Evol. Intel. (2011) 4:17–29
DOI 10.1007/s12065-010-0046-8

SPECIAL ISSUE

Composed compact differential evolution

Giovanni Iacca · Ernesto Mininno · Ferrante Neri

Received: 14 August 2010 / Revised: 24 October 2010 / Accepted: 28 November 2010 / Published online: 24 December 2010

Abstract This paper proposes a novel algorithm for solving continuous complex optimization problems with a relatively low memory consumption. The proposed approach, namely Composed compact Differential Evolution, consists of a set of compact Differential Evolution units which simultaneously search the decision space from various perspectives. A randomization in the virtual population allows the algorithm to behave, on one hand, as a multiple local search with a multi-start logic integrated within it. On the other hand, the compact units communicate among each other by means of a ring topology and propagation of information. More specifically, the most promising elite solutions and scale factor values of each compact unit are migrated to the neighbour unit so that the search of the global optimum is performed. In other words, while each single compact unit performs a local search by exploiting the direction suggested by each elite solution, the entire structure combines the achievement of each local search operation towards the direction of the global search. The proposed algorithm is characterized by a limited memory consumption and is memory-wise equivalent to a population-based algorithm with a small population. Numerical results show that the proposed approach outperforms other compact algorithms and various modern population-based structures.

Keywords Differential evolution · Distributed algorithms · Compact algorithms · Randomization · Scale factor inheritance

1 Introduction

Compact Evolutionary Algorithms (cEAs) have been developed in order to address optimization problems characterized by limited memory resources. This situation is typical for robotics and control problems where a full power computing device may be unavailable due to cost and/or space limitations. For example, in industrial applications, in order to guarantee a quick reaction of the actuators, the optimization algorithm should run directly on a control card instead of a personal computer.

A cEA is an Evolutionary Algorithm (EA) belonging to the class of Estimation of Distribution Algorithms (EDAs), see [1]. Algorithms belonging to this class do not store and process an entire population and all its individuals therein but make use of a statistic representation of the population in order to perform the optimization process. In this way, a much smaller number of parameters must be stored in the memory. Thus, a run of these algorithms requires much less capacious memory devices compared to their correspondent standard EAs.

The first cEA was the compact Genetic Algorithm (cGA) introduced in [2]. The cGA simulates the behavior of a standard binary encoded Genetic Algorithm (GA). In

This research is supported by the Academy of Finland, Akatemiatutkija 130600, Algorithmic Design Issues in Memetic Computing and Tutkijatohtori 140487, Algorithmic Design and Software Implementation: a Novel Optimization Platform. This research is also supported by Tekes—the Finnish Funding Agency for Technology and Innovation, grant 40214/08 (Dynergia).

G. Iacca · E. Mininno · F. Neri (✉)
Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35, Agora 40014, Finland
e-mail: ferrante.neri@jyu.fi

G. Iacca
e-mail: giovanni.iacca@jyu.fi

E. Mininno
e-mail: ernesto.mininno@jyu.fi

[2] can be seen that cGA has a performance (for low dimensional problems) almost as good as that of GA. As expected the main advantage of a cGA with respect to a standard GA is the memory saving. An analysis of the convergence properties of cGA by using Markov chains is given in [3]. In [4] (see also [5]) the extended compact Genetic Algorithm (ecGA) has been proposed. The ecGA is based on the idea that the choice of a good probability distribution is equivalent to linkage learning. The measure of a good distribution is based on Minimum Description Length (MDL) models: simpler distributions are better than the complex ones. The probability distribution used in ecGA is a class of probability models known as Marginal Product Models (MPMs). A theoretical analysis of the ecGA behavior is presented in [6]. A hybrid version of ecGA integrating the Nelder-Mead algorithm is proposed in [7]. A study on the scalability of ecGA is given in [8]. The cGA and its variants have been intensively used in hardware implementation, see [9–11]. A cGA application to neural network training is given in [12].

Paper [13] analyzes analogies and differences between cGAs and (1 + 1)-ES and extends a mathematical model of ES [14] to cGA obtaining useful information on the performance. Moreover, [13] introduces the concept of elitism, and proposes two new variants, with strong and weak elitism respectively, that significantly outperform both the original cGA and (1 + 1)-ES. A real-encoded cGA (rcGA) has been introduced in [15]. Some examples of rcGA applications to control engineering are given in [16] and [17]. A simple real-encoded version of ecGA has been proposed in [18] and [19].

The main drawback of these algorithms is that, since cEAs simulate the population by means of a statistical representation, they are subject to premature convergence, especially when the dimensionality grows. As experimentally shown in [20], a rcGA suffers from premature convergence already for dimensionality values higher than ten. More specifically, the interdependencies among the decision variables (non-separability of the landscape, epistasis) cause the premature convergence of compact algorithms since a rcGA generates a candidate solution by independently sampling a parameter over each dimension. In highly multivariate problems, there is a high number of interactions among variables and thus rcGA can likely fail at detecting solutions with a high performance. In order to improve the performance of cEAs for complex problems, a compact algorithm employing the Differential Evolution (DE) search logic has been proposed in [20]. This algorithm, namely compact Differential Evolution (cDE) exploits the features of DE schemes to generate new candidate solutions and makes use of the knowledge of DE structures to obtain a high performance despite the limited memory usage. In other words, two algorithmic features make cDE superior to the algorithms belonging to the cGA family for a broad set of problems. First, unlike other compact evolutionary algorithms, in cDE, the survivor selection scheme of DE can be straightforwardly encoded. The one-to-one spawning typical of DE is equivalent to a tournament selection having size two. Since compact algorithms require the employment of the latter kind of selection scheme, the compact encoding of a GA heavily jeopardizes the original algorithmic nature. On the contrary, the compact encoding of a DE-based algorithm does not jeopardize the algorithmic logic of the original population-based DE algorithm, see [20]. Second, due to its nature cDE uses an implicit randomization of the offspring generation which corrects and improves the DE search logic. As shown in [21], DE is characterized by a limited amount of search moves and the employment of a proper degree of randomization can significantly improve upon the algorithmic performance.

An improvement of the cDE scheme has been obtained by coupling it, in a memetic fashion, with a local search algorithm, see [22]. The resulting algorithm, namely Memetic compact Differential Evolution (McDE) has been applied for the control of a Cartesian robot manipulator. Although the employment of the cDE structure mitigates the risk of premature convergence, since compact algorithms require mush less resources in terms of memory and hardware with respect to modern population-based algorithms, see e.g. [23], compact algorithms are likely to be outperformed (especially for problems characterized by more than ten dimensions) in terms of final solutions by population-based algorithms. This fact leads to the obvious conclusion that when memory requirements forbid the employment of population-based algorithms, compact algorithms can be a cheap and efficient solution. On the other hand, when there is no hardware limitation, population-based algorithms are preferable.

This paper proposes a compromise solution between the necessity of having a respectable performance and the memory limitations. In other words, this paper proposes an unexplored algorithmic concept in evolutionary intelligence, i.e. the employment of compact optimization algorithms as units of a distributed system. The proposed algorithm is named Composed compact Differential Evolution (CcDE). This algorithm employs multiple cDE cores arranged by means of a ring topology. In the context of networks, the term ring topology is classically used to define a structure of nodes where each node has two neighbours and the entire set of nodes forms a single continuous pathway for signals through each node. In the context of distributed optimization algorithms the ring topology is widely used as a structure for computational units in order to build up the communication in the search, see e.g. [24] and [25]. In the proposed CcDE, each core

communicates with the others by exchanging pieces of information, in an adaptive fashion, about the most appropriate parameter setting during the various stages of the evolution. In other words, a migration of the elite individual, as well as the scale factor inheritance mechanism (see [26]) have been implemented. In addition, a perturbation mechanism on the virtual population attempts to inhibit the premature convergence and thus promotes the exploration of unexplored areas of the decision space. The proposed CcDE can be seen as a small population-based algorithm where each individual is a cDE with its elite and virtual population. According to the memory usage this kind of algorithms should be comparable to light population based algorithms which work with a small population size. Thus, CcDE attempts to be still useful for applications with memory limitations but employs multiple cDE cores for exploring complex decision spaces without a premature diversity loss, consequently displaying a good performance on a broad and diverse set of test problems.

The remainder of this article is organized in the following way. Section 2 describes the algorithmic components characterizing the proposed algorithm. Section 3 shows the numerical results and highlights the performance of the CcDE with respect to compact and population-based algorithms. Section 4 gives the conclusion of this work.

2 Composed compact differential evolution: algorithmic description

In order to clarify the notation used throughout this article we refer to the minimization problem of an objective function $f(x)$, where x is a vector of n design variables in a decision space D. Without loss of generality, let us assume that design variables are normalized so that each search interval is $[-1, 1]$.

At the beginning of the optimization process, $N_c(2 \times n)$-matrices are generated. Parameter N_c stands for number of compact units. Each matrix PV_m $(m = 1, 2, \ldots, N_c)$ is called Probability Vector, see [15]. More specifically, PV_m is a $n \times 2$ matrix:

$$PV_m^t = [\mu^t, \sigma^t] \tag{1}$$

where μ and σ are, respectively, vectors containing, for each design variable, mean and standard deviation values of a Gaussian Probability Distribution Function (PDF) truncated within the interval $[-1, 1]$. The height of the PDF has been normalized in order to keep its area equal to 1. The apex t indicates the generation (number of performed comparison).

In the initialization phase, for each design variable i, $\mu^1[i] = 0$ and $\sigma^1[i] = \lambda$ where λ is a large positive constant (e.g. $\lambda = 10$). This initialization of σ values is done in order to simulate a uniform distribution. Then, one individual is sampled from each PV_m. These individuals are indicated as elite x_e^m and are the elite individual related to the mth probability vector PV_m. In other words, N_c compact units for cDE, see [20], are generated. For each unit m, a scale factor F^m is uniformly sampled from the interval $[0, 2]$.

For simplicity of notation, let us refer to the generic mth compact unit as PV and x_e, since the same operations are repeated within each unit. The search mechanism, within each unit and at each generation, is organized in the following way. Three individuals x_r, x_s and x_t are pseudo-randomly sampled from the PV. More specifically, the sampling mechanism of a design variable $x_r[i]$ associated to a generic candidate solution x_r from PV consists of the following steps. As mentioned above, for each design variable indexed by i, a truncated Gaussian PDF characterized by a mean value $\mu[i]$ and a standard deviation $\sigma[i]$ is associated. The formula of the PDF is:

$$PDF(\mu[i], \sigma[i]) = \frac{e^{\frac{-(x-\mu[i])^2}{2\sigma[i]^2}}\sqrt{\frac{2}{\pi}}}{\sigma[i]\left(\operatorname{erf}\left(\frac{\mu[i]+1}{\sqrt{2}\sigma[i]}\right) - \operatorname{erf}\left(\frac{\mu[i]-1}{\sqrt{2}\sigma[i]}\right)\right)} \tag{2}$$

where erf is the error function, see [27].

From the PDF, the corresponding Cumulative Distribution Function (CDF) is constructed by means of Chebyshev polynomials according to the procedure described in [28]. It must be observed that the codomain of CDF is $[0, 1]$. In order to sample the design variable $x_r[i]$ from PV, a random number $rand(0,1)$ is sampled from a uniform distribution. The inverse function of CDF, in correspondence of $rand(0,1)$, is then calculated. This latter value is $x_r[i]$. A detailed description of the sampling mechanism is given in [20].

A provisional offspring x'_{off} is then generated by mutation, according to a DE logic, as:

$$x'_{off} = x_t + F(x_r - x_s) \tag{3}$$

where $F \in [0, 2[$ is a scale factor which controls the length of the exploration vector $(x_r - x_s)$ and thus determines how far from point x_t the offspring should be generated. This scale factor is the value Fm (corresponding to the mth unit) mentioned above. The mutation scheme shown in formula (3) is also known as DE/rand/1. It is important to remark that other variants of the mutation rule have been proposed in literature for standard DE, see [21].

When the provisional offspring has been generated by mutation, the exponential crossover is applied. A design variable of the provisional offspring $x'_{off}(j)$ is randomly selected and copied into the jth design variable of the elite solution x_e (its copy). This guarantees that parent and offspring have different genotypes. Subsequently, a set of

```
x_off = x_e
generate j = 1 + round(n × rand(0, 1))
x_off(j) = x'_off(j)
p ← 0
while rand(0, 1) ≤ Cr AND p < n − 1 do
   x_off(1 + (j + p) mod n) = x'_off(1 + (j + p) mod n)
   p = p + 1
   generate rand(0, 1)
end while
```

Fig. 1 Exponential crossover pseudo-code

random numbers between 0 and 1 are generated. As long as $rand(0,1) \leq Cr$, where the crossover rate Cr is a predetermined parameter, the design variables from the provisional offspring (mutant) are copied into the corresponding positions of the parent x_i. The first time that $rand(0,1) > Cr$ the copy process is interrupted. Thus, all the remaining design variables of the offspring are copied from the parent. For the sake of clarity the pseudo-code of the exponential crossover is shown in Fig. 1

In other words, this crossover operator generates offspring composed of the elite x_e and contains, within it, a section of the chromosome of the mutant vector x'_{off}.

When the offspring is generated, its fitness value is computed and compared with that of the elite individual. The comparison allows the definition of winner and loser solutions. The winner solution biases the virtual population by affecting the PV values. The update rule for μ values is given by:

$$\mu^{t+1} = \mu^t + \frac{1}{N_p}(winner - loser), \tag{4}$$

where N_p is the virtual population size. The update rule for σ values is given by:

$$\left(\sigma^{t+1}\right)^2 = (\sigma^t)^2 + (\mu^t)^2 - \left(\mu^{t+1}\right)^2 + \frac{1}{N_p}\left(winner^2 - loser^2\right). \tag{5}$$

Details for constructing formulas (4) and (5) are given in [15].

The formulas displayed in Eqs. (4) and (5) rule the convergence of the virtual population. More specifically, the mean value of the PDF representing the population is moved towards the winner solution while the standard deviation tends to progressively narrow around the most promising solution, thus resulting in a σ value tending toward zero. The latter condition is here indicated as convergence, see [20].

In this paper extra rules for modifying the PV values are employed. More specifically, with a probability M_p the PV is perturbed. The mean value μ is perturbed according to the following formula:

$$\mu^{t+1} = \mu^{t+1} + 2\tau \cdot rand(0,1) - \tau \tag{6}$$

where τ is a weight representing the maximum amplitude of perturbation. Similar to typical DE schemes, a toroidal mechanism (see [29]) ensures that μ is bounded by 0 and 1 (for example $1 + 0.1 = 0.1$). The perturbation rule for the sigma is given by:

$$\left(\sigma^{t+1}\right)^2 = \left(\sigma^{t+1}\right)^2 + \tau \cdot rand(0,1). \tag{7}$$

It is worthwhile commenting the perturbation mechanism reported in formulas (6) and (7). The proposed compact DE scheme employs a fairly exploitative structure due to its nature and the employment of the exponential crossover. Thus, each compact unit is likely to prematurely converge towards suboptimal solution. In order to mitigate this effect the perturbation mechanism then inhibits the algorithmic convergence and forces the algorithm to search elsewhere in the decision space, possibly detecting new promising solutions. It can be observed that the perturbation is equivalent to the replacement of part of the population in a population-based DE.

While the N_c compact units evolve independently, two mechanisms promote the communication amongst the units. In order to understand both these mechanisms let us consider the compact units to be arranged according to a ring topology. In other words, each mth unit has two neighbour units: unit $(m-1)$th and unit $(m+1)$th. For the sake of clarity, we remark that the neighbours of the N_c^{th} unit are $(N_c - 1)$th and 1st units, respectively. The first mechanism is the unidirectional migration (see [24]) of the elite individual. More specifically, at each step (comparison between offspring and elite) of each compact unit, with a probability M_e, the elite solution x_e^m is duplicated and replaces the solution in x_e^{m+1} if the sender outperforms the receiver. In other words, x_e^m is overwritten in x_e^{m+1} if $f\left(x_e^m\right) < f\left(x_e^{m+1}\right)$ (minimization problem).

The second mechanism is the scale factor inheritance proposed in [26]. The scale factor inheritance mechanism occurs contextually with the migration. More specifically, when the migration occurs the $(m+1)$th unit inherits the scale factor F^m after a perturbation. More specifically, the scale factor F^{m+1} related to the $(m+1)$th unit is updated according to the following formula:

$$F^{m+1} = F^m + \alpha\mathcal{N}(0,1) \tag{8}$$

where $\mathcal{N}(0,1)$ is a pseudo-random value sampled from a normal distribution characterized by a zero mean and variance equal to 1. The constant value α has the role of controlling the range of perturbation values $\alpha\mathcal{N}(0,1)$. It must be observed that we did not impose any bounds for the variation of F. On the contrary, we decided to allow an unbounded variation of the control parameter and rely on the self-adaptation mechanism.

Fig. 2 Pseudo-code of elite migration and scale factor inheritance at the *m*th unit

```
for each generation do
  perform a cDE generation (offspring, comparison, replacement)
  if f(x_e^m) < f(x_e^{m+1}) AND rand(0,1) < M_e then
    send a copy of the elite individual to the neighbour unit
    replace the elite individual: x_e^{m+1} = x_e^m
    apply the scale factor inheritance mechanism
    replace the scale factor: F^{m+1} = F^m + αN(0,1)
  end if
end for
```

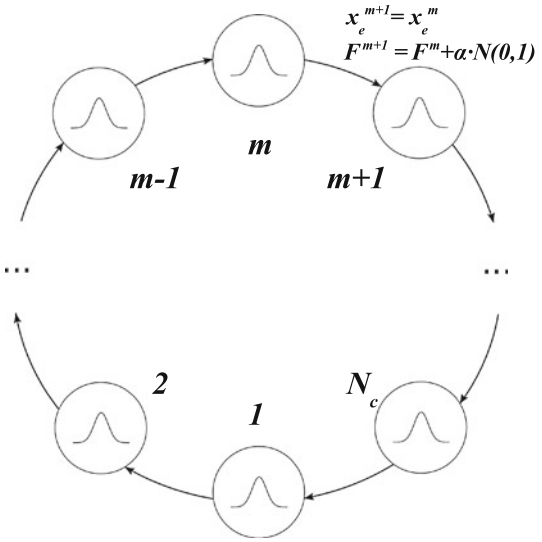


Fig. 3 Graphical representation of CcDE

For the sake of clarity the pseudo-code of the elite migration and scale factor inheritance at the generic *m*th unit is given in Fig. 2.

These steps are repeated until a budget condition is satisfied.

A graphical representation of the proposed CcDE is given in Fig. 3. Each compact unit is schematically represented as truncated Gaussian distribution function. The arrows indicate elite migration and scale factor inheritance mechanism. For the sake of clarity, the pseudo-code summarizing the working principles of CcDE is shown in Fig. 4.

2.1 Algorithmic philosophy

This section summarizes the considerations regarding the functioning of the CcDE. As highlighted above, CcDE is composed of multiple compact units (cDEs) interconnected by means of the mechanisms of elite migration and scale factor inheritance. In order to understand the working principle of the proposed algorithm it is necessary to make some considerations about each cDE unit. As highlighted in [30], the success of standard population-based DE is due to an implicit self-adaptation contained within the algorithmic structure. Since, for each candidate solution, the search rule depends on other solutions belonging to the population (e.g. x_t, x_r, and x_s), the capability of detecting new promising offspring solutions depends on the current distribution of the solutions within the decision space. During early stages of the optimization process, solutions tend to be spread out within the decision space. For a given scale factor value, this implies that the mutation appears to generate new solutions by exploring the space by means of a large step size (if x_r and x_s are distant solutions, $F^m(x_r - x_s)$ is a vector characterized by a large modulus). During the optimization process, the solutions of the population tend to concentrate on specific parts of the decision space. Therefore, the step size in the mutation is progressively reduced and the search is performed in the neighbourhood of the solutions. In other words, due to its structure, a DE scheme is highly explorative at the beginning of the evolution and subsequently becomes more exploitative during fine tuning.

However, as highlighted in [21], the DE mechanism hides an important limitation. If, for some reasons, the algorithm does not succeed in generating offspring solutions which outperform the corresponding parent, the search is repeated again with similar step size values and will likely fail by falling into an undesired stagnation condition (see [31]). In other words, DE offers a wide margin of improvement. For this reasons, several modified DE schemes have been proposed in order to enhance the performance of DE of various problems, e.g. [23, 32], and [33]. In almost all these improved versions of DE a certain degree of randomization is introduced within the search logic, see [21]. Although cDE schemes have been introduced in order to deal with hardware characterized by a limited memory, due to the presence of a statistical model of the population a certain degree a randomization is implicitly introduced every time the solutions (e.g. x_t, x_r, and x_s) are sampled from the truncated Gaussian distributions. This aspect of the algorithm has, apparently, a positive effect on the algorithmic performance since, as shown in [20], despite its limited memory employment, cDE is likely to outperform its population-based version in various problems for a relatively low dimensionality. On the other hand, due to its inner structure cDE has more

Fig. 4 CcDE pseudo-code

```
counter t = 0
for m = 1 : N_c do
  for i = 1 : n do
    {** PV initialization **}
    initialize μ[i] = 0
    initialize σ[i] = λ
  end for
end for
generate N_c elite solutions x_e by means of the corresponding PV
while budget condition do
  {** Mutation **}
  for m = 1 : N_c do
    generate 3 individuals x_r, x_s, and x_t by means of PV
    compute x'_off = x_t + F(x_r − x_s)
    {** Crossover **}
    apply exponential crossover shown in Fig. 1 and generate x_off
    {** Elite Selection **}
    [winner, loser] = compete(x_off, x_e)
    if x_off == winner then
      x_e = x_off
    end if
    {** PV Update **}
    μ^{t+1} = μ^t + (1/N_p)(winner − loser)
    σ^{t+1} = sqrt((σ^t)^2 + (μ^t)^2 − (μ^{t+1})^2 + (1/N_p)(winner^2 − loser^2))
    {** Perturbation of the PV **}
    if rand(0,1) < M_p then
      μ^{t+1} = μ^{t+1} + 2τ · rand(0,1) − τ
      σ^{t+1} = sqrt((σ^{t+1})^2 + τ · rand(0,1))
    end if
    perform elite migration and scale factor inheritance as shown in Fig. 2
    t = t + 1
  end for
end while
```

exploitative properties with respect to its corresponding population-based version. In other words, since compact algorithms do not handle an actual population of solution but only its statistical representation which tends to converge around the most promising solutions, see [20], each compact unit of CcDE can be seen as a local search algorithm which attempts to improve its elite solution. More specifically, the working principle of each compact unit can be seen as a local search which is periodically "disturbed" (by the perturbation mechanism). The moving operators of mutation and crossover are supposed to detect promising search directions and quickly exploit them. This fact corresponds to the convergence of the virtual population towards the elite. This convergence is likely to be premature. The perturbation mechanism then inhibits the algorithmic convergence and forces the algorithm to search elsewhere in the decision space, possibly detecting new promising solutions. In conclusion, each compact unit of the CcDE can be seen as a multi-start local search algorithm which performs a highly exploitative mini-search between each pair of *PV* perturbations. However, this mini-search occurs while the memory of the previously achieved enhancements is kept.

The multiple local search is then coordinated by the entire ring structure. Similar to distributed systems, see e.g. [24] and [25], the migration of the elite has the role of propagating the most promising genotypes throughout the entire ring. The migrated promising solutions can then suggest search directions and modify the convergence process over the virtual populations. In addition, the injection of the new elite into a compact unit where the virtual population is centred on a different area of the search space promotes the search of unexplored genotypes since the solutions generating the provisional offspring are likely to be different from the elite solution. This mechanism is thus supposed to improve upon the elite and support the search of the global optimum.

The scale factor inheritance mechanism has a slightly different role since it may be seen as the propagation of the search rule instead of the promising genotype. As highlighted in [26], the employment of a unique and constant scale factor value can be improper since the exploratory moves depend on distribution of the solution within the decision space. For example, for a highly multi-modal problem, a scale factor $F \approx 1$ can generate very long moving vectors $F(x_r - x_s)$ if the population is spread out

within the decision space and very short moving vectors if the population is concentrated in some areas of the decision space. This fact may lead to an excessively explorative behaviour during some stages of the evolution and an excessively exploitative behaviour during other stages. These two behaviours can cause, respectively, stagnation due to an incapability to detect a promising search direction and a premature convergence due to the excessive exploitation of a suboptimal basin of attraction. In other words, a proper choice of the scale factor depends not only on the optimization problem but also on the stage of the evolution. In this sense, the scale factor inheritance relies on the fact that the most promising values are propagated throughout the ring topology. However, the perturbation of the scale factor while propagated guarantees a certain degree of diversity in the search logic over the compact units and thus that exploration is continued despite the detection of strong genotypes.

Thus, CcDE can be seen as an algorithm composed out of multiple local search units coordinated by means of a self-adaptation which promotes highly performing genotypes and efficient search rules. The two self-adaptation mechanisms supervise the multiple local search by recombining the available pieces of information in order to solve global optimization problems. In other words, the local knowledge of the fitness landscape is gained by means of each compact unit while the global knowledge is pursued by the ring structure with elite migration and scale factor inheritance.

Finally, it is worthwhile commenting the memory requirements of CcDE. As explained in [20], cDE requires the memory equivalent to four solutions since one memory slot is taken by the elite two by the *PV* and one for offspring generation. In CcDE each compact unit requires three memory slots: one for the elite and two for the *PV*. In addition one memory slot, common to all the compact units, is used for offspring generation. The global memory employment of CcDE is then given by $3 \cdot N_c + 1$ slots. As it will be shown in Sect. 3, not a high value of N_c is necessary to solve complex problems. For this reason, the memory employment of CcDE can be considered comparable to that of a simple population-based algorithm which does not require neither a large population size nor extra memory structure such as an archive or a learning period database.

3 Numerical results

Table 1 lists the test problems that have been considered in this study. All the algorithms in this paper have been run for these test problems. Test problems $f_{19} - f_{26}$ are characterized by a unique dimensionality value (indicated in the list).

Table 1 Test problems

f_1 Shifted sphere function: F_1 from [34] with $n = 30$.
f_2 Shifted Schwefel's Problem 1.2: F_2 from [34] with $n = 30$
f_3 Rosenbrock's function: f_3 from [23] with $n = 30$
f_4 Shifted Ackley's function: f_5 from [23] with $n = 30$
f_5 Shifted rotated Ackley's function: f_6 from [23] with $n = 30$
f_6 Shifted Griewank's function: f_7 from [23] with $n = 30$
f_7 Shifted rotated Griewank's function: f_8 from [23] with $n = 30$
f_8 Shifted Rastrigin's function: F_9 from [34] with $n = 30$
f_9 Shifted rotated Rastrigin's function: F_{10} from [34] with $n = 30$
f_{10} Shifted non continuous Rastrigin's function: f_{11} from [23] with $n = 30$
f_{11} Schwefel's function: f_{12} from [23] with $n = 30$
f_{12} Schwefel Problem 2.22: f_2 from [35] with $n = 10$
f_{13} Schwefel Problem 2.21: f_4 from [35] with $n = 10$
f_{14} Generalized penalized function 1: f_{12} from [35] with $n = 10$
f_{15} Generalized penalized function 2: f_{13} from [35] with $n = 10$
f_{16} Schwefel's Problem 2.6 with Global Optimum on Bounds: F_5 from [34] with $n = 30$
f_{17} Shifted Rotated Weierstrass Function: F_{11} from [34] with $n = 30$
f_{18} Schwefel's Problem 2.13: F_{12} from [34] with $n = 30$
f_{19} Kowalik's function: f_{15} from [36] with $n = 4$
f_{20} Six-hump camel-back function: f_{20} from [23] with $n = 2$
f_{21} Branin function: f_{17} from [35] with $n = 2$
f_{22} Hartman's function 1: f_{19} from [36] with $n = 4$
f_{23} Hartman's function 2: f_{20} from [36] with $n = 6$
$f_{24} - f_{26}$ Shekel's family: $f_{21} - f_{24}$ from [36] with $n = 4$.

Thus, 26 test problems in total have been used in this study. For each algorithm, 30 independent runs have been performed. The budget of each single run has been fixed equal to $5,000 \cdot n$ fitness evaluations. The proposed CcDE has been compared with modern compact and population-based algorithms. For all the experiments reported in this paper the CcDE has been run with $N_c = 10$, $N_p = 20$, $Cr = 0.1$, $M_p = 0.0001$, $\tau = 0.1$, $\alpha = 0.1$ and $M_e = 0.0001$. It must be remarked that for compact algorithms the so called "generation" is actually a single comparison/fitness evaluation. Thus, the probabilities indicated above refer to each fitness evaluation. This explains why although the numbers appear to be low, the events occur fairly often during each run. For example, in the case of $n = 10$ a number of fitness evaluations equal to 50,000 is performed. During a single run the perturbation occurs on average 50 times over the entire ring structure, which are apparently enough to detect solutions with a high performance.

3.1 Comparison with modern compact algorithms

The following compact algorithms have been considered for comparison against CcDE.

Table 2 Average final fitness values ± standard deviations for compact algorithms

Test problem	cGA	rcGA	cDE	McDE	(1 + 1)-CMA-ES	CcDE
f_1	1.446e+04 ± 4.63e+03	1.906e+04 ± 9.62e+03	**4.520e−28 ± 1.74e−27**	3.235e−22 ± 3.52e−22	1.961e−27 ± 1.47e−27	2.380e−02 ± 3.47e−02
f_2	1.628e+06 ± 7.00e+05	2.677e+04 ± 4.78e+03	9.865e+03 ± 2.52e+03	3.896e+03 ± 1.66e+03	**6.430e−26 ± 7.81e−26**	2.062e+02 ± 2.74e+02
f_3	2.432e+09 ± 1.62e+09	1.803e+09 ± 2.02e+09	9.898e+01 ± 1.41e+02	1.795e+02 ± 2.17e+02	**1.017e+00 ± 1.80e+00**	9.967e+01 ± 7.48e+01
f_4	1.681e+01 ± 9.45e−01	1.859e+01 ± 4.15e−01	1.074e+01 ± 1.75e+00	7.761e−02 ± 2.63e−01	1.946e+01 ± 1.77e−01	**1.500e−02 ± 1.67e−02**
f_5	1.721e+01 ± 1.41e+00	1.880e+01 ± 4.54e−01	1.028e+01 ± 1.83e+00	**1.084e+00 ± 9.46e−01**	1.942e+01 ± 1.99e−01	2.586e+00 ± 1.07e+00
f_6	8.840e+02 ± 3.08e+01	**2.259e−03 ± 4.11e−03**	1.883e−01 ± 2.03e−01	2.630e−02 ± 6.65e−02	9.842e−03 ± 1.05e−02	2.188e−02 ± 2.46e−02
f_7	8.778e+02 ± 3.34e+01	**3.403e−02 ± 9.71e−02**	1.891e−01 ± 2.06e−01	2.050e−01 ± 2.39e−01	2.843e−01 ± 2.34e−01	1.292e−01 ± 1.85e−01
f_8	2.265e+02 ± 4.06e+01	2.037e+02 ± 2.74e+01	5.959e+01 ± 1.33e+01	1.390e+02 ± 2.22e+01	1.912e+02 ± 3.13e+01	**1.433e−02 ± 2.83e−02**
f_9	3.013e+02 ± 4.72e+01	1.985e+02 ± 3.06e+01	1.219e+02 ± 2.58e+01	1.805e+02 ± 3.06e+01	1.892e+02 ± 4.33e+01	**7.212e+01 ± 5.20e+01**
f_{10}	1.307e+05 ± 4.96e+04	2.900e+03 ± 3.07e+03	6.448e+03 ± 2.75e+03	9.842e+03 ± 4.21e+03	3.980e+02 ± 1.14e+02	**8.339e+00 ± 1.04e+01**
f_{11}	4.947e+03 ± 7.03e+02	3.156e+03 ± 7.54e+02	9.972e+02 ± 3.25e+02	1.138e+03 ± 5.10e+02	5.815e+03 ± 8.42e+02	**5.546e−02 ± 9.79e−02**
f_{12}	5.614e+00 ± 2.91e+00	4.127e+00 ± 4.90e+00	2.558e−02 ± 7.10e−03	8.208e−01 ± 1.74e−01	**6.646e−04 ± 1.81e−03**	1.694e−02 ± 1.82e−02
f_{13}	3.309e+01 ± 1.13e+01	**−1.000e+02 ± 5.06e−09**	−1.000e+02 ± 1.73e−06	−1.000e+02 ± 4.92e−04	−1.000e+02 ± 1.20e−03	−9.999e+01 ± 2.93e−02
f_{14}	4.472e+04 ± 1.37e+05	1.401e+00 ± 1.91e+00	1.982e−04 ± 1.74e−04	3.132e−01 ± 2.15e−01	4.295e+00 ± 5.06e+00	**1.277e−04 ± 2.39e−04**
f_{15}	1.121e+06 ± 3.66e+06	−7.869e−01 ± 8.94e−01	**−1.148e+00 ± 1.67e−03**	−2.771e−01 ± 2.63e−01	1.534e+00 ± 4.07e+00	−1.141e+00 ± 1.78e−02
f_{16}	1.172e+04 ± 2.57e+03	8.975e+03 ± 2.38e+03	8.023e+03 ± 3.42e+03	6.918e+03 ± 2.97e+03	**4.066e+03 ± 1.14e+03**	5.638e+03 ± 2.28e+03
f_{17}	1.307e+02 ± 2.03e+00	1.226e+02 ± 3.21e+00	1.242e+02 ± 3.21e+00	1.300e+02 ± 1.22e+00	**1.223e+02 ± 4.11e+00**	1.223e+02 ± 3.07e+00
f_{18}	2.958e+05 ± 1.05e+05	3.089e+05 ± 1.38e+05	5.480e+04 ± 3.21e+04	1.280e+05 ± 5.48e+04	**3.661e+03 ± 6.25e+03**	1.041e+04 ± 5.98e+03
f_{19}	5.296e−02 ± 9.80e−09	**5.296e−02 ± 5.22e−18**	5.296e−02 ± 3.28e−11	5.296e−02 ± 4.63e−10	5.296e−02 ± 4.34e−18	5.296e−02 ± 2.20e−16
f_{20}	−9.632e−01 ± 4.22e−02	**−1.067e+00 ± 4.09e−16**	−1.067e+00 ± 1.50e−05	−1.067e+00 ± 2.82e−05	−1.067e+00 ± 3.37e−16	−1.067e+00 ± 1.31e−09
f_{21}	2.337e+01 ± 1.14e+00	3.979e−01 ± 9.70e−13	3.979e−01 ± 1.71e−05	3.979e−01 ± 5.64e−05	**3.979e−01 ± 0.00e+00**	3.979e−01 ± 9.67e−11
f_{22}	−3.760e+00 ± 1.95e−02	**−3.863e+00 ± 1.94e−15**	−3.863e+00 ± 1.21e−06	−3.863e+00 ± 5.89e−06	−3.863e+00 ± 2.04e−15	−3.863e+00 ± 5.50e−07
f_{23}	−4.819e−01 ± 4.27e−02	−3.238e+00 ± 5.53e−02	−3.288e+00 ± 5.54e−02	−3.317e+00 ± 2.43e−02	−3.293e+00 ± 5.27e−02	**−3.322e+00 ± 2.99e−06**
f_{24}	−2.773e+00 ± 1.13e+00	−6.458e+00 ± 2.47e+00	−5.451e+00 ± 3.24e+00	−8.905e+00 ± 2.30e+00	−6.082e+00 ± 3.36e+00	**−9.943e+00 ± 1.03e+00**
f_{25}	−2.628e+00 ± 9.63e−01	−7.258e+00 ± 3.01e+00	−5.504e+00 ± 3.33e+00	−1.011e+01 ± 1.07e+00	−5.557e+00 ± 3.30e+00	**−1.018e+01 ± 1.08e+00**
f_{26}	−2.764e+00 ± 9.00e−01	−6.940e+00 ± 3.19e+00	−6.239e+00 ± 3.75e+00	**−1.014e+01 ± 1.56e+00**	−4.572e+00 ± 2.96e+00	−9.910e+00 ± 2.03e+00

- compact Genetic Algorithm (cGA) with persistent elitism proposed in [13].
- real compact Genetic Algorithm (rcGA) with persistent elitism proposed in [15].
- (1 + 1) Covariance Matrix Adaptation Evolution Strategy ((1 + 1)-CMA-ES) with Cholesky update proposed in [37]. All the parameters have been set as proposed in [37] and the initial σ has been set equal to 1.
- compact Differential Evolution (cDE) proposed in [20] with persistent elitism, DE/rand/1 mutation and binomial crossover. The cDE has been run with $F = 0.5$ and $Cr = 0.7$.
- Memetic compact Differential Evolution (McDE) proposed in [22] with persistent elitism, DE/rand/1 mutation and binomial crossover. The McDE has been run with $F = 0.7$, $Cr = 0.7$, probability of local search activation $p_{ls} = 0.005$, reduction factor $\beta = 0.8$, and initial hyper-cube dimension δ equal to 10% of the search space.

The virtual population size N_p for cGA, rcGA, cDE, and McDE has been set equal to 300, as suggested in [20] and [22].

Table 3 Wilcoxon Rank-Sum test for compact algorithms

Test Problem	cGA	rcGA	cDE	McDE	(1 + 1)-CMA-ES
f_1	+	+	−	−	−
f_2	+	+	+	+	−
f_3	+	+	=	=	−
f_4	+	+	+	=	+
f_5	+	+	+	−	+
f_6	+	−	+	=	−
f_7	+	−	=	=	+
f_8	+	+	+	+	+
f_9	+	+	+	+	+
f_{10}	+	+	+	+	+
f_{11}	+	+	+	+	+
f_{12}	+	+	+	+	−
f_{13}	+	=	=	=	=
f_{14}	=	+	=	+	+
f_{15}	=	=	=	+	+
f_{16}	+	+	+	=	−
f_{17}	+	=	+	+	=
f_{18}	+	+	+	+	−
f_{19}	+	=	=	+	=
f_{20}	+	=	+	+	=
f_{21}	+	=	=	+	=
f_{22}	+	=	+	+	=
f_{23}	+	+	+	=	+
f_{24}	+	+	+	+	+
f_{25}	+	+	+	=	+
f_{26}	+	+	+	=	+

Table 2 shows the average of the final results detected by each algorithm ± the corresponding standard deviation values calculated over the performed 30 runs. The best results are highlighted in bold face. In order to strengthen the statistical significance of the results, the Wilcoxon Rank-Sum test has also been applied according to the description given in [38], where the confidence level has been fixed to 0.95. Table 3 shows the results of the Wilcoxon test for each version of CcDE against the other algorithms considered in this study. A "+" indicates the case in which CcDE statistically outperforms, for the corresponding test problem, the algorithm indicated in the top of the column; a "=" indicates that no significant difference between the performances can be detected with the Wilcoxon test; a "−" indicates that CcDE is outperformed.

Numerical results show that the proposed CcDE outperforms the other algorithms considered in this study.

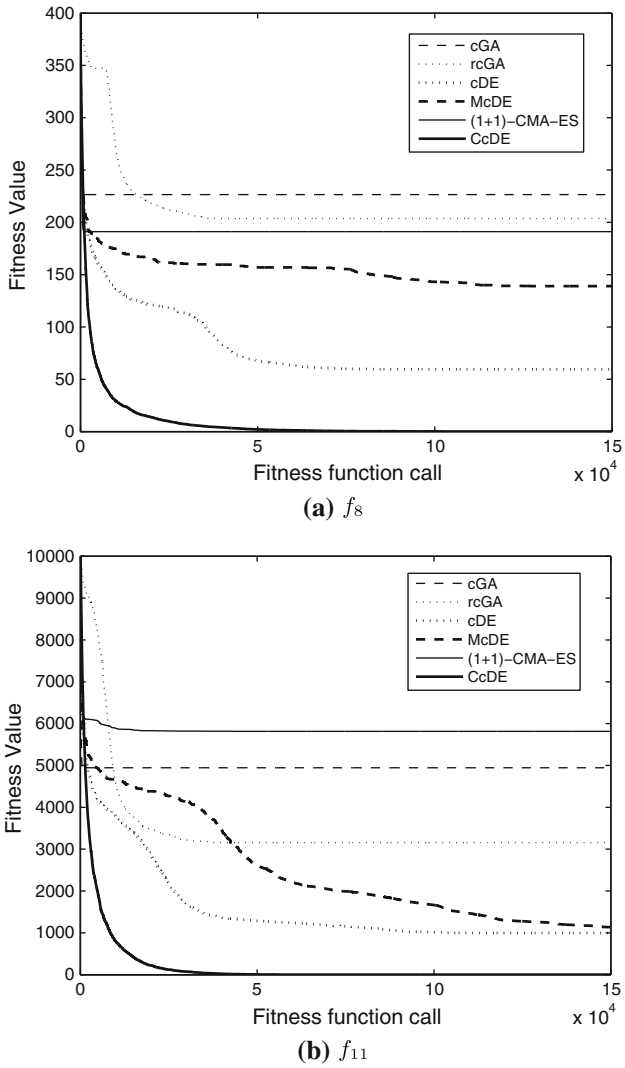


Fig. 5 Performance trends of CcDE and compact algorithms

While the comparison with cGA, rcGA, cDE and McDE displays a clear superiority of CcDE for the considered problems, the last column in Table 3 shows that CcDE outperforms, on average, $(1+1)$-CMA-ES (7 lost and 13 won comparisons). Numerical results thus show that the proposed compact structure appears promising while compared to other algorithms employing a similar logic. With reference to the other compact algorithms considered in this study, CcDE instead of performing the search along one direction, explores the decision space from multiple search directions. In most cases, the redundant action of the various compact units appears to guarantee that the search towards the optimum is not stopped by premature convergence but, on the contrary, is continued and new promising genotypes are detected. This effect can be clearly see for non-separable functions, for example rotated function as f_9.

Figure 5 shows average performance trends of the five considered compact algorithms over a selection of the test problems considered in this study.

3.2 Comparison with modern population-based algorithms

The following population-based algorithms have been considered for comparison against DEcDE.

- Estimation of Distribution Algorithm with MultiVariate Gaussian model (EDA_{mvg}) proposed in [39]. EDA_{mvg} has been run with learning rate $\alpha = 0.2$, population size $N_p = 50$, selection ratio $\tau = 0.3$, and maximum amplification value $Q = 1.5$.
- Real Coded Memetic Algorithm (RCMA) proposed in [40]. RCMA has been run with population size $N_p = 50$, crossover parameter $\alpha = 0.5$, mutation probability 0.125, maximum generation number $T = 10000$, $\beta = 5$, maximum number of individuals taking part to the negative assortative mating $N_{nam} = 3$, roulette wheel selection. The other fixed parameters have been set as suggested in [40].
- Differential Evolution with adaptive hill-climb Simplex Crossover (DEahcSPX) proposed in [41]. DEahcSPX

Table 4 Average final fitness values ± standard deviations for population-based algorithms

Test Problem	$RCEDA_{mvg}$	RCMA	DEahcSPX	CcDE
f_1	6.955e+01 ± 1.47e+02	**3.411e−16 ± 6.85e−16**	9.690e+01 ± 1.50e+01	2.380e−02 ± 3.47e−02
f_2	3.145e+02 ± 3.01e+02	**1.255e−13 ± 2.95e−13**	1.805e+03 ± 2.36e+02	2.062e+02 ± 2.74e+02
f_3	2.319e+06 ± 5.55e+06	**2.827e+01 ± 1.28e−01**	1.110e+05 ± 3.11e+04	9.967e+01 ± 7.48e+01
f_4	3.271e+00 ± 4.71e−01	**7.012e−09 ± 1.04e−08**	2.567e+00 ± 4.64e−01	1.500e−02 ± 1.67e−02
f_5	3.406e+00 ± 1.07e+00	**8.119e−09 ± 8.45e−09**	2.509e+00 ± 4.09e−01	2.586e+00 ± 1.07e+00
f_6	2.638e+02 ± 3.90e+01	6.575e+00 ± 3.85e+00	4.886e+00 ± 3.60e−01	**2.188e−02 ± 2.46e−02**
f_7	2.735e+02 ± 4.50e+01	6.063e+00 ± 3.50e+00	4.824e+00 ± 3.50e−01	**1.292e−01 ± 1.85e−01**
f_8	1.770e+02 ± 1.35e+01	**7.105e−15 ± 2.55e−14**	3.405e+01 ± 3.48e+00	1.433e−02 ± 2.83e−02
f_9	1.816e+02 ± 1.26e+01	**4.737e−15 ± 2.32e−14**	1.172e+02 ± 2.56e+01	7.212e+01 ± 5.20e+01
f_{10}	1.402e+04 ± 5.35e+04	1.003e+04 ± 1.78e+04	3.026e+03 ± 3.75e+02	**8.339e+00 ± 1.04e+01**
f_{11}	1.015e+04 ± 4.23e+02	2.989e+03 ± 5.98e+02	8.258e+02 ± 8.20e+01	**5.546e−02 ± 9.79e−02**
f_{12}	1.294e+00 ± 6.90e−01	**6.329e−10 ± 2.63e−09**	1.333e−01 ± 3.16e−02	1.694e−02 ± 1.82e−02
f_{13}	−9.475e+01 ± 2.21e+01	−6.845e+01 ± 1.17e+01	−8.561e+01 ± 1.39e+00	**−9.999e+01 ± 2.93e−02**
f_{14}	5.928e−01 ± 1.34e+00	**1.030e−06 ± 1.29e−06**	1.157e−02 ± 8.74e−03	1.277e−04 ± 2.39e−04
f_{15}	−1.290e−01 ± 1.90e+00	**−1.149e+00 ± 3.70e−03**	−8.535e−01 ± 1.08e−01	−1.141e+00 ± 1.78e−02
f_{16}	1.007e+04 ± 3.36e+03	9.318e+03 ± 2.45e+03	9.314e+03 ± 1.02e+03	**5.638e+03 ± 2.28e+03**
f_{17}	1.303e+02 ± 7.51e−01	1.256e+02 ± 3.65e+00	1.298e+02 ± 1.15e+00	**1.223e+02 ± 3.07e+00**
f_{18}	5.272e+05 ± 2.93e+05	1.298e+05 ± 4.67e+04	7.203e+04 ± 1.03e+04	**1.041e+04 ± 5.98e+03**
f_{19}	5.296e−02 ± 6.61e−09	**5.296e−02 ± 4.34e−18**	5.296e−02 ± 3.64e−09	5.296e−02 ± 2.20e−16
f_{20}	**−1.067e+00 ± 4.54e−16**	−1.067e+00 ± 6.64e−06	−1.067e+00 ± 2.16e−05	−1.067e+00 ± 1.31e−09
f_{21}	3.987e−01 ± 3.42e−03	3.980e−01 ± 1.95e−04	3.980e−01 ± 1.97e−04	**3.979e−01 ± 9.67e−11**
f_{22}	−3.847e+00 ± 3.37e−02	−3.863e+00 ± 5.91e−06	−3.863e+00 ± 5.56e−06	**−3.863e+00 ± 5.50e−07**
f_{23}	−3.126e+00 ± 1.65e−01	−3.268e+00 ± 6.07e−02	−3.322e+00 ± 1.33e−04	**−3.322e+00 ± 2.99e−06**
f_{24}	−5.633e+00 ± 3.50e+00	−5.302e+00 ± 3.20e+00	**−1.004e+01 ± 1.19e−01**	−9.943e+00 ± 1.03e+00
f_{25}	−9.252e+00 ± 2.42e+00	−4.648e+00 ± 3.68e+00	**−1.024e+01 ± 1.06e−01**	−1.018e+01 ± 1.08e+00
f_{26}	−8.449e+00 ± 3.01e+00	−5.482e+00 ± 3.33e+00	**−1.038e+01 ± 1.93e−01**	−9.910e+00 ± 2.03e+00

has been run with $N_p = 50$, scale factor $F = 0.9$ and crossover rate $Cr = 0.9$ as suggested in the paper, the number of points involved in the hill-climb $n_p = 3$, factor $\epsilon = 1$.

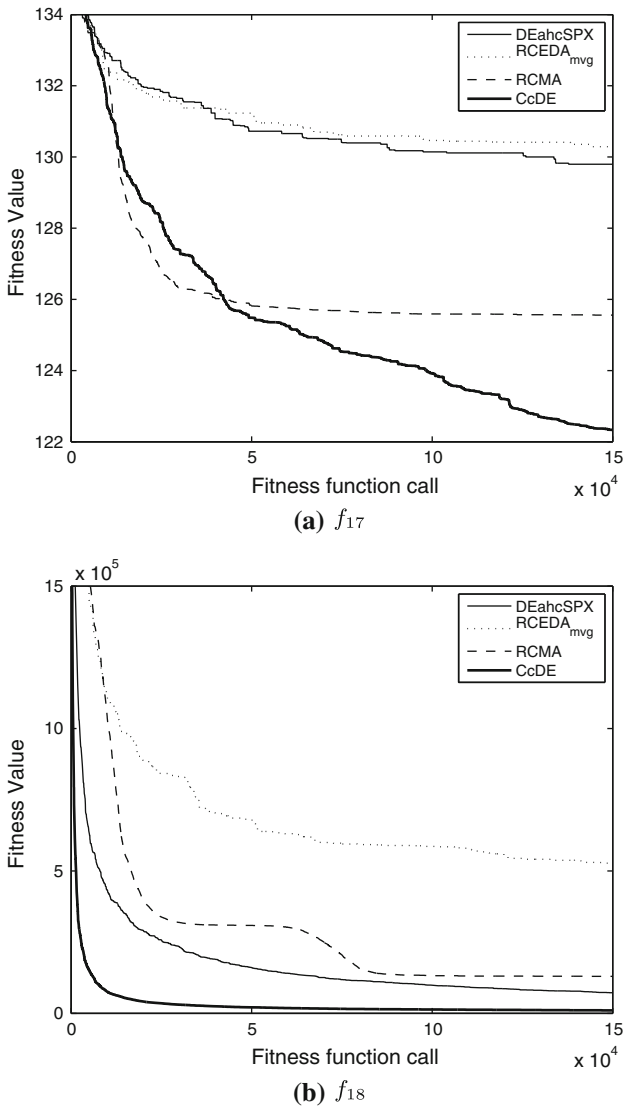
Numerical results in terms of average final values and Wilcoxon test are shown in Tables 4 and 5, respectively. Numerical results show that the proposed CcDE is very promising and competitive with modern and complex population-based algorithms. More specifically, CcDE clearly outperforms, for the selected problems, RCEDA$_{mvg}$ and DEahcSPX. The RCMA is competitive with CcDE, especially for low dimensional problems, uni-modal, and separable functions, but is on average outperformed by the proposed algorithm. According to our interpretation, the RCMA succeeds at outperforming the proposed CcDE in some cases thanks to the employment of the local search. It can be clearly seen that when the function is uni-modal (as for example for f_1), the local search employment during the early generations allows the quick solution of the problem. This consideration can be extended to those fitness landscapes which, albeit multi-modal, are characterized by a strong globally optimal basin of attraction, for example f_{14}. A further improvement to our proposed scheme is the integration of local search units within the ring topology. These local search units will allow an enhancement in robustness and a high performance also in the case of uni-modal and separable functions. Figure 6 shows average performance trends of the four considered algorithms over a selection of the test problems considered in this study.

Table 5 Wilcoxon Rank-Sum test for population-based algorithms

Test problem	RCEDA$_{mvg}$	RCMA	DEahcSPX
f_1	+	−	+
f_2	=	−	+
f_3	+	−	+
f_4	+	−	+
f_5	+	−	=
f_6	+	+	+
f_7	+	+	+
f_8	+	−	+
f_9	+	−	+
f_{10}	=	+	+
f_{11}	+	+	+
f_{12}	+	−	+
f_{13}	=	+	+
f_{14}	+	−	+
f_{15}	+	−	+
f_{16}	+	+	+
f_{17}	+	+	+
f_{18}	+	+	+
f_{19}	=	=	+
f_{20}	=	+	+
f_{21}	=	+	+
f_{22}	+	+	+
f_{23}	+	+	+
f_{24}	+	+	=
f_{25}	=	+	=
f_{26}	=	+	=

3.3 Algorithmic functioning

In order to better explain the working principle of the proposed CcDE, the following experiment has been performed. For a selected function, f_{18}, CcDE has been run with $N_c = 3$ and all the other parameters as reported above. Under these conditions, the evolution of the three corresponding elite solutions has been studied. Figure 7



(a) f_{17}

(b) f_{18}

Fig. 6 Performance trends of CcDE and population-based algorithms

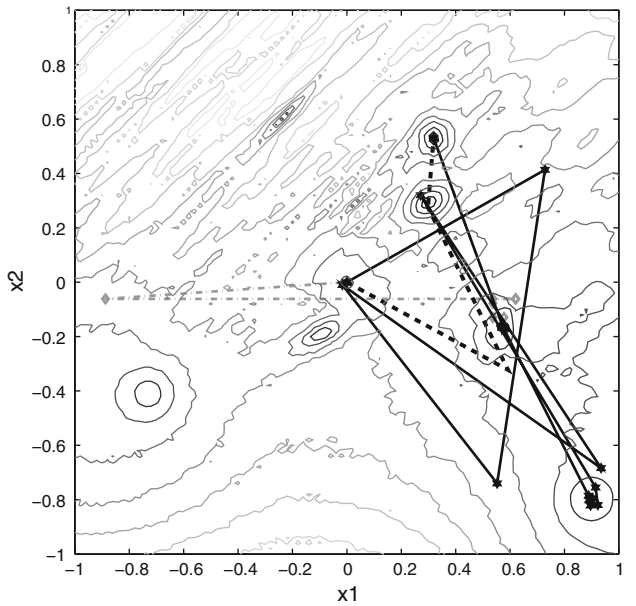


Fig. 7 Example of algorithmic functioning for a CcDE with 3 compact units. The three lines, *black solid*, *black dashed*, and *grey dashed*, represent the evolution of the elites within each compact unit. The three evolutions start from (0,0), explore different areas, and eventually reach the same solution in (0.3191,0.5288)

reports the result of our study. As explained above the domain is normalized in the domain $[0, 1]^2$. The three elite solutions start from the origin (0, 0) and perform their evolutions, marked with black solid line, black dashed line, and grey dashed line. Eventually, the three evolutions end up in the point (0.3191, 0.5288). It can be observed that the three elite solutions follow different pathways while exploring the decision space and that these pathways, due to the effect of the migration, cross in some points. The three compact units eventually tend to focus their search in the same promising subregion of the decision space and detect at the end of the search the same promising solution.

4 Conclusion

This paper has proposed a novel algorithmic structure for solving global optimization problems. The proposed CcDE is composed of a set of compact units. Each unit is a compact optimizer based on DE logic. The various units have the role of locally exploring the decision space from different perspectives. The multiple local search is coordinated and supervised by a self-adaptive logic which promotes the migration, and thus the propagation, of the most promising genotypes, as well as the propagation of the search rules displaying the highest performance. The latter effect is carried out by adapting to compact structure the scale factor inheritance mechanism for distributed systems. In this sense, the proposed CcDE can be seen as a parallel algorithm where virtual populations play the role of actual population and perform the search, simultaneously. On the other hand, CcDE can also be seen as a population-based algorithm where each element is a search strategy. The search strategies interact and evolve towards the detection, for each specific phase of the evolution, of a suitable search logic and finally the search of the global optimum. The proposed algorithm, despite its simplicity, displays a good performance on a set of various test problems with respect to compact algorithms and modern population-based meta-heuristics. The success of CcDE is partly due to the intuition that compact optimizers have a behaviour more similar to local search rather than global search. For this reason the study of algorithmic structures employing compact units and a supervising rule is, in our opinion, a promising research topic in evolutionary optimization. Future works will investigate structures composed of diverse compact units (e.g. employing different mutation strategies) and more advanced supervising models.

References

1. Larrañaga P, Lozano JA (2001) Estimation of distribution algorithms: a new tool for evolutionary computation. Kluwer, Norwell
2. Harik GR, Lobo FG, Goldberg DE (1999) The compact genetic algorithm. IEEE Trans Evol Comput 3(4):287–297
3. Rastegar R, Hariri A (2006) A step forward in studying the compact genetic algorithm. Evol Comput 14(3):277–289
4. Harik G (1999) Linkage learning via probabilistic modeling in the ECGA. Technical Report 99010, University of Illinois at Urbana-Champaign, Urbana, IL
5. Harik GR, Lobo FG, Sastry K (2006) Linkage learning via probabilistic modeling in the extended compact genetic algorithm (ECGA). In: Pelikan M, Sastry K, Cantú-Paz E (eds) Scalable optimization via probabilistic modeling vol 33 of studies in computational intelligence. Springer, pp 39–61
6. Sastry K, Goldberg DE (2000) On extended compact gentic algorithm. Technical Report 2000026, University of Illinois at Urbana-Champaign, Urbana, IL
7. Sastry K, Xiao G (2001) Cluster optimization using extended compact genetic algorithm. Technical Report 2001016, University of Illinois at Urbana-Champaign, Urbana, IL
8. Sastry K, Goldberg DE, Johnson DD (2007) Scalability of a hybrid extended compact genetic algorithm for ground state optimization of clusters. Mater Manuf Process 22(5):570–576
9. Aporntewan C, Chongstitvatana P (2001) A hardware implementation of the compact genetic algorithm. In: Proceedings of the IEEE congress on evolutionary computation, 1:624–629
10. Gallagher JC, Vigraham S, Kramer G (2004) A family of compact genetic algorithms for intrinsic evolvable hardware. IEEE Trans Evol Comput 8(2):111–126
11. Jewajinda Y, Chongstitvatana P (2008) Cellular compact genetic algorithm for evolvable hardware. In: Proceedings of the international conference on electrical engineering/electronics, computer, telecommunications and information technology, 1:1–4

12. Gallagher JC, Vigraham S (2002) A modified compact genetic algorithm for the intrinsic evolution of continuous time recurrent neural networks. In: Proceedings of the genetic and evolutionary computation conference, pp 163–170
13. Ahn CW, Ramakrishna RS (2003) Elitism based compact genetic algorithms. IEEE Trans Evol Comput 7(4):367–385
14. Rudolph G (2001) Self-adaptive mutations may lead to premature convergence. IEEE Trans Evol Comput 5(4):410–414
15. Mininno E, Cupertino F, Naso D (2008) Real-valued compact genetic algorithms for embedded microcontroller optimization. IEEE Trans Evol Comput 12(2):203–219
16. Cupertino F, Mininno E, Naso D (2006) Elitist compact genetic algorithms for induction motor self-tuning control. In: Proceedings of the IEEE congress on evolutionary computation, pp 3057–3063
17. Cupertino F, Mininno E, Naso D (2007) Compact genetic algorithms for the optimization of induction motor cascaded control. In: Proceedings of the IEEE international conference on electric machines and drives, 1:82–87
18. Fossati L, Lanzi PL, Sastry K, Goldberg DE (2007) A simple real-coded extended compact genetic algorithm. In: Proceedings of the IEEE congress on evolutionary computation, pp 342–348
19. Lanzi P, Nichetti L, Sastry K, Goldberg DE (2008) Real-coded extended compact genetic algorithm based on mixtures of models. In: Linkage in evolutionary computation, vol 157 of studies in computational intelligence. Springer, pp 335–358
20. Mininno E, Neri F, Cupertino F, Naso D (2011) Compact differential evolution. IEEE Trans Evol Comput (to appear)
21. Neri F, Tirronen V (2010) Recent advances in differential evolution: a review and experimental analysis. Artif Intell Rev 33(1):61–106
22. Neri F, Mininno E (2010) Memetic compact differential evolution for cartesian robot control. IEEE Comput Intell Mag 5(2):54–65
23. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13:398–417
24. Tasoulis DK, Pavlidis NG, Plagianakos VP, Vrahatis MN (2004) Parallel differential evolution. In: Proceedings of the IEEE congress on evolutionary computation, pp 2023–2029
25. Weber M, Neri F, Tirronen V (2009) Distributed differential evolution with explorative-exploitative population families. Genet Program Evol Mach 10(4):343–371
26. Weber M, Tirronen V, Neri F (2010) Scale factor inheritance mechanism in distributed differential evolution. Soft Comput Fusion Found Method Appl 14(11):1187–1207
27. Gautschi W (1972) Error function and fresnel integrals. In: Abramowitz M, Stegun IA (eds) Handbook of mathematical functions with formulas, graphs, and mathematical tables, Chap. 7. pp 297–309
28. Cody WJ (1969) Rational chebyshev approximations for the error function 23(107):631–637
29. Price KV, Storn R, Lampinen J (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin
30. Feoktistov V (2006) Differential evolution in search of solutions. Springer
31. Lampinen J, Zelinka I (2000) On stagnation of the differential evolution algorithm. In: Oŝmera P (eds) Proceedings of 6th international mendel conference on soft computing, pp 76–83
32. Brest J, Greiner S, Bošković B, Mernik M, Žumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10(6):646–657
33. Zhang J, Sanderson AC (2009) Jade: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13(5):945–958
34. Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, Tiwari S (2005) Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report 2005005, Nanyang Technological University and KanGAL, Singapore and IIT Kanpur, India
35. Vesterstrøm J, Thomsen R (2004) A comparative study of differential evolution particle swarm optimization and evolutionary algorithms on numerical benchmark problems. In: Proceedings of the IEEE congress on evolutionary computation, 3:1980–1987
36. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3:82–102
37. Igel C, Suttorp T, Hansen N (2006) A computational efficient covariance matrix update and a (1 + 1)-CMA for evolution strategies. In: Proceedings of the genetic and evolutionary computation conference. ACM Press, pp 453–460
38. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics Bull 1(6):80–83
39. Yuan B, Gallagher M (2005) Experimental results for the special session on real-parameter optimization at cec 2005: a simple, continuous eda, pp 1792–1799
40. Molina D, Herrera F, Lozano M (2005) Adaptive local search parameters for real-coded memetic algorithm. In: Proceedings of the IEEE congress on evolutionary computation, pp 888–895
41. Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12(1):107–125