Computers & Operations Research 43 (2014) 169–180

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Differential evolution methods based on local searches

Marco Locatelli [a], Mirko Maischberger [b], Fabio Schoen [b,*]

[a] Dip. Ingegneria dell'Informazione, Univ. di Parma, Viale G.P.Usberti 181/A, 43100 Parma, Italy
[b] Dip. Ingegneria dell'Informazione, Univ. di Firenze, via di Santa Marta, 3, 50139 Firenze, Italy

ARTICLE INFO

Available online 26 September 2013

Keywords:
Global optimization
Population-based methods
Memetic algorithms
Disk and sphere packing

ABSTRACT

In this paper we analyze the behavior of a quite standard Differential Evolution (DE) algorithm applied to the objective function transformed by means of local searches. First some surprising results are presented which concern the application of this method to standard test functions.

Later we introduce an application to disk- and to sphere-packing problems, two well known and particularly hard global optimization problems. For these problems some more refined variations of the basic method are necessary in order to take at least partially into considerations the many symmetries those problems possess. Coupling these techniques with DE and local optimization resulted in a new method which, when tested on moderately sized packing problems, was capable of confirming known putative optima for the problem of packing disks, and of discovering quite a significant number of new putative optima for the problem of packing spheres.

1. Introduction and problem statement

Differential Evolution (DE) is a very simple population-based global optimization algorithm. If we denote by $f:\mathbb{R}^n\to\mathbb{R}$ the objective function of the problem, the original DE method (see [1]), can be described as in Algorithm 1, where:

- $\mathcal{U}(S)$ represents a uniform random number generator in the set S;
- p is the population size;
- $\{x_1,\ldots,x_p\}\subset\mathbb{R}^n$ is the population;
- $x_i^{(j)}$ denotes the j-th element of vector $x_i\in\mathbb{R}^n$.

The idea of the algorithm can be easily seen within the context of genetic algorithms:

- the algorithm maintains and evolves a whole population of solutions;
- a sort of *crossover* can be performed (if $CR>0$), where a new individual is formed by substituting some of its components with a linear combination of three other members in the population;
- population overall improvement is implemented through an acceptance criterion by which a new trial solution is accepted and replaces a current individual if the associated function value improves.

The method reported in Algorithm 1 is one of the possible variants of DE; different implementations of the basic scheme can be identified thanks to a simple taxonomy which takes the form of *DE/x/y/z*. Here *x*, which takes the values *rand* or *best*, represents the criterion by which the solution to be perturbed is chosen (a solution chosen at random using a uniform distribution, or the current best member of the population). The second field, *y*, stands for the number of difference vectors used in the perturbation of *x*, where a difference vector is the difference between two distinct randomly selected elements in the population. Finally, *z* identifies the crossover operator—its value represents the discrete probability distribution used to generate the individuals which participate to the crossover; its value can be *bin* indicating that the choice is made following a, binomial probability distribution function, as in the scheme reported in Algorithm 1, or *exp* when the choice on which components to keep from the current population is based on an exponential distribution. The strategy outlined in Algorithm 1 can be classified as DE/rand/1/bin. We refer to the existing literature for a description of its many variants (see, in particular, [2]).

Algorithm 1. Differential evolution.

Data: $F\in(0,2)$, a real constant; $CR\in[0,1]$, a probability threshold

* Corresponding author. Tel.: +39 055 479 6358.
E-mail addresses: locatelli@ce.unipr.it (M. Locatelli), maischberger@dsi.unifi.it (M. Maischberger), fabio.schoen@unifi.it (F. Schoen).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.09.010

for each $i \in 1,\ldots,p$ **do**
  let $\bar{\imath} := \mathcal{U}(1,\ldots,n)$;
  randomly choose $k_1, k_2, k_3 \in \{1,\ldots,p\}\backslash\{i\}$, all different;
  let $Trial := x_{k_1} + F(x_{k_2} - x_{k_3})$;
  for $j \in 1,\ldots,n : j \neq \bar{\imath}$ **do**
    if $\mathcal{U}(0,1) < CR$ **then**
      let $Trial^{(j)} := x_i^{(j)}$;
    end
  end
  if $f(Trial) < f(x_i)$ **then**
    let $x_i := Trial$;
  end
end

The theoretical aspects of DE have received only moderate attention in the literature. In [3], under mild conditions, convergence of the whole population to a single point is proven for strictly convex objective functions, although such point is not guaranteed to be the local (global) minimizer. In fact, for more general functions some phenomena may occur, like, e.g., the population collapsing to a single point or a frozen population (due to the finite number of possible moves, none of them might be improving with respect to the current members of the population), which prevent further progress of the population and thus the convergence to a global minimizer (see, e.g., [4,5]).

Despite this lack of theoretical support, the practical behavior of the method is quite interesting and the algorithm is used in a very large number of cases.

As it is common in many global optimization heuristics (see, e.g., [6]), the algorithm switches from an initial, exploratory, phase to a local refinement one where members of the population group together. While this behavior is needed in most good heuristics, it should be recalled that, when it comes to local refinement, it is usually much more efficient to base the search on standard local optimization methods, instead of performing local steps without exploiting the power of local descent methods.

In this paper we first explore a quite simple modification of the basic DE scheme, following the approach which, in combinatorial optimization, goes under the name of *memetic*: in Algorithm 1, each time the evaluation of the objective function f is required, we instead perform a local descent by means of a local optimization method $\mathcal{L}$ which, given an objective function f and a starting point x returns $\bar{x} = \mathcal{L}(f,x) \in \mathbb{R}^n$, a local minimizer for f where $f(\bar{x}) \leq f(x)$ for all x in a neighborhood of $\bar{x}$. To be more precise, local optimization methods are often only guaranteed to return a stationary point, which is not necessarily a local minimizer. Since the following discussion is not affected by that, in what follows we will always assume, for the sake of simplicity, that the point returned by a local optimization method is a local minimizer. Note that memetic algorithms proved to be efficient ones as testified, e.g., by the fact that a memetic algorithm (see [7]) won the CEC competition 2010 in large scale global optimization (see http://sci2s.ugr.es/eamhco/cec2010_functions.pdf). The basic algorithm is transformed as represented in Algorithm 2 (MDE).

Although in this algorithm only a single line is changed with respect to Algorithm 1, the overall behavior of the whole method radically changes, as it transforms a continuous, derivative free, method like DE into a combinatorial search in the space of local minima—we remark here that in this implementation all members x_i in the population are local optima of the objective function.

Algorithm 2. Memetic differential evolution (MDE).

Data: $F \in (0,2)$, a real constant; $CR \in [0,1]$, a probability threshold

for each $i \in 1,\ldots,p$ **do**
  let $\bar{\imath} := \mathcal{U}(1,\ldots,n)$;
  randomly choose $k_1, k_2, k_3 \in \{1,\ldots,p\}\backslash\{i\}$, all different;
  let $Trial := x_{k_1} + F(x_{k_2} - x_{k_3})$;
  for $j \in 1,\ldots,n : j \neq \bar{\imath}$ **do**
    if $\mathcal{U}(0,1) < CR$ **then**
      let $Trial^{(j)} := x_i^{(j)}$;
    end
  end
  let $Trial := \mathcal{L}(f, Trial)$;
  if $f(Trial) < f(x_i)$ **then**
    let $x_i := Trial$;
  end
end

The original contribution in this paper can be found in two aspects:

- first, we analyze in a quite systematic way the behavior of Algorithm 2 when applied to quite standard test functions and we comment on the somewhat surprising results (not previously observed in the literature, to the authors' knowledge), for which we give some (tentative) explanations;
- then, after introducing some specific modifications, we show how the method can be applied, with quite unexpected good results, on classical problems of packing disks in a square or spheres in a cube.

The results obtained and described in this paper are, in our opinion, interesting and further research will be surely needed in order to fully understand the capabilities of this method.

The paper is structured as follows. In Section 2 we shortly describe Monotonic Basin Hopping (MBH), a simple but efficient global optimization approach based on local searches, which represents a quite natural basis for comparison for MDE. In Section 3 we describe and interpret some computational experiments with MBH and a basic version of MDE on standard global optimization functions and some, more challenging, variants of these functions. In Section 4 we describe some experiments with MDE on packing problems. For these problems variations of the basic MDE method are required in order to take at least partially into consideration the many symmetries those problems possess and, thus, the many equivalent solutions existing for this kind of problems. Finally, in Section 5 we draw some conclusions.

2. A short introduction to Monotonic Basin Hopping

In order to evaluate the behavior of the proposed algorithm, we compared its performance with a standard global optimization algorithm, Monotonic Basin Hopping, MBH (see, e.g., [6,8,9]), whose general scheme is reported in Algorithm 3. This algorithm is strongly based on the application of local searches. It is an Iterated Local Search method in the space of local optima.

It consists in repeatedly performing a local optimization from a point which is randomly, uniformly, generated in a prescribed neighborhood of the current iterate. Here we choose, as a neighborhood, $D(x,\Delta)$, the hypercube centered at x with edge length equal to 2Δ , but other choices for the shape of the neighborhood, like, e.g., hyperrectangles, spheres or ellipsoids, are possible.

Algorithm 3. Basin Hopping Method.

while *GlobalStoppingRule is false* **do**