finally identifying points with all neighbors having larger function values as candidate starting points for local optimization.

The rationale behind *CoRSO* is to design mechanisms with a higher level of coordination to effectively manage many local search streams. One is not constrained by genetic algorithms but free to experiment with different and more organized ways of coordinating search streams, following sociological and political paradigms. Knowledge is transferred between different searchers in a way similar to efficient political organizations, like a smoothly living community of monks.

3. CoRSO: a political analogy. Although not necessary, analogies can help in reasoning about problems. But if we accept the helpfulness of analogies, we prefer analogies derived from the human experience more that analogies based on animals or genetics. Politics is a process by which groups of people make collective decisions. Groups can be governments, but also corporate, academic, and religious institutions.

Local search is an effective building block for starting from an initial configuration of a problem instance and progressively building better solutions by moving to neighboring configurations. In an organized institution, like a corporation composed of individuals with intelligent problem-solving capabilities, each expert, when working on a tentative solution in his competence area, will after some time come up with an improved solution. The objective is to strategically allocate the work so that, depending on the accumulated performance of the different experts and competencies, superior solutions are obtained.

Memetic Algorithms start from local search and consider a hybridized genetic mechanism to implicitly accumulate knowledge about past local search performance by the traditional biologically-motivated mechanisms of selection/reproduction, mutation and cross-over. The first observation is that an individual can exploit its initial genetic content (its initial position) in a more directed and determined way. This is effected by considering the initial string as a *starting point* and by initiating a run of local search from this initial point, for example scouting for a local optimum. The term *memetic algorithms* (Krasnogor and Smith, 2005, Moscato, 1989) has been introduced for models which combine the evolutionary adaptation of a population with individual learning within the lifetime of its members. Actually, there are two obvious ways in which individual learning can be integrated: a first way consists of replacing the initial genotype with the better solution identified by local search (*Lamarckian evolution*), a second way can consist of modifying the fitness function by taking into account not the initial value but the final one obtained through local search. In other words, the fitness does not evaluate the initial state but the value of the "learning potential" of an individual, measured by the result obtained after local search. This evaluation changes the fitness landscape, while the evolution is still Darwinian in nature.

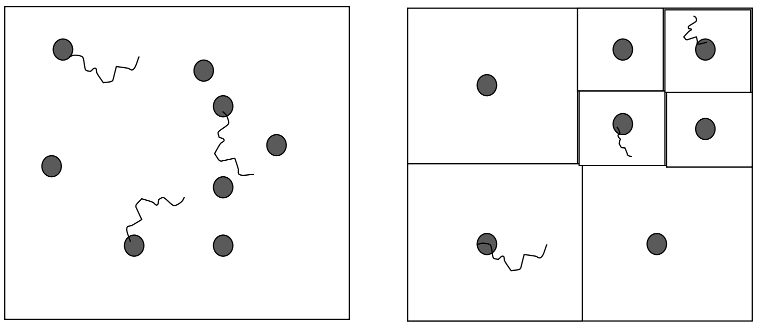

Fig. 1. Different ways of allocating local searchers: by Memetic Algorithms (left) and by a political analogy of *CoRSO* (right). Crosses represent starting points, circles local optima reached after running local search. In the second case each individual is responsible for an area of configuration space. Some local search streams are shown.

When the road of cultural paradigms is followed, it is natural to consider models derived from organizations of intelligent individuals equipped with individual learning and social interaction capabilities also in the *strategic allocation of resources* to the different search streams. In particular, this work presents a hybrid algorithm for the global optimization of functions, in which a fast combinatorial component (the Reactive Search Optimization based on prohibitions) identifies promising *districts* (boxes) in a tree-like partition of the initial search space, and a stochastic local search minimizer (the Inertial Shaker — IS — algorithm) finds the local minimum in a promising attraction basin.

The development of the *CoRSO* framework is guided by the following design principles.

- **General-purpose optimization:** no requirements of differentiability or continuity are placed on the function f to be optimized.
- **Global optimization:** while the local search component identifies a local optimum in a given attraction basin, the combinatorial component favors jumps between different basins, with a *bias* toward regions that plausibly contain good local optima.
- **Multi-scale search:** the use of grids at different scales in a tree structure is used to spare CPU time in slowly-varying regions of the search space and to intensify the search in critical regions.
- **Simplicity, reaction and adaptation:** the algorithmic structure of *CoRSO* is simple, the few parameters of the method are adapted in an automated way during the search, by using the information derived from memory. The intensification-diversification dilemma is solved by using intensification until there is evidence that diversification is needed (when too many districts are repeated excessively often along the search trajectory). The tree-like discretization of the search space in districts is activated by evidence that the current district contains more than one attraction basin.
- **Tunable precision:** the global optimum can be located with high precision both because of the local adaptation of the grid size and because of the decreasing sampling steps of the stochastic IS when it converges.

CoRSO is characterized by an efficient use of *memory* during the search, as advocated by the Reactive Search Optimization. In addition, simple *adaptive* (*feedback*) mechanisms are used to tune the space discretization, by growing a *tree* of search districts, and to adapt the prohibition period of RSO acting on prohibitions. This adaptation limits the amount of user intervention to the definition of an initial search region, by setting upper and lower bounds on each variable, no parameters need to be tuned.

CoRSO fuses combinatorial Reactive Search Optimization with an efficient stochastic Local Search component. An instance of an optimization problem is a pair $(\mathcal{X}, f)$, where $\mathcal{X}$ is a set of feasible points and f is the cost function to be *minimized*: $f : \mathcal{X} \to \mathbb{R}$. In the following we consider *continuous optimization* tasks where $\mathcal{X}$ is a compact subset of $\mathbb{R}^N$, defined by bounds on the N independent variables x_i, where $B_{Li} \leq x_i \leq B_{Ui}$ (B_L and B_U are the lower and upper bounds, respectively).

In many popular algorithms for continuous optimization one identifies a "local minimizer" that locates a local minimum by descending from a starting point, and a "global" component that is used to diversify the search and to reach the global optimum. We define as *attraction basin* of a local minimum X_l the set of points that will lead to X_l when used as starting configurations for the local minimizer.

In some cases, as we noted in our starting assumptions, an effective problem-

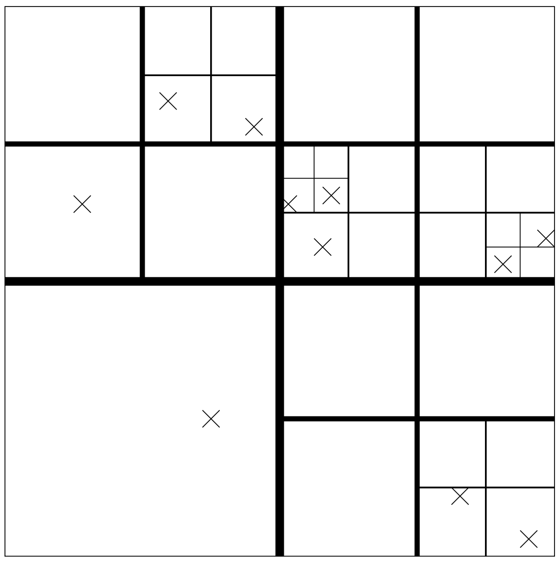

Fig. 2. *CoRSO*: tree of search *districts*. Thick borders identify upper-level boxes, crosses show the position of local minima found by the search component; finer divisions are the tree leaves and partition the whole function domain.

specific local search component is available for the problem at hand, and one is therefore motivated to consider a *hybrid strategy*, whose local minimizer has the purpose of finding the local minimum with adequate precision, and whose combinatorial component has the duty of discovering promising attraction basins for the local minimizer to be activated. Because the local minimizer is costly, it is activated only when the plausibility that a region contains a good local optimum is high. On the contrary, a fast evaluation of the search districts is executed by the combinatorial component, and the size of the candidate districts is adapted so that it is related to that of a single attraction basin. A district is split when there is evidence that at least two different local minima are located in the same district.

4. CoRSO: blending RSO with stochastic local search. In the hybrid *CoRSO* scheme, *RSO* identifies promising regions for the local minimizer to be activated. In this section we describe how the two components are interfaced. The specific stochastic local search component Inertial Shaker (IS) will be presented in Section 4.4.

The basic structure through which the initial search region is partitioned consists of a *tree of districts* (boxes with axes parallel to the coordinate axes), see Fig. 2. The tree is born with 2^N equal-size leaves, obtained by dividing in half the initial range on each variable. Each district is then subdivided into 2^N equally-sized

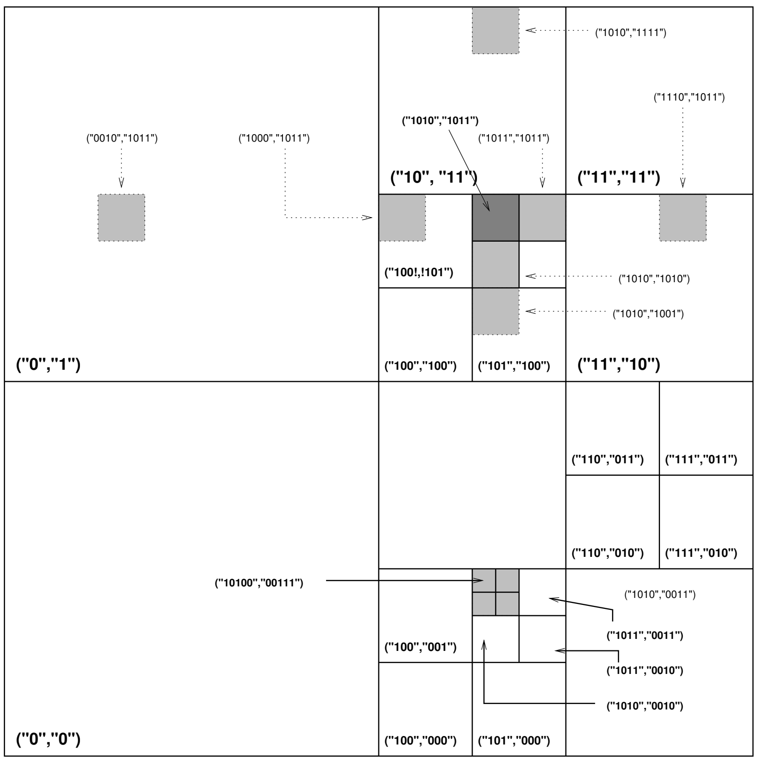

Fig. 3. *CoRSO*: a concrete example of the tree of search *districts*. Thick borders and bold strings identify existing leaf-districts, hatched districts show the neighborhood of district (1010,1011).

children, as soon as two different local minima are found in it. Because the subdivision process is triggered by the local properties of f, after some iterations of *CoRSO* the tree will be of varying depth in the different regions, with districts of smaller sizes being present in regions that require an intensification of the search. Only the *leaves* of the tree are admissible search points for the combinatorial component of *CoRSO*. The leaves partition the initial region: the intersection of two leaves is empty, the union of all leaves coincides with the initial search space. A typical configuration for a two-dimensional task is shown in Fig. 3, where each leaf-district is identified by thick borders and a bold binary string.

Each existing district for a problem of dimension N is identified by a unique binary string B_S with $n \times N$ bits: $B_S = [g_{11}, ..., g_{1n}, ..., g_{N1}, ..., g_{Nn}]$. The value n is the depth of the district in the tree: $n = 0$ for the *root* district, $n = 1$ for the leaves of the initial tree (and therefore the initial string has N bits), n increases by one when a given district is subdivided. The length of the district edge along the i-th coordinate is therefore equal to $(B_{Ui} - B_{Li})/2^n$. The position of the district origin B_{Oi} along the i-th coordinate is

$$B_{Oi} = B_{Li} + (B_{Ui} - B_{Li}) \sum_{j=1}^{n} \frac{g_{ij}}{2^j}.$$

The evaluated neighborhood of a given district consists only of existing leaf-districts: no new districts are created during the neighborhood evaluation. Now, after applying the elementary moves to the identifying binary string B_S of a given district B, one obtains $N \times n$ districts of the same size distributed over the search space as illustrated in Fig. 3, for the case of $B_S = (1010, 1011)$. Because the tree can have different depth in different regions, it can happen that some of the obtained strings do *not* correspond to leaf-districts, others can cover *more* than a single leaf-district. In the first case one evaluates the smallest *enclosing* leaf-district, in the second case one evaluates a randomly-selected *enclosed* leaf-district. The random selection is executed by generating a point with uniform probability in the original district, and by selecting the leaf that contains the point. This assures that the probability for a leaf to be selected is proportional to its volume.

4.1. Evaluating opportunities for the different districts. While the RSO algorithm for combinatorial optimization generates a search trajectory consisting of points $X^{(t)}$, *CoRSO* generates a trajectory consisting of *leaf-districts* $B^{(t)}$. There are two important changes to be underlined: firstly, the function $f(X)$ must be substituted with a routine measuring the potential that the current district contains good local optima, secondly, the tree is *dynamic* and the number of existing districts grows during the search.

The combinatorial component must identify promising districts quickly. In the absence of detailed models about the function f to be minimized, a simple evaluation of a district B can be obtained by generating a point X with a uniform probability distribution inside its region and by evaluating the function $f(X)$ at the obtained point. Let us use the same function symbol, the difference being evident from its argument: $f(B) \equiv f(rand\ X \in B)$. The potential drawback of this simple evaluation is that the search can be strongly biased in favor of a district in the case of a "lucky" evaluation (e.g., $f(X)$ close to the minimum in the given district), or away from a district in the opposite case. To avoid this drawback, when a district is encountered again during the search, a *new* point X is generated and evaluated and some collective information is returned. The value $f(B)$ returned is then the *minimum* of the evaluated X_i: $f(B) \equiv \min_i\{f(X_i)\}$.

Let us consider the example of Fig. 3. The current district (1010,1011) has the neighbors shown with a hatched pattern. The neighbor (0010,1011) in the upper left part is not an existing leaf-district, it is therefore transformed into the enclosing existing leaf-district (0,1). *Vice versa* the neighbor (1010,0011) in the lower right part contains four leaves, one of them (10100,00111) is the output of

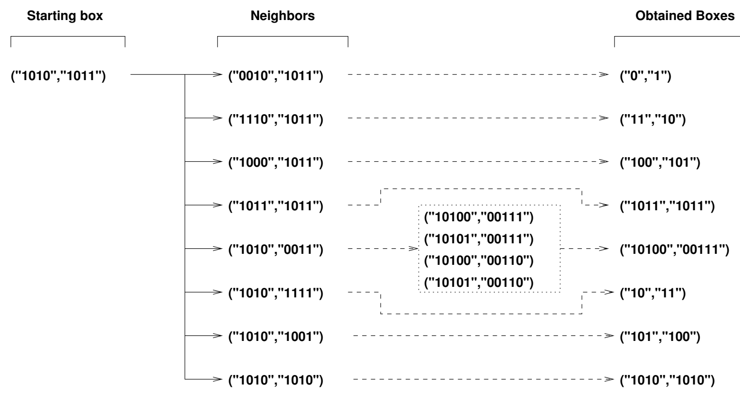

Fig. 4. *CoRSO*: Evaluation of the neighborhood of district (1010,1011).

a random selection. Fig. 4 specifies the complete final neighborhood obtained for the given example.

4.2. Decision about activating Local Search in a given region. According to the RSO dynamics the neighborhood districts obtained starting from a current district are evaluated only if the corresponding basic move from the current point is not prohibited. Only if the evaluation $f(B^{(t)})$ of the current district is less than all evaluations executed in the neighborhood, a decision is taken about the possible triggering of the Local Search component (the Inertial Shaker). In other words, a necessary condition for activating high-precision and expensive searches with Local Search is that there is a high plausibility — measured by $f(B)$ — that the current region can produce local minima that are better with respect to the given neighborhood of candidate districts. Given the greedy nature of the combinatorial component, the current district $B^{(t)}$ on the search trajectory moves toward non-tabu locally optimal districts, therefore it will eventually become locally optimal and satisfy the conditions for triggering IS. Let us note that, if a given district B loses the above contest (i.e., it is not locally optimal for RSO), it *maintains* the possibility to win when it is encountered again during the search, because the evaluation of a different random point X can produce a better $f(B)$ value. Thanks to the evaluation method *CoRSO* is *fast in optimal conditions*, when the f surface is smooth and $f(B)$ is a reliable indicator of the local minimum that can be obtained in region B, but it is *robust in harder cases*, when the $f(B)$ values have a high standard deviation or when they are unreliable indicators of good local minima obtainable with the Inertial Shaker.

The local optimality of the current district B is necessary for activating IS but it is not sufficient, unless B is locally optimal for the first time, a case in which IS is always triggered. Otherwise, if $r > 1$ is the number of times that district B has been locally optimal during the search, an additional IS run must be justified by a sufficient probability to find a *new* local minimum in B. Bayesian rules to estimate the probability that all local optima have been visited can be applied in the context of a single district, where a multi-start technique is realized with repeated activations of IS from uniformly distributed starting points. Because of our splitting criterion, at most one local optimum will be associated to a given district (a district is split as soon as two different local optima are found, see Section 4.3). In addition, some parts of the district can be such that IS will exit the borders if the initial point belongs to these portions. One can therefore partition the district region into W components, the attraction basins of the local minima contained in the district and a possible basin that leads IS outside, so that the probabilities of the basins sum up to one ($\sum_{w=1}^{W} P_w = 1$).

According to (Boender and Rinnooy Kan, 1983), if $r > W + 1$ restarts have been executed and W different cells have been identified, the total relative volume of the "observed region" (i.e., the posterior expected value of the relative volume Ω) can be estimated by

$$E(\Omega|r, W) = \frac{(r - W - 1)\,(r + W)}{r\,(r - 1)}\;; \qquad\qquad r > W + 1. \tag{1}$$

The Inertial Shaker is always triggered if $r \leq W+1$, because the above estimate is not valid in this case, otherwise the IS is executed again with probability equal to $1 - E(\Omega|r, W)$. In this way, additional runs of IS tend to be spared if the above estimate predicts a small probability to find a new local optimum, but a new run is never completely prohibited for the sake of robustness: it can happen that the Bayesian estimate of equation (1) is unreliable, or that the unseen portion $(1 - E(\Omega|r, W))$ contains a very good minimum with a small attraction basin.

The initial conditions for IS (described in Fig. 6) are that the initial search point is extracted from the uniform distribution inside B, the initial search frame is $\vec{b}_i = \vec{e}_i \times (1/4) \times (B_{Ui} - B_{Li})$ where $\vec{e}_i$ are the canonical basis vectors of $\mathbb{R}^N$. The Inertial Shaker generates a trajectory that must be contained in the district B enlarged by a border region of width $(1/2) \times (B_{Ui} - B_{Li})$, and it must converge to a point contained in B. If IS exits the enlarged district or the root-district, it is terminated, the function evaluations executed by IS are discarded. If it converges to a point outside the original district but inside the enlarged district, the point location is saved. In both cases the *CoRSO* combinatorial component continues in the normal way: the next district $B^{(t+1)}$ is the best one in the admissible neighborhood of $B^{(t)}$. In any case the "best so far" value is always updated by considering all admissible points evaluated (those that are inside of the root-district).

A possible exception to the normal *CoRSO* evolution can happen only in the event that IS converges inside $B^{(t)}$ to a local minimum X_l. If X_l is the first local minimum found, it is saved in a memory structure associated to the district. If a local minimum Y_l was already present, and X_l corresponds to the same point, it is discarded, otherwise the current district is *split* until the "siblings" in the

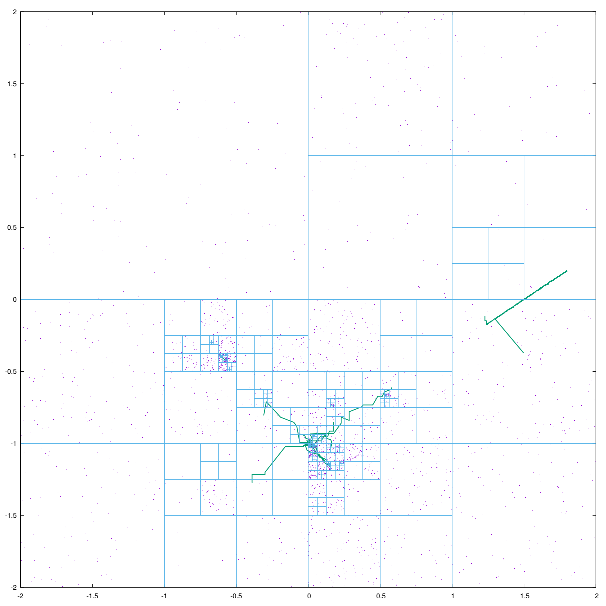

Fig. 5. A *CoRSO* tree structure adapted to a fitness surface. boxes (thin lines), evaluated points (dots) and LS trajectories (thick lines).

tree divide the two points. After the splitting is completed, the current district $B^{(t)}$ does not correspond to an existing leaf anymore: to restore legality a point is selected at random with uniform distribution in $B^{(t)}$ and the legal $B^{(t)}$ becomes the leaf-district that contains the random point. Therefore each leaf-district in the partition of the initial district has a probability of being selected that is proportional to its volume. The splitting procedure is explained in the following section.

4.3. Adapting the district area to the local fitness surface. As soon as two different local minima X_l and Y_l are identified in a given district B, the current district is subdivided into 2^N equal-sized boxes. If X_l and Y_l belong to two different leaf-districts of the new partition, the splitting is terminated, otherwise the splitting is applied to the district containing X_l and Y_l, until their separation is obtained.

In all cases the old district ceases to exist and it is substituted with the collection obtained through the splitting. The local minima X_l and Y_l are associated with their new boxes. Numerically, the criterion used in the tests for considering *different* two local minima X_l and Y_l is that $\|X_l - Y_l\| < \epsilon$, where ϵ is a user-defined precision requirement.

All local minima identified are saved and reported when *CoRSO* terminates.

An example of the tree structure produced during a run of *CoRSO* is shown in Fig. 5, for the case of a two-dimensional function (the Goldstein-Price function also used in experiments, see Sec. 5). The local optima are clearly visible as the zones in which boxes are recursively broken up, while the global optimum is in $\boldsymbol{x} = (0, -1)$. One notices that the points evaluated (the points used to calculate $f(B)$) are distributed quasi-uniformly over the search space: this is a result of the volume-proportional selection, and it guarantees that all regions of the search space are treated in a fair manner. The IS trajectories either converge to a local minimum or are terminated when they exit from the enlarged district, as explained in Section 4.2. Because of our splitting criterion, each district contains at most one local minimum. Although it is not visible from the figure, most points (about 85% in the example) are evaluated during the local search phases, that are the most expensive parts of the *CoRSO* algorithm.

We underline that *CoRSO* is a methodology to integrate a local search component with a strategic allocation and sizing of districts and the generation of starting points in a multi-dimensional space. Feel free to experiment with different local search components, or different details about splitting the original space and firing local searches.

4.4. Local Search with the Inertial Shaker (IS). RAS, the previous default algorithm used in C-RTS (Battiti and Tecchiolli, 1996), requires matrix-vector multiplications to update the search region, and therefore slows down when the number of dimensions becomes very large. The simpler *Inertial Shaker* (IS) technique (Battiti and Tecchiolli, 1994) can be a more effective choice in this case: the search box is always identified by vectors parallel to the coordinate axes (therefore the search box is defined by a single vector $\boldsymbol{\beta}$ and no matrix multiplications are needed) and a *trend direction* is identified by averaging the d previous displacements where d is the domain's dimensionality. An important parameter of the IS heuristic is the amplification coefficient defined as $f_{\text{ampl}} > 0$ controls the extent at which the trend direction modifies the search box.

RAS requires matrix-vector multiplications to update the search region, and therefore slows down when the number of dimensions becomes very large. The simpler *Inertial Shaker* (IS) technique (Battiti and Tecchiolli, 1994), outlined in Fig. 6 can be a more effective choice in this case: the search box is always identified by vectors parallel to the coordinate axes (therefore the search box is defined by a single vector $\boldsymbol{\beta}$ and no matrix multiplications are needed) and a *trend direction* is identified by averaging a number of previous displacements: the *find_trend* function used at line 7 simply returns a weighted average of the previous displacements:

$$\boldsymbol{\delta}_t = f_{\text{ampl}} \cdot \frac{\displaystyle\sum_{u=1}^{d} \boldsymbol{\delta}_{t-u} e^{-\frac{u}{h_{\text{decay}}^2}}}{\displaystyle\sum_{u=1}^{d} e^{-\frac{u}{h_{\text{decay}}^2}}},$$

where the amplification coefficient f_{ampl} and the history decay factor h_{decay} are defined in the algorithm; the number of past displacements in the weighted average

f	(input)	Function to minimize
$\boldsymbol{x}$	(input)	Initial and current point
$\boldsymbol{\beta}$	(input)	Box defining search region $\mathcal{R}$ around $\boldsymbol{x}$
$\boldsymbol{\delta}$	(parameter)	Current displacement
amplification	(parameter)	Amplification factor for future displacements
history_depth	(parameter)	Weight decay factor for past displacement average

```
1.  function InertialShaker (f, $\boldsymbol{x}$, $\boldsymbol{\beta}$)
2.      $t \leftarrow 0$
3.      repeat
4.          success ← double_shot_on_all_components ( $\boldsymbol{\delta}$)
5.          if success = true
6.              $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\delta}$
7.              find_trend ( $\boldsymbol{\delta}$)
8.              if f( $\boldsymbol{x} + \boldsymbol{\delta}$) < f( $\boldsymbol{x}$)
9.                  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \boldsymbol{\delta}$;
10.                 increase amplification and history_depth
11.             else
12.                 decrease amplification and history_depth
13.     until convergence criterion is satisfied
14.     return $\boldsymbol{x}$;
```

Fig. 6. The Inertial Shaker algorithm, from (Battiti and Tecchiolli, 1994).

is chosen to be equal to the function's dimensionality d in order to cut off negligible exponential weights and to keep the past history reasonably small.

Fig. 7 shows how the double-shot strategy is applied to all components of the search position $\boldsymbol{x}$. A displacement is applied at every component as long as it improves the result. If no improvement is possible, then the function returns `false`, and the search box is accordingly shrunk.

4.4.1. On the importance of local search in CoRSO. The district separation mechanism of *CoRSO* can be used to recursively divide the function's domain into smaller and smaller regions, therefore isolating local minimum points, without the need for a local search algorithm; every time a district is evaluated to be a local minimum, instead of executing an IS run starting from the district we simply evaluate a new point and proceed at splitting the district as if a new local minimum had been found.

The four charts in Fig. 8 show the effect of "switching off" the local search component on two functions. The Rastrigin benchmark function used in the top charts is multi-modal (it actually contains a grid of local minima, gradually decreasing towards the global one at $\boldsymbol{x} = (0, 0)$). In the 10^5 evaluations allotted for the test, both tests (with IS on the left, without on the right) correctly identify the global minimum, although the version without local search finds it later and

f	Function to minimize
$\boldsymbol{x}$	Current position
$\boldsymbol{\beta}$	Vector defining current search box
$\boldsymbol{\delta}$	Displacement

```
function double_shot_on_all_components (f, $\boldsymbol{x}$, $\boldsymbol{\beta}$, $\boldsymbol{\delta}$)
    $success \leftarrow$ false
    $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x}$
    for $i \in \{1, \dots, n\}$
        $E \leftarrow$ f( $\hat{\boldsymbol{x}}$)
        r $\leftarrow$ random in $[-b_i, b_i]$
        $\hat{x}_i \leftarrow \hat{x}_i + r$
        if $f(\hat{\boldsymbol{x}}) > E$
            $\hat{x}_i \leftarrow \hat{x}_i - 2r$
            if $f(\hat{\boldsymbol{x}}) > E$
                $b_i \leftarrow \rho_{\mathrm{comp}} b_i$
                $\hat{x}_i \leftarrow \hat{x}_i + r$
            else
                $b_i \leftarrow \rho_{\mathrm{exp}} b_i$
                $success \leftarrow$ true
        else
            $b_i \leftarrow \rho_{\mathrm{exp}} b_i$
            $success \leftarrow$ true
    if $success =$ true
        $\boldsymbol{\delta} \leftarrow \hat{\boldsymbol{x}}$ - $\boldsymbol{x}$
    return $success$
```

Fig. 7. The double-shot strategy from (Battiti and Tecchiolli, 1994): apply a random displacement within the search box to all coordinates, keep the improving steps; return `false` if no improvement is found.

with much more approximation.

The same can be said about the Rosenbrock function (bottom charts of Fig. 8); the function is unimodal, with a deep parabolic valley that gets progressively narrow towards the global minimum in $\boldsymbol{x} = (1, 1)$ (i.e., away from the paraboloid's vertex). The bottom of the valley can be traced both by the IS trajectories in the left picture and by the finer districts in the right chart.

In both cases, the version without local search is much slower, due to the need to maintain the district tree data structure with multiple subdivisions, and becomes prohibitively slow for $d > 3$. For this reason, and for the higher precision attained by allowing the local search phase, we can conclude that local search is indeed a fundamental component in the proposed heuristic.

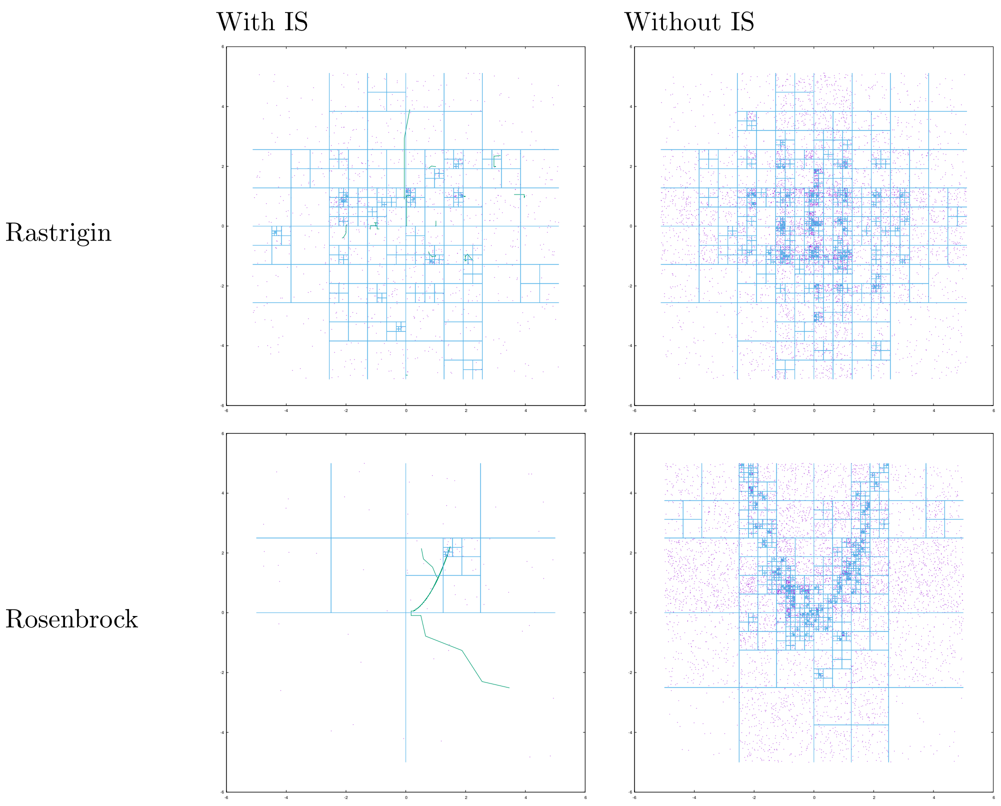

Fig. 8. *CoRSO* tree structure with (left) and without (right) the local search component, applied to the 2-dimensional Rastrigin (top) and Rosenbrock (bottom) function.

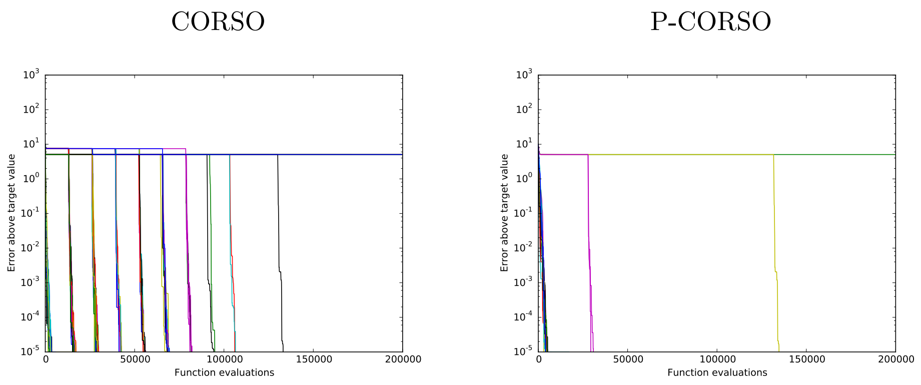

Fig. 9. Optimization history traces of all runs of *CoRSO* and its portfolio version *P-CoRSO* on the order-5 Shekel function. The record value is plotted against the number of evaluations.

4.4.2. P-CoRSO: A portfolio of CoRSO optimizers As a preliminary test, the left part of Fig. 9 tracks the execution of 150 runs of *CoRSO* on the order-5 Shekel function (one of the benchmark functions used for testing, as described in Sec. 5). For every run, the record value is plotted against the number of function evaluations. In the top left, we can observe the interaction between the combinatorial component and the Inertial Shaker. In some runs, the initial exploration by the Inertial Shaker gets stuck in a local minimum; in such case, we need to wait until the IS termination condition is met before the combinatorial component can search a more promising region and restart the local search phase. Once a good starting point has been found, convergence to the global minimum is quite fast, but the number of restarts needed can be high and unfortunate searches can exhaust the evaluation budget.
This problem can be mitigated by maintaining a portfolio of *CoRSO* optimizers.

The *P-CoRSO* (Portfolio of *CoRSO*) technique works by initially maintaining a set of 5 *CoRSO* searchers. In this initial phase, these searchers are executed in a round-robin fashion until 1% of the evaluation budget is consumed. At this point, the remaining iterations are reserved to the searcher having found the best objective value. The *P-CoRSO* heuristic is shown in the right chart of Fig. 9. Some unlucky cases are still possible, but the overwhelming majority of the runs converges at the first try.

5. Experimental results.

In this section we describe the experimental results obtained by comparing *CoRSO*, and its portfolio version *P-CoRSO*, with other state-of-the-art population-based meta-heuristics for continuous optimization:

- The Comprehensive Learning Particle Swarm Optimizer (CLPSO) (Liang *et al.*, 2006) is a PSO-based heuristic that prevents premature convergence by using the historical best information of all particles[1]. The code is implemented in MATLAB.
- Differential Evolution (DE) (Price *et al.*, 2006) is a Genetic Algorithm based on adding the difference between two population members to a selected vector before crossing over, so that "no separate probability distribution has to be used which makes the scheme completely self-organizing"[2]. The C version has been used in this paper.

The *CoRSO* code described in this paper has been implemented in C++.

5.1. Benchmark functions.

A subset of classical benchmark functions from the GenOpt challenge[3] has been used for testing, in particular the unimodal Rosenbrock ($d = 10, 30$), Sphere ($d = 10, 30$) and Zakharov ($d = 10, 30$) function families, and the multi-modal Goldstein-Price ($d = 2$), Hartmann ($d = 3, 6$), Rastrigin ($d = 10, 30$), and Shekel ($d = 4$, order $= 5, 7, 10$) families.

[1]Code available at `http://www.ntu.edu.sg/home/epnsugan/`.
[2]Code available at `http://www1.icsi.berkeley.edu/~storn/code.html`.
[3]Available at `http://genopt.org/`.

5.2. Settings. All tests have been carried out on an 8-core 2.33GHz Intel Xeon server running a 64-bit Linux OS with kernel 3.13.0. However, due to the large performance gap between the MATLAB implementation of CLPSO and the other algorithms, all comparisons are based on the number of function evaluations rather than on physical time.

The relevant parameters of the *CoRSO* optimizer have been empirically tested on short runs on the 5-dimensional Rastrigin and Rosenbrock functions ($d = 5$ is not used in the subsequent tests). The tabu period reduction factor, to which the technique has proved to be very little sensitive, has been set to $f_{\text{red}} = 0.7$; the prohibition period is divided by f_{red} whenever 3 boxes are re-visited 3 times. The amplification factor of the Inertial Shaker has been set to $f_{\text{ampl}} = 0.99$. The Inertial Shaker phase is stopped whenever the improving step size in the domain is shorter than $10^{-8} \cdot \sqrt{d}$, where d is the domain's dimensionality.

Following the authors' suggestions, the population size for the CLPSO algorithm has been set to 10 for lower-dimensional functions ($d \leq 10$), and to 40 for $d = 30$, while the DE parameters are: $\text{NP} = \max\{10 \cdot d, 40\}$, $\text{F} = 0.8$, $\text{CR} = 0.9$.

In the subsequent tests, the main objective is to reach a pre-defined error $\delta = 10^{-5}$ with respect to the function's known global minimum, recording the number of iterations required to achieve it. This kind of task is more representative of many real-world situations in which a tolerance is given on the process to be optimized, and the objective function is often noisy, so that arbitrarily low errors are not realistic. The chosen error value is low enough to ensure that the algorithm has correctly chosen the global minimum's attraction basin in the case of the multi-modal benchmark functions.

5.3. Comparison and discussion. The three meta-heuristics have been run for at least 50 times on each benchmark function. Every run was interrupted at the earliest of two occurrences: (i) an evaluation of the objective function produced a value below $f_{\min} + \delta$, where $f_{\min}$ is the global minimum value of the function, or (ii) the total budget of $2 \cdot 10^5$ evaluations was consumed.

The *CoRSO* heuristic was run 150 times for each benchmark function; with this larger number of runs, the portfolio version could be simulated by aggregating 5 traces at a time, yielding 30 simulated *P-CoRSO* runs.

Table 1 reports the median number of evaluations required by each algorithm to achieve the target objective value. For every benchmark function, the lowest number is highlighted in boldface. As a measure of confidence in the winner, the underlined values are those in which the inter-quartile ranges of the winner and of the runner-up don't intersect (i.e., the third quartile of the winning algorithm's runs is still below the first quartile of the second best algorithm).

CoRSO achieves good results in high-dimensional cases such as the Rastrigin, Sphere and Zakharov function families (although in the 10-D Zakharov case DE achieves a better median result) and in the Hartmann functions. DE works better in most low-dimensional cases, in particular on the Goldstein-Price function and

Table 1. Median iterations for 30+ runs of *CoRSO*, *P-CoRSO*, CLPSO, DE, to get within 10^{-5} from the global minimum. Boldface numbers indicate the best median; underlined if the IRQs don't intersect (see explanation in the text). No result if technique does not reach the desired approximation within the allotted number of function evaluations.

Function	Dim	CORSO	P-CORSO	CLPSO	DE
Goldstein-Price	2	5276	2814	4021	$\underline{\mathbf{716}}$
Hartmann	3	$\underline{\mathbf{804}}$	2253	3291	1172
Hartmann	6	**1847**	3229	13306	—
Rastrigin	10	**10190**	11569	51241	—
Rastrigin	30	94401	**64300**	143952	—
Rosenbrock	10	80852	82159	—	$\underline{\mathbf{29862}}$
Rosenbrock	30	(20%)	(3%)	—	—
Shekel 5	4	28127	4642	74757	**4283**
Shekel 7	4	40419	4843	30909	$\underline{\mathbf{4006}}$
Shekel 10	4	42972	5060	26264	**4321**
Sphere	10	$\underline{\mathbf{2964}}$	4366	15615	6044
Sphere	30	**12174**	13797	61440	—
Zakharov	10	18992	20181	28950	**16254**
Zakharov	30	**172276**	173163	—	—

on the Shekel family. Unfortunately, DE is very fragile and it fails completely to reach the desired approximation in the allotted budget in five cases for which the median results are still very far.

While the portfolio version *P-CoRSO* only wins in one case (30-dimensional Rastrigin), its capability of quickly identifying unfortunate situations put it on par with DE on the Shekel family, where it achieves a dramatic reduction with respect to the *CoRSO* heuristic alone.

It is also worth noting that, while DE can be very fast on certain function classes, it is significantly underperforming in many other cases, where no runs achieve the required target value within the allotted number of iterations. This is possibly due to sensitivity to parameters, which (as a general rule) were kept constant through all benchmark functions. The *CoRSO* heuristic achieved the local minimum for all functions for at least a fraction of the runs.

Fig. 10 shows a more detailed comparison on four benchmark functions. In the top two charts, referring to the Shekel 7 and Shekel 10 functions, the success of the portfolio version *P-CoRSO* in lowering the median value of *CoRSO* is apparent. However, in both cases we can see a heavy tail in the form of outliers (Shekel 7) or a large IQR (Shekel 10). The bottom charts show two cases in which the portfolio heuristic is ineffective in lowering the median.

As a specific example, let us follow the evolution in time of the algorithms on the 3-dimensional Hartmann function, whose final outcomes are represented in the bottom-right chart of Fig. 10. In Fig. 11 the median value of all the runs

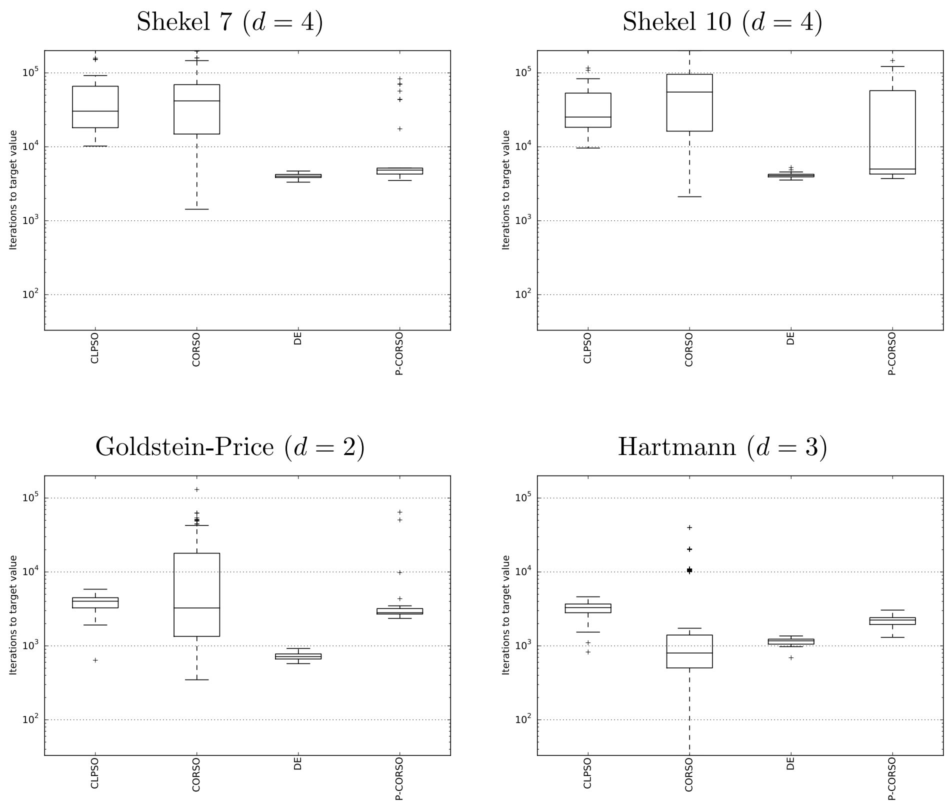

Fig. 10. Distribution of the number of iterations required to achieve the target value for the four algorithms on four benchmark functions.

of each algorithm on the 3-dimensional Hartmann function is plotted against the number of evaluations. In order to better appreciate the variability of the single runs, error bars represent the interval between the first and the third quartile, so that 50% of the runs actually pass within the error bar.

Fig. 12 shows the individual traces of all runs of the four algorithms on the same benchmark function. The charts have the same scale as in Fig. 11. In this figure, we can appreciate in particular the "outlier" *CoRSO* runs that do not converge immediately, but need to be restarted from a better position. The same outliers are visible in the bottom-right chart of Fig. 10 on the *CoRSO* column. The relationship between *CoRSO* and *P-CoRSO* is explained by looking at the top-left and top-right traces shown in Fig. 12. While the unlucky runs are removed by the portfolio mechanism, the evaluation overhead introduced by the initial round-robin evaluation delays the earliest successful runs by reducing their downward slope. The net result is an increase of the median number of evaluations from 804 to 2253, even though outlier runs are eliminated.

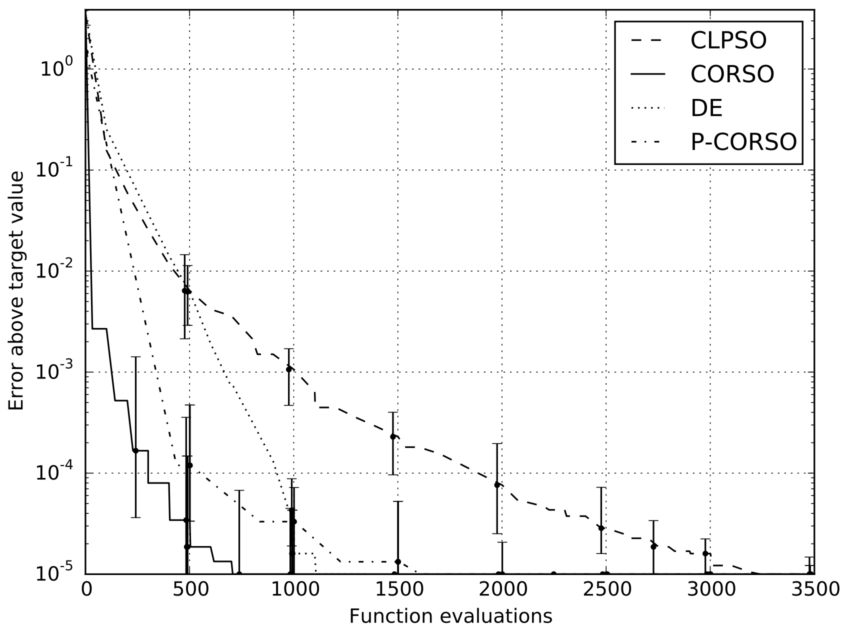

Fig. 11. Optimization history of the considered meta-heuristics on the 3-dimensional Hartmann function. The median record value of all runs is plotted against the number of evaluations. Inter-quartile ranges are shown as error bars. Only the initial 3500 iterations are shown.

6. Conclusions. Many optimization problems of real-world interest are complex and need enormous computing times for their solution. The use of many computers working in parallel (maybe living in the cloud, rented when they are needed) comes to the rescue to reduce the clock time to produce acceptable solutions. In some cases, one can consider *independent search streams*, periodically reporting the best solutions found so far to some central coordinator. In other cases, more *intelligent schemes of coordination* among the various computers lead to a higher efficiency and effectiveness.

The *CoRSO* algorithm presented in this paper and its portfolio version *P-CoRSO* deal with coordinating a team of interacting solvers through an organized subdivision of the configuration space which is adapted in an online manner. The main building block is local search, acting in two different levels. Combinatorial local search with reactive prohibitions (RSO) manages an adaptive tree of search districts and generates a trajectory of search boxes, biased towards more promising areas. Continuous stochastic local search (Inertial Shaker) is started in a frugal manner only when the current search district looks sufficiently promising.

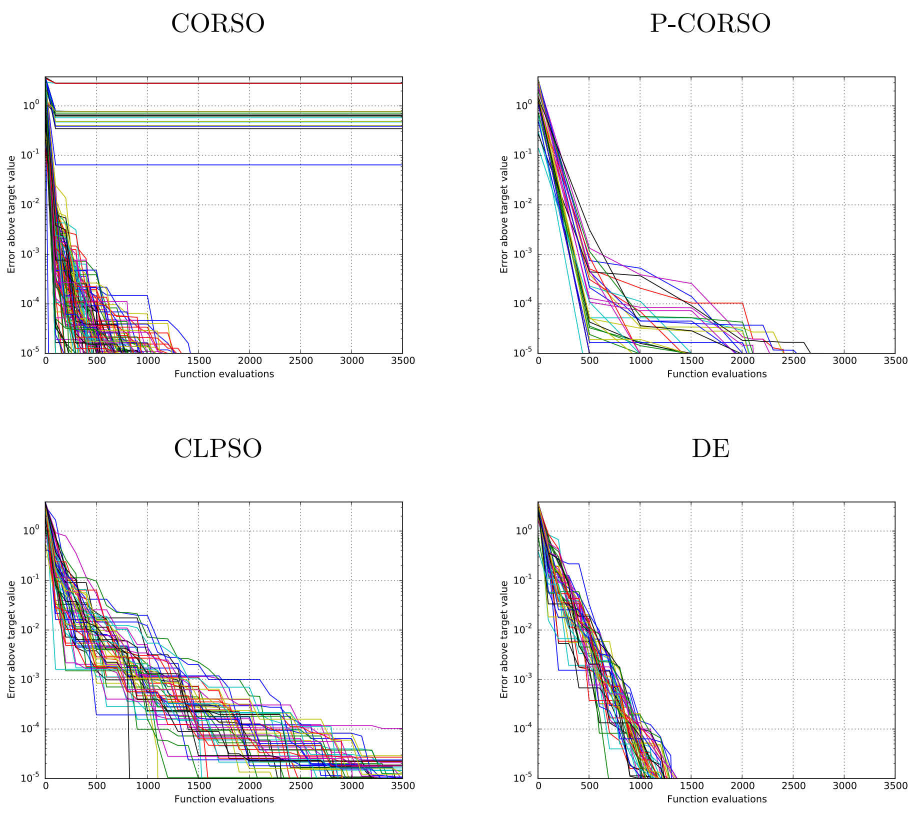

Fig. 12. Traces of all runs of the four algorithms on the 3-dimensional Hartmann function. Only the initial 3500 iterations are shown.

The experimental results of *CoRSO* and *P-CoRSO*, when compared with widely used genetic and particle swarm algorithms, demonstrates either better median results for some problems or a greater robustness level. In particular, the competing techniques appear to be stuck with values very far from the global optimum for some instances, within the allotted budget. In some cases DE and CLPSO do not reach the desired approximation even if the budget is increased by an order of magnitude.

As a future step it would be of high scientific and practical interest to compare *CoRSO* with different strategies like the *P*-algorithm with simplicial partitioning or Divide-the-Best algorithm based on Lipschitz assumptions and on efficient diagonal partitions. Possible adaptive hybrids which determine in an online manner the best methods (or the best combination) to use based on preliminary results are also of high scientific and practical interest. A possible context can be that of scientific competitions like `genopt.org`.

Our experimental evidence supports the hypothesis that paradigms derived from human organizations, characterized by "learning on the job" capabilities, can lead to superior or more robust results with respect to paradigms derived from simpler living organisms or genetic principles.

Acknowledgments The research of Roberto Battiti was supported by the Russian Science Foundation, project no. 15–11–30022 "Global optimization, supercomputing computations, and applications."

REFERENCES

Battiti, R., Brunato, M., Mascia, F. (2008). *Reactive Search and Intelligent Optimization.* Vol. 45 of *Operations research/Computer Science Interfaces.* Springer Verlag.

Battiti, R., Tecchiolli, G. (1994). Learning with First, Second, and no Derivatives: a Case Study in High Energy Physics. *Neurocomputing*, 6, 181–206.

Battiti, R., Tecchiolli, G. (1996). The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization. *Annals of Operations Research – Metaheuristics in Combinatorial Optimization*, 63, 153–188.

Boender, C., Rinnooy Kan, A. (1983). A Bayesian Analysis of the Number of Cells of a Multinomial Distribution. *The Statistician*, 32, 240–249.

Gutmann, H.M. (2001). A radial basis function method for global optimization. *Journal of Global Optimization*, 19(3), 201–227.

Krasnogor, N., Smith, J. (Oct 2005). A tutorial for competent memetic algorithms: model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5), 474–488.

Liang, J.J., Qin, A.K., Suganthan, P.N., Baskar, S. (2006). Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *Evolutionary Computation, IEEE Transactions on*, 10(3), 281–295.

Moscato, P. (1989). On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech Concurrent Computation Program, C3P Report*, 826.

Price, K., Storn, R.M., Lampinen, J.A. (2006). *Differential evolution: a practical approach to global optimization.* Springer Science & Business Media.

Sergeyev, Y.D., Kvasov, D.E. (2015). A deterministic global optimization using smooth diagonal auxiliary functions. *Communications in Nonlinear Science and Numerical Simulation*, 21(1), 99–111.

Törn, A., Viitanen, S. (1992). Topographical global optimization. *Recent advances in global optimization*, 384–398.

Törn, A., Žilinskas, A. (1989). *Global optimization.* Springer-Verlag Berlin Heidelberg.

Zhigljavsky, A., Žilinskas, A. (2007). *Stochastic global optimization.* Vol. 9. Springer Science & Business Media.

Žilinskas, A. (1985). Axiomatic characterization of a global optimization algorithm and investigation of its search strategy. *Operations Research Letters*, 4(1), 35–39.

Žilinskas, A., Žilinskas, J. (2002). Global optimization based on a statistical model and simplicial partitioning. *Computers & Mathematics with Applications*, 44(7), 957–967.