Applied Soft Computing 22 (2014) 585–604

Contents lists available at ScienceDirect

Applied Soft Computing

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

Group counseling optimization

M.A. Eita*, M.M. Fahmy

Tanta University, Egypt

ARTICLE INFO

Article history:
Received 29 September 2012
Received in revised form 7 February 2014
Accepted 30 March 2014
Available online 18 April 2014

Keywords:
Optimization
Group counseling
Particle swarm optimizer
NBIPOP-aCMA-ES
Benchmark test functions
Real-world application

ABSTRACT

In this paper, a new population-based optimization algorithm – which we call a group counseling optimizer (GCO) – is developed. Instead of mimicking the behavior of living organisms such as birds, fish, ants, and bees, we emulate the behavior of human beings in life problem solving through counseling within a group. This is motivated by the fact that the human's thinking is often predicted to be the most reasonable and influential. The inspiration radiates from the various striking points of analogy between group counseling and population-based optimization which we have discovered, as elucidated in Section 2. The algorithm is tested using seven unrotated benchmark functions and five rotated ones. Further, a comparison is made with the comprehensive learning particle swarm optimizer (CLPSO) which outperforms many other variants of the particle swarm optimizer. Using new eight composition benchmark functions, another comparison is made with the BI-population covariance matrix adaptation evolution strategy with alternative restart strategy (NBIPOP-$_a$CMA-ES) which is the winner of the competition on real-parameter single objective optimization at IEEE CEC-2013. The results are all highly promising, demonstrating the soundness and efficacy of the proposed approach. GCO is applied to real-world application which is spacecraft trajectory design problem. Also, the results show that GCO outperforms well-known optimizers.

1. Introduction

Optimization is the computational discipline devoted to the study of the 'best' solution of a problem [1,2]. It, mathematically, means either minimization or maximization of a certain objective function. Losses and drawbacks are to be minimized, whereas profits and merits are to be maximized. All of us seek optimum or, at least suboptimum, solutions because we often aspire to a better way of life. It is no exaggeration to assert that looking for a solution of an optimization problem is as old as human history itself.

A great many optimization approaches have been developed and consolidated over the decades. From 1960 onwards, attention has been particularly focused on the category of population-based optimizers [3]. A prominent property of the computational algorithms of this category is that an iterative policy is followed, which relies on a group, or population, of candidate solutions, not just one solution. During the iterations, a population of constant size is maintained, and the group of solutions is improved progressively. The adoption of successive solution groups is advantageous in that working in groups is generally more productive than individual efforts. Specifically, improving a solution in an iteration can benefit from other solutions in the group, in the sense that the new value of a solution (to be used in the next iteration) can be deduced through 'interaction' or 'cooperation' with other solutions in an algorithm-dependent manner. Having a group of solutions 'working together' is the key to the development of modern biology-inspired optimizers, in which the behavior of biological organisms is emulated. In other words, a biological 'metaphor' does stimulate the algorithm. In what follows, we refer to five famous example algorithms and their pertinent metaphors.

The genetic algorithm (GA) [4,5] emulates genetic evolution in biological organisms according to the theory of the Charles Darwin. It depends on the construction of an evolutionary computation scheme, using models of evolutionary processes such as 'natural selection', 'survival of the fittest', and 'reproduction'. In a world with limited resources, each individual enthusiastically competes with others for survival. Individuals having the best traits are more likely to survive and reproduce, and these traits will be passed on to their offspring. With time, desirable qualities are inherited from generation to generation, and they turn dominant in the population.

The particle swarm optimization (PSO) [6–8] began as a computer simulation of the social behavior of biological organisms living in groups such as a flock of birds, and a school of fish, where

* Corresponding author. Tel.: +20 403306059.
E-mail addresses: Mohammad.Eita@ejust.edu.eg, eta1232002@yahoo.com (M.A. Eita), mfn_288@hotmail.com (M.M. Fahmy).

http://dx.doi.org/10.1016/j.asoc.2014.03.043
1568-4946/

no leader can be recognized. Within such social groups, individuals are not very knowledgeable about the overall behavior of the group, nor are they fully aware of their environment. But they do have the capability of gathering as well as travelling and managing together, without any collision or apparent conflict. In doing all this, a plain principle is obeyed: imitation of successful activities of neighboring individuals. Intrinsic local interaction among individuals brings about intricate, graceful behavior that characterizes bird flocking, fish schooling, collective foraging, and many other aspects of living. In analyzing group dynamics of bird social behavior, inter-individual distances play a major role; that is, the synchrony of flocking behavior is conceived to hinge on the effort exerted by the birds to preserve optimum separation between themselves and their neighbors, so that cooperation of individuals becomes feasible and effective.

The ant colony optimization (ACO) [9] is based on the foraging behavior of ants. Such social insects are capable of finding the shortest path between their nest and a food source, with no visible, centralized coordinating mechanism. There exists an initial chaotic activity pattern in the search for food but, once a food source is located, activity patterns become well organized and ant groups come to go along the shortest path heading for the food source. In an infinitesimal time interval, all ants follow the same path. Here, through an orderly recruitment process, the ants that discovered a food source direct other ants toward it. Most ants indirectly communicate with each other by means of secreting a chemical scented substance called pheromone. When an ant locates a food source, it carries a food item to the nest and deposits pheromone along the trail. Forager ants decide which path to select on the basis of the concentration of pheromone on the various paths. The path with higher pheromone concentration has a greater probability of being selected. As more ants follow a specific path, the desirability of that path is strengthened by extra pheromonal secretion of the foragers, and thus more and more ants are attracted to it.

The artificial bee colony (ABC) optimization [10–12] is based on the foraging behavior of honey bees. A forager bee leaves the hive looking for a rich food source, a patch of flowers, to gather nectar from. For multiple food sources, forager bees are allocated among different flower patches in such a way as to maximize total nectar intake. The bee stores the nectar in her honey stomach, and a honey-making process begins with secreting an enzyme on the nectar. On returning to the hive, the bee unloads the nectar into empty honeycomb cells, and some extra enzymes are added to avoid fermentation and bacterial attacks. Then, the forager bee that has found a food source performs attractive movements, visualized to as a 'dance', around the place of the comb. Through dancing, she announces her information about the food source, such as how plentiful it is and where it is located. Other bees touch her with their sensory antennas and learn, moreover, the scent and taste of the food of the source. In this way, groups of bees are recruited to exploit the same source. ABC was improved to be able to detect the global optima *via* a lot of research works. Some of newly improvements can be found in [13,14].

The differential evolution algorithm (DE), which is considered an extension to GA, was originated by Storn and Price in 1997, for minimization problems in terms of a cost function [15]. It participated in the First International IEEE Competition on Evolutionary Optimization (ICEC'96), and proved to be the fastest evolutionary algorithm at the time (although it came third among deterministic methods). In this algorithm, use is made of concepts of mutation, crossover, and selection, but with specific mathematical definitions, generally different form those of the genetic algorithm. The guiding principle is that information from within a population of parameter vectors is utilized to produce a new vector population of the same size. The scheme of computations depends on using a weighted difference vector of two randomly chosen vectors so as to vary (perturb) some other third vector. The perturbation is done for every population vector, without resort to a predefined probability distribution function. This is the process of mutation in differential evolution. The vector resulting after perturbation is a mutant vector. Some of the components of the mutant vector are 'mixed' with some components of the target vector to form the so-called trial vector. This is the process of crossover. Over the past years, researchers enhanced the DE behavior through various ideas. Some of the recent enhancements exist in [16,17].

The above-mentioned algorithms, and many others, are useful and the accompanying metaphors are interesting. Yet the field of computational optimization is extensive and still open for further research work and advanced ideas with no foreseeable end. In the present paper, we do not concern ourselves with improving or even overcoming shortcomings of any one of these algorithms. Our main aim is to introduce a new population-based optimization algorithm inspired by an utterly different metaphor. Instead of mimicking the behavior of living organisms such as birds, fish, ants, or bees, we emulate the behavior of *human beings* in life problem solving through *counseling within a group* [18–20]. This is motivated by the fact that the human's thinking is, or should be, the most reasonable and influential. The subject of counseling is well known in sciences like psychology and sociology, but may seem rather obscure in computational optimization. The contribution of this paper is therefore twofold. First, we investigate some of the basic counseling concepts and procedures in an attempt to make counseling emerge as a convincing and appealing metaphor for population-based computational optimization. In this conceptual framework, we identify *twenty* striking items of significant analogy. Second, we utilize these metaphoric items to develop what we call a group counseling optimizer (GCO). The proposed algorithm is compared with the comprehensive learning particle swarm optimizer (CLPSO) [21], which outperforms several variants of the particle swarm optimizer, through use of seven unrotated benchmark functions and five rotated ones. An additional comparison is made with the BI-population covariance matrix adaptation evolution strategy with alternative restart strategy (NBIPOP-$_a$CMA-ES) [22,23], which is the winner of the competition on real-parameter single objective optimization at IEEE CEC-2013 [24], using new eight composition benchmark functions. Results, including error values and convergence characteristics, obtained for GCO are highly satisfactory, demonstrating that the link we have established between group counseling and computational optimization is healthy, authentic, and valuable. We point out that a preliminary version of GCO, with unrotated benchmark functions alone, has been published in [25]. A multi-objective version of GCO is recently published in [26], which gives promising results in solving multi-objective optimization problems. To test the applicability, GCO is applied to a real-world application. Also, the results show that GCO outperforms well-known optimizers.

The remainder of the paper is organized as follows: Section 2 introduces the analogy items between group counseling and the population-based optimization. In Section 3, the proposed algorithm, based on group counseling is introduced and the steps of GCO algorithm are explained in details. In Section 4, algorithmic comparison between GCO and other optimizers is presented. In Sections 5–7 the results of the experiments conducted on seven unrotated and five rotated and eight composition benchmark functions are given. A real world application of GCO is presented in Section 8. The conclusions are finally discussed in Section 9.

2. Analogy between group counseling and population-based optimization

People with problems often seek out another person as a sounding board: someone with whom they can talk over their

problems, experiment with various solutions and finally reach some resolution. Examples of this approach are seen when people have relationship difficulties or want to change jobs or places of residence. The person, for instance, who wants to change his or her job may be advised by another person who has experience of job opportunities that exist in related careers [18]. Hence, people start to seek help when they should make a decision or solve a problem.

Counseling can be thought of as a process of problem solving [19]. Individual counseling is an activity in which one person is helping (counselor) and one is receiving help (counselee) and in which the emphasis of that help is on enabling the other person to find solutions to problems [18]. However, individuals function most of their lives within groups. So, instead of the individual counseling there is another kind of counseling called group counseling [20] that offers the unique advantages of providing group members with the opportunity to discover that their peers also have problems and to learn new ways of resolving problems by observing other members in the group deal with those problems. Unlike individual counseling relationships, a group provides each individual the opportunity to give as well as to receive help.

In the group, the members can discover that they are capable of understanding, accepting, and helping their peers, and that they can contribute to another person's life. Thus, members gradually begin to understand and accept themselves. The emerging trust in self and others facilitates the sharing of ideas and behaviors in a safe testing ground before applying those ideas and behaviors in relationships outside the group.

Group members come to function not merely as counselees, but they practically behave as counselees at certain times in the sessions and as counselors at other times. Unlike individual counseling, where information and care flow in a single direction, in a group, the flow of information and care is multi-directional, where each member participates in the giving and receiving of advice.

Population-based computational optimization approaches are widely used in practical applications especially when the objective function is complex and the solution vector of parameters is of high dimensionality. Reflecting on the nature and procedural phases of counseling among people, we can infer that there exist many highly interesting aspects of *analogy* between a group counseling process and a population-based optimization process, in spite of being in completely different settings. Table 1 lists prominent points of resemblance between the two processes that well justify the adoption of group counseling as a *metaphor* for population-based optimization. This is the first scientific contribution of this paper.

3. The proposed GCO algorithm

In this section, we develop a population-based, gradient-free, optimization algorithm – which we call a group counseling optimizer and abbreviate it as GCO. This is the second scientific contribution of the present paper. Instead of mimicking the behavior of biological organisms such as birds, fish, ants, and bees, our development is inspired by the human social behavior in solving life problems through counseling within a group, as explained in some detail in Section 2. The various intriguing points of analogy, or resemblance, between a group counseling process and a population-based optimization process, consolidated and itemized in Table 1, pave the way for a novel optimization approach with group counseling being the metaphor. We will here concentrate on the computational scheme of the proposed approach and its characteristics from the optimization point of view. Reference, when significant, will be made to the metaphoric items of Table 1. When we say, for instance, *cf.* Item 1 (problem solving), we mean to refer to item 1, entitled 'problem solving', in Table 1, for comparison of a certain computational aspect with a resemblant group-counseling aspect regarding problem solving.

The problem under investigation is a single-objective, multivariate, unconstrained, continuous optimization problem. Given is a scalar objective function $f(X)$, where X is a set of D components (variables),

$$X = (x_1, x_2, \ldots x_D) \tag{1}$$

which can also be expressed as a D-dimensional vector,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_D \end{bmatrix} \tag{2}$$

The form (1) of X will often be used because we will be interested in the individual components x_d, $d = 1, 2, \ldots, D$, rather than the vector X as a whole, Eq. (2). It is required to optimize $f(X)$ by means of a suitably selected set X^*,

$$X* = (x_1^*, x_2^*, \ldots, x_D^*) \tag{3}$$

Each component x_d has its own prespecified interval of values in which the component is allowed to vary. This interval is considered the *search range* of x_d and is expressed as $[x_{d\min}, x_{d\max}]$, where $x_{d\min}$ and $x_{d\max}$ are the minimum and maximum allowable values of x_d, respectively. This means that the *length* of the range of variation of x_d, denoted by $range\_length_d$, is

$$range\_length_d = x_{d\,\max} - x_{d\,\min} \tag{4}$$

The problem at hand is to be computationally solved in the conceptual framework of the solution of a life problem through group counseling; *cf.* Item 1 (problem solving). The optimizing vector X^* and the corresponding global optimum $f(X^*)$ are obtained as the most appropriate way out of a difficulty is found in group counseling; *cf.* Item 2 (purpose of solution). In general, there is no one unique method for achieving optimality. The parameters of the approach taken are problem-dependent, in both optimization and group counseling; *cf.* Item 3 (method of solution).

Like other heuristics-guided approaches, we begin with a *population size*, m; that is, a certain number m of initial candidate solution vectors X^i in the D-dimensional search space,

$$X^i = (x_1^i, x_2^i, \ldots, x_D^i), \quad i = 1, 2, \ldots, m \tag{5}$$

This vector subspace corresponds to the minisociety of participating members (persons) in group counseling; *cf.* Item 5 (work-force). Such solution vectors are to be improved in increments through successive iterations, as counseling is incrementally effected through successive counseling sessions; *cf.* Item 4 (increments of solution). This means that an iteration with m solution vectors is visualized as a group counseling session with m participating members. Member i is represented by a vector X^i, which in turn accommodates D components x_d^i, $d = 1, 2, \ldots, D$, designating what we consider the best experiences (so far) gained by the member. In solution improvement, candidate vectors therefore use their best possible positional values as group members use their best possible experiences in doing counseling; *cf.* Item 7 (best possible participation). Also, a key idea is that candidate vectors contribute to the improvement of a solution (act as counselors) at times and receive improving contributions (act as counselees) at other times; *cf.* Item 6 (role of participants). All candidate vectors are indeed eligible to give and receive contributions; *cf.* Item 8 (equal chances).

It is particularly noted that the representation of a specific member generally varies from iteration to iteration [experiences vary (generally improve) from session to session]. A candidate vector is

Table 1
Items of analogy between group counseling and population-based optimization.

Item	Group counseling	Population-based optimization
Problem solving	A life problem is to be solved.	A computational problem is to be solved.
Purpose of solution	The most appropriate way out of a difficulty is sought.	The optimum value of an objective function is sought.
Method of solution	There is no one specific method for doing counseling. The approach varies from one situation to another.	There is no one specific method for achieving optimality. The approach is problem-dependent.
Increments of solution	Counseling is effected in increments through a number of successive counseling sessions.	Optimization is effected in increments through a number of successive computational iterations.
Work-force	Counseling is carried out by a group of participating members, a minisociety.	Optimization is carried out by a population of candidate vectors, a subspace of the real vector space.
Role of participants	Group members act as counselors at times and as counselees at other times.	Candidate vectors contribute to the improvement of a solution at times and receive improving contributions at other times (see Section 3).
Best possible participation	Group members use their best possible experiences in doing counseling.	Candidate vectors use their best possible positions in solution improvement.
Equal chances	All group members are eligible to give and receive help.	All candidate vectors are eligible to give and receive contributions. Contributing vectors are chosen according to a uniform distribution probability measure. (see Section 3)
Active members	Not every group member necessarily participates in doing counseling – only verbally active members. Also, participating members vary with the variation of problems being discussed.	Not every candidate vector necessarily contributes to the improvement of vectors – only a subset of vectors chosen according to a probability measure. Also, contributing vectors vary with the variation of vector components being improved (see Section 3).
Brainstorming	Good ideas provided by counselors may be combined together to form a better idea.	Good contributions from candidate vectors may be added together in specific proportions to form a better contribution (see Section 3).
From general terms to details	Life problems are first discussed in general terms, then fine details are dealt with.	The search technique proceeds first in relatively large steps in an exploration stage, and then in small steps in an exploitation stage.
Thorough investigation	Immature solutions are to be avoided.	Local optima should not be trapped in; premature convergence is to be avoided.
Problem complication	Multidimensional life problems need a compromise between conflicting solutions.	Multi-objective function optimization needs a compromise between conflicting solutions.
Inferior solution	Counseling, in the domain of support, can only help to contain the counselee's distress.	A suboptimum solution can, in certain cases, be all what is obtainable.
Dependence on other members	A group member can solve his problem after receiving help from other members.	A candidate vector component can be improved through contributions from other vectors (see Section 3).
Self-dependence	Sometimes, a group member can solve his problem without receiving explicit help from other members – through self-counseling, after self-discovery.	Sometimes, a candidate vector component can improve itself without receiving contributions from other vectors – through self-improvement, according to a probability measure (see Section 3).
Remaining unchanged	Sometimes, a group member is persuaded to stay where he is; otherwise, things will worsen.	Sometimes, a candidate vector is kept unchanged; otherwise, the objective function will take on worse values (see Section 3).
Standardization	Counseling approaches and counselor's competencies conform to standard norms for a trustworthy counseling job.	Optimization approaches are tested using benchmark functions, and rotated benchmark functions, to demonstrate efficiency and robustness.
Judgment	The group progress is evaluated to check that the counseling job has been satisfactorily done.	The objective function is evaluated to check that convergence to the global optimum has successfully occurred.
Termination	The counseling process terminates at the end of sessions. A termination policy is followed.	The optimization process terminates at the end of iterations. A stopping criterion is used.

modified (hopefully improve) *component-wise*. A new value of each component in a vector is produced either by invoking contributions (experiences) of the corresponding components in some (not necessarily all) other vectors, or by modifying (directly) the current value of the component itself. These are two different strategies, each having distinct behavior properties. The situation is again analogous to what happens in group counseling, where a person – in solving his problem – asks other people for help; *cf.* Item 15 (dependence on other members) or, sometimes, he depends on himself only after self-discovery; *cf.* Item 16 (self-dependence). In group counseling, life problems are first discussed in general terms, then fine details are dealt with. Similarly, the search technique in the optimization algorithm proceeds first in relatively large steps in an *exploration* stage, and then in small steps in an *exploitation* stage; *cf.* Item 11 (from general terms to details). The optimization process terminates at the end of iterations using a stopping criterion as the group counseling process terminates at the end of sessions following a termination policy; *cf.* Item 20 (termination). We choose to employ a certain maximum number of evaluations of the objective function to be the stopping criterion. This is the product of the population size m and the maximum number of iterations, *itr_max*.

The proposed GCO algorithm requires a number of effective parameters to be set. We employ four algorithmic parameters:

- Number of group members acting as counselors, c, generally $c \leq m-1$.
- Counseling probability, cp, set in the range $[0,1]$.
- Search range reduction coefficient, *red*, set in the range $[0,1]$.
- Transition rate from the stage of exploration to that of exploitation, tr.

The role of these parameters will become apparent as we proceed.

In the following, the steps of the GCO algorithm are explained in details:

Step 1

The algorithm starts with m D-dimensional initial candidate solution vectors X^i being placed randomly in the search space. We locate the values of the vector components x^i_d, $d = 1, 2, \ldots, D$, in accordance with a beta probability distribution [27–29],

$$\beta(x) = \frac{x^{a-1}(1-x)^{b-1}}{B(a,b)} \quad 0 < x < 1 \tag{6}$$

where the function in the denominator is a beta function defined as

$$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}dt \tag{7}$$

We adopt the case in which the two shaping parameters a and b are equal and less than unity, $a = b < 1$. In such a case, the density function has a symmetric U-shape. This has the consequence that, most probably, the candidate solutions lie near the boundaries of the search space and that the global optimum is well within this candidate solution set.

Step 2

The solution vectors X^i are substituted, respectively, into the objective function $f(X)$, yielding m values for $f(X^i)$, called *fitness values*.

Step 3

This is the first iterative step. For each solution X^i, we produce an alternative solution

$$X'^i = (x'^i_1, x'^i_2, \ldots, x'^i_D) \tag{8}$$

The production process is carried out component-wise. Each component x'^i_d is obtained through one of two counseling strategies:

(a) Other-members counseling.
(b) Self-counseling.

Having set the counseling probability cp to some value in the range [0, 1], we, for each component x^i_d, generate a random number in the range [0, 1] according to a uniform probability distribution. This number is here termed a *counseling decisive coefficient* (cdc). We choose to do other-members counseling when cdc is less than or equal to cp, and do self-counseling otherwise. In what follows, we explain how to calculate x'^i_d, given x^i_d, in each of these strategies.

Step 3a: Other-members counseling ($cdc \le cp$)

In this strategy, member i, or the vector X^i, is regarded as a counselee. It asks for counseling of c other members (counselors), chosen randomly out of the population, so that an alternative (hopefully better) component x'^i_d is obtained; *cf.* Item 9 (active members). The value of x'^i_d is calculated by summing *weighted* values of the corresponding components (best experiences) of the c counselors. These are the contributions of the relevant counselors, in a brainstorming process; *cf.* Item 10 (brainstorming).

The weight, w_q, of component d in counselor q ($q = 1, 2, \ldots, c$) is a random number in the range [0, 1] with a uniform probability distribution. It should be obvious that counselor q is some member i. As indicated in Eq. (9), the c weights sum to unity,

$$\sum_{q=1}^{c} w_q = 1 \tag{9}$$

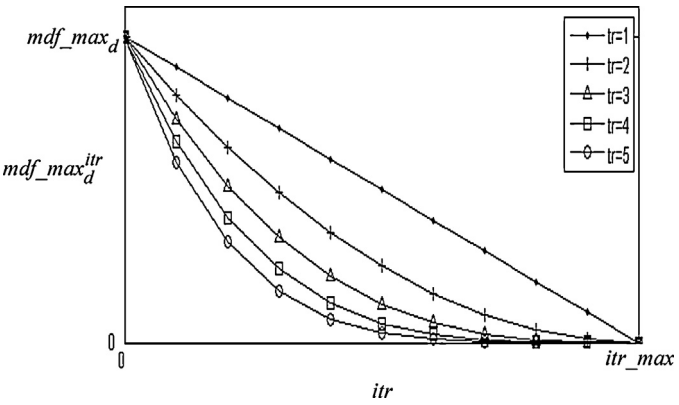

Fig. 1. Effect of transition rate (tr) on modification.

The form of x'^i_d is expressed as

$$x'^i_d = \sum_{q=1}^{c} w_q x^{int\_rand_q}_d \tag{10}$$

which is valid for $d = 1, 2, \ldots, D$ and $i = 1, 2, \ldots, m$ (that is, $m.D$ components). The superscript int_rand_q is an integer random number in the range $[1, m]$ with a uniform distribution. Note that, according to Eq. (10), we generate c such random numbers for each component x^i_d, with a total of $D.m.c$ random numbers for all vector components. The component denoted by $x^{int\_rand_q}_d$ is the value of component d of counselor q (which is member int_rand_q).

It should be evident that the set of c counselors in general varies from component to component (as d varies from 1 to D). A little thought reveals that w_q and int_rand_q are both dependent on the values of i and d; these symbols are not superimposed on Eq. (10) for notational simplicity.

Step 3b: Self-counseling ($cdc > cp$)

In this strategy, an alternative component x'^i_d is obtained in an iteration through modification of the current component x^i_d. That is, the new component x'^i_d depends on the best experience x^i_d of member i with a specific modification.

In the problem statement, each component in the vector X is assigned an overall permissible range of variation. Let the length of this range for component d be denoted by $range\_length_d$. We choose to search for a modification value of the component in a *reduced* range with length $red.range\_length_d$, where red, as mentioned previously, is the search range reduction coefficient, set in the range [0, 1]. The set value of red is kept unchanged for all components of all vectors in all iterations.

Component d, in the various iterations, is allowed to be modified in a *maximum* range $[-mdf\_\max_d, mdf\_\max_d]$ about the current component, where

$$mdf\_\max_d = 0.5.red.range\_length_d \tag{11}$$

Eq. (11) implies that the maximum modification range is divided into two halves about the current component. This results from adding the current component to the modification range.

The maximum modification value *in a certain iteration*, $mdf\_\max^{itr}_d$, is estimated from the relation

$$mdf\_\max^{itr}_d = mdf\_\max_d \left(1 - \frac{itr}{itr\_\max}\right)^{tr} \tag{12}$$

where itr is the iteration number, and itr_max, as defined previously, is the maximum number of iterations. The exponent tr in Eq. (12) refers to a transition rate at which the search method transfers from exploration to exploitation. Relation (12) is a *rule of thumb*, supported by the mathematical illustration of Fig. 1, which shows the

variation of $mdf\_\max_d^{itr}$ from $mdf\_\max_d$ (at the very beginning of iterations) to zero (at itr_max) for different values of tr. It is seen that at a certain iteration, the value of $mdf\_\max_d^{itr}$ decreases as tr increases. In other words, as tr increases, exploration tends to exploitation in a smaller number of iterations. Here the value of $mdf\_\max_d^{itr}$ plays a central role in whether the optimization algorithm, in a certain stage, performs exploration or exploitation. It is a well-accepted principle that all optimization algorithms have to *compromise* between exploration and exploitation so that the global optimum is eventually attained.

For each component x_d^i, we generate a random number in the range

$[-mdf_\max_d^{itr},\ mdf_\max_d^{itr}]$and this is added to x_d^i to obtain the modified value x'^i_d of the form

$$x'^i_d = x_d^i + rand_d^i(-mdf\_\max_d^{itr}, mdf\_\max_d^{itr}) \quad (a)$$
$$if(x'^i_d > x_{d\max}) \ \ then \ \ x'^i_d = x_d^i + rand_d^i(0, x_{d\max} - x_d^i) \quad (b) \qquad (13)$$
$$if(x'^i_d < x_{d\min}) \ \ then \ \ x'^i_d = x_d^i + rand_d^i(x_{d\min} - x_d^i, 0) \quad (c)$$

Note that the modification value (the random number) varies from one vector to another in an iteration.

Step 4

Step 2 is repeated for X'^i (instead of X^i) and the fitness value, $f(X'^i)$, is evaluated; *cf.* Item 19 (judgment). If $f(X'^i)$ is better (*less* in minimization or *greater* in maximization) than $f(X^i)$, then X'^i replaces X^i; otherwise, X'^i is ignored and X^i is kept unchanged for possible subsequent improvement; *cf.* Item 17 (remaining unchanged).

Repetition steps (iterations)

Steps 3 and 4 are repeated until the stopping criterion is met.

Final step

This is a decision-making step. The m solutions, resulting from the last repetition step, are compared with each other based on the fitness values of the objective function. The best solution is taken as the optimum solution X^* (with acceptable error).

Fig. 2 shows a general flowchart for the proposed GCO (for the minimization problem).

4. Algorithmic comparison between GCO and other optimizers

The algorithmic comparison between GCO and other optimizer can be summarized in the following points:

1. Like GA, PSO, ACO, ABC, etc., GCO is a population-based optimization algorithm in which an iterative policy is used, depending on a group, or population, of candidate solutions, not just one solution.
2. Like GA, PSO, ACO, ABC, The decision into GCO is made depending on a probability (counseling probability cp). GA uses crossover and mutation probability. ACO utilizes the probability by which the ants choose a route from a node to another node. ABC employs the probability of an onlooker bee to choose to go to the preferred food source at some location.
3. Unlike ABC, the metaphor of GCO, counseling, has no subtle concepts or sophisticated procedures [18–20]. This facilitates the implementation process of the metaphor. Metaphors of ABC have several details, leading to some difficulty in the implementation process.
4. Unlike PSO, GCO does not depend on the best solution of the iteration to enhance the candidate solutions. Thus GCO avoids trapping to local optima (premature convergence).
5. Unlike GA, PSO, ACO, ABC, etc., GCO uses an initialization process according to beta distribution. To the best of our knowledge, there is no optimizer uses the beta distribution in its initialization process.
6. Unlike PSO, GCO represents the candidate solution by only its position. PSO represents the candidate solution by two items: its position and velocity.
7. Unlike GA, GCO does not require to encode the candidate solution to another representation such as binary encoding.
8. Unlike GA, GCO employs one of two strategies to improve the candidate solutions: other-members counseling or self-counseling strategy. GA utilizes two strategies together in the same time to enhance the candidate solutions: crossover and mutation.

5. Test experiments: unrotated benchmark functions

The proposed GCO algorithm is tested in *minimization* problems using seven *unrotated* benchmark functions [30–33]; *cf.* Item 18 (standardization). Two of these functions are of the *unimodal* type:

- Sphere function, $f_1(X)$
- Rosenbrock function, $f_2(X)$and five are of the *multimodal* type:
- Ackley function, $f_3(X)$
- Griewank function, $f_4(X)$
- Weierstrass function, $f_5(X)$
- Rastrigin function, $f_6(X)$
- Schwefel function, $f_7(X)$

The algorithm is then compared with the comprehensive learning particle swarm optimizer (CLPSO) presented by Liang et al. [21], which renders distinguished performance in comparison with many other variants of the particle swarm optimizer.

The definitions of the unrotated benchmark functions used here are given in what follows. Note that the dimension D is left free to be chosen by the designer of the algorithm. Apart from minor exceptions, the function becomes more complicated from the minimization viewpoint as D gets higher.

5.1. Sphere function

A unimodal function,

$$f_1(X) = \sum_{d=1}^{D} x_d^2 \qquad (14)$$

It is simple in form, with no interaction between the variables (components x_d of the vector X).

5.2. Rosenbrock function

A unimode function,

$$f_2(X) = \sum_{d=1}^{D-1} [100(x_d^2 - x_{d+1})^2 + (x_d - 1)^2] \qquad (15)$$

It has strong interaction between some of the variables.

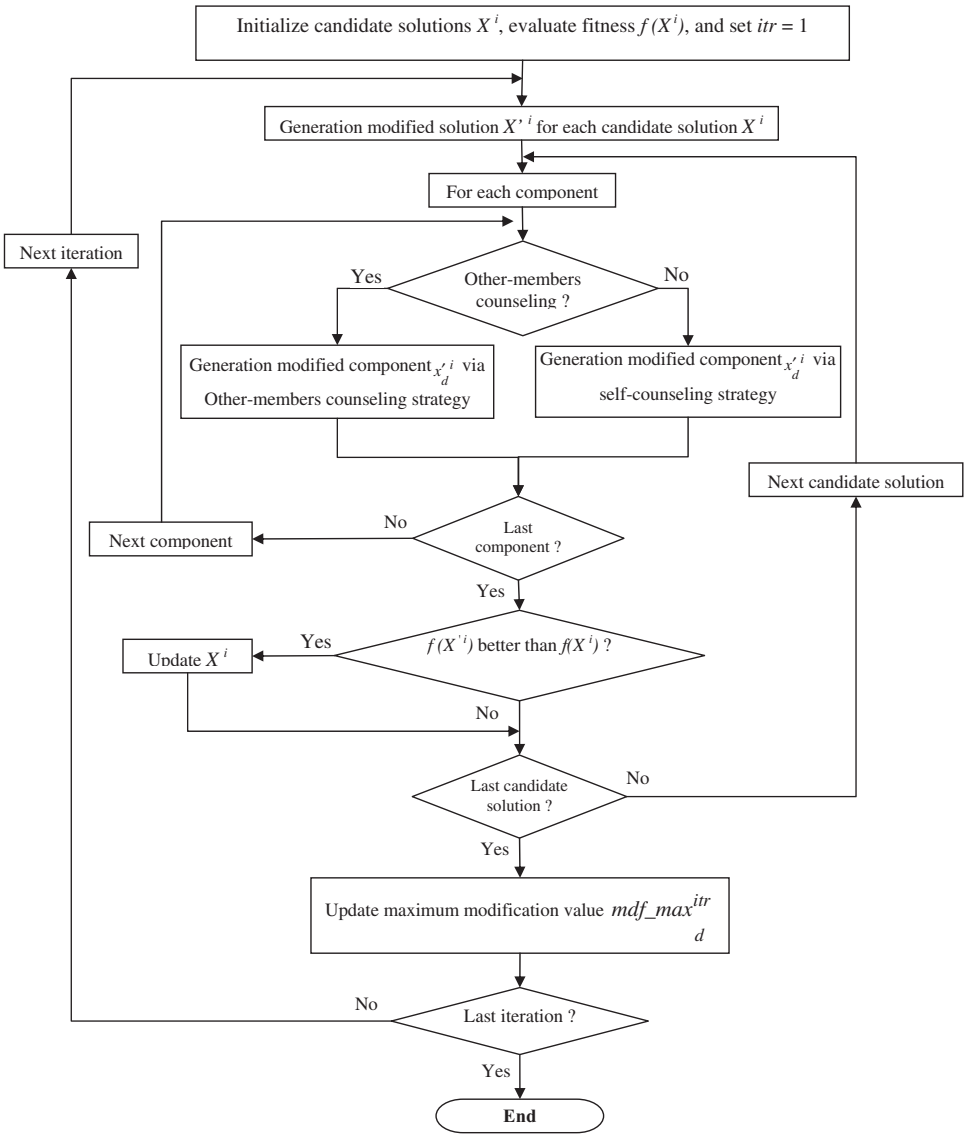

Fig. 2. Flowchart of GCO.

5.3. Ackley function

A multimodal function

$$f_3(X) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{D}\sum_{d=1}^{D} x_d^2}} - e^{\frac{1}{D}\sum_{d=1}^{D}\cos(2\pi x_d)} \quad (16)$$

It has several local minima. The minimization process is facilitated to some extent because the local minima are shallow [21].

5.4. Griewank function

A multimodal function,

$$f_4(X) = 1 + \sum_{d=1}^{D}\frac{x_d^2}{4000} - \prod_{d=1}^{D}\cos\left(\frac{x_d}{\sqrt{d}}\right) \quad (17)$$

It has strong interaction between all the variables. However, it is interesting to note that as the dimension D is increased, this function becomes simpler because the local minima induced by the cosine product term decrease in number and complexity [34].

5.5. Weierstrass function

A multimodal function,

$$f_5(X) = \sum_{d=1}^{D}\left(\sum_{k=0}^{k\max}[\alpha_1{}^k\cos(2\pi\alpha_2{}^k(x_d + 0.5))]\right) - D\sum_{k=0}^{k\max}[\alpha_1{}^k\cos(2\pi\alpha_2{}^k.0.5)] \quad \alpha_1 = 0.5, \alpha_2 = 3, kmax = 20 \quad (18)$$

Despite being continuous, this function is differentiable only on a specific set of points [21].

5.6. Rastrigin function

A multimodal version of the sphere function,

$$f_6(X) = \sum_{d=1}^{D}[10 + x_d^2 - 10\cos(2\pi x_d)] \quad (19)$$

Table 2
Particulars of minimization of unrotated benchmark functions.

Function	Search range	Minimizing vector X^*	Global minimum $f(X^*)$
$f_1(X)$	$[-100, 100]$	$[0, 0, \ldots, 0]$	0
$f_2(X)$	$[-2.048, 2.048]$	$[1, 1, \ldots, 1]$	0
$f_3(X)$	$[-30, 30]$	$[0, 0, \ldots, 0]$	0
$f_4(X)$	$[-600, 600]$	$[0, 0, \ldots, 0]$	0
$f_5(X)$	$[-0.5, 0.5]$	$[0, 0, \ldots, 0]$	0
$f_6(X)$	$[-5.12, 5.12]$	$[0, 0, \ldots, 0]$	0
$f_7(X)$	$[-500, 500]$	$[420.96, 420.96, \ldots, 420.96]$	0

It has a large number of local minima arranged as sinusoidal bumps. The variables are independent.

5.7. Schwefel function

A multimodal function,

$$f_7(X) = 418.9829D - \sum_{d=1}^{D} x_d \sin\left(\sqrt{|x_d|}\right) \quad (20)$$

This function is complex and often difficult to deal with. Its complexity stems from the fact that it possesses deep local minima which are far from the global minimum. The global minimum is located near one corner of the search space [35].

For each of the above unrotated functions, Table 2 gives the search range for the components x_d of the vector X (all components have the same search range), the minimizing vector X^*, and the global minimum $f(X^*)$. Note that all global minima are at the *zero* value.

In the test experiments on unrotated functions, we take:

- Dimension, $D = 30$.
- Population size, $m = 40$.
- Maximum number of evaluations of the objective function, FEs = 200,000; that is, the maximum number of iterations, $itr\_max = 5000$ (stopping criterion).
- Shaping parameters of the beta probability distribution, $a = b = 0.1$.

All experiments are run 30 times. The *mean* and *standard deviation* of the numerical results are recorded. The mean is a measure of the *accuracy* and the standard deviation is a measure of the *robustness* in quantifying the performance of an optimizer [36].

The four parameters: c (number of group members acting as counselors), cp (counseling probability), red (search range reduction coefficient), and tr (transition rate from exploration to exploitation) defined specifically for the GCO algorithm are set at different values according to the test function in question, as indicated in Table 3.

Table 3
GCO parameter settings for unrotated benchmark functions.

Function	c	cp	red	tr
$f_1(X)$	3	0.5	0.25	30
$f_2(X)$	3	0.01	0.01	2
$f_3(X)$	3	0.5	0.25	20
$f_4(X)$	3	0.5	0.25	20
$f_5(X)$	3	0.5	0.25	30
$f_6(X)$	3	0.02	0.25	15
$f_7(X)$	3	0.02	0.25	18

Table 4
Mean and standard deviation of error for unrotated benchmark functions: GCO *vs.* CLPSO.

Function	GCO	CLPSO
$f_1(X)$	3.54E−61 ± 1.03E−60	4.46E−14 ± 1.73E−14
$f_2(X)$	1.34E−03 ± 4.03E−03	2.1E+01 ± 2.98E+00
$f_3(X)$	3.55E−15 ± 0.0	0 ± 0
$f_4(X)$	0.0 ± 0.0	3.14E−10 ± 4.64E−10
$f_5(X)$	0.0 ± 0.0	3.45E−07 ± 1.94E−07
$f_6(X)$	0.0 ± 0.0	4.85E−10 ± 3.63E−10
$f_7(X)$	8.67E−12 ± 7.82E−13	1.27E−12 ± 8.79E−13

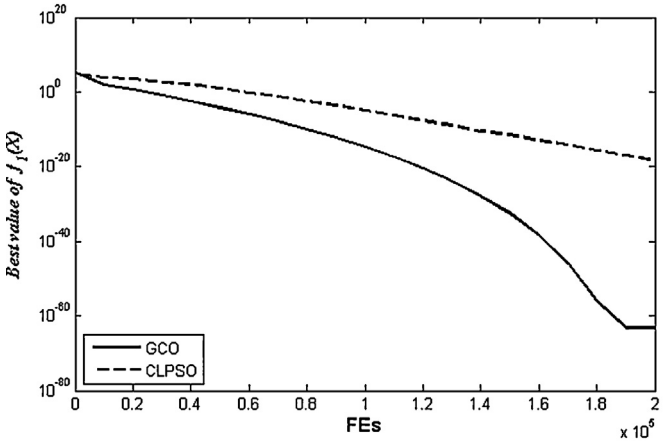

Fig. 3. Convergence for sphere function: GCO *vs.* CLPSO.

5.8. Results of GCO using unrotated benchmark functions

The results of the GCO algorithm when applied to the unrotated benchmark functions $f_1(X)$ through $f_7(X)$ are statistically represented in Table 4. The second column of this table gives the mean and standard deviation (mean ± standard deviation) of the error, defined as the deviation from the zero-value global minimum, of the 30 runs of the algorithm on each function. It turns out that the error resulting from the minimization process of all functions is infinitesimally small; for $f_4(X)$, $f_5(X)$, and $f_6(X)$, a zero-value error is even arrived at. This means that the GCO has successfully converged to the global minima of all unrotated test functions. It has not fallen into any local minimum of multimodal functions $f_3(X)$ through $f_7(X)$, thus avoiding premature convergence; *cf.* Item 12 (thorough investigation). Such a success is attributed mainly to a proper choice of the parameter tr and the value of $mdf\_\max_d^{itr}$, Eq. (12), so that a compromise is smartly made between exploration and exploitation in the course of algorithm running.

The convergence characteristics of the GCO algorithm for the unrotated functions $f_1(X)$ through $f_7(X)$ are demonstrated in Figs. 3–9, respectively. The solid curves in these figures represent the variation of the best (least) function value with the number of function evaluations, FEs (up to 200,000), in the best run. By the 'best run', we refer to the run of the algorithm which yields the best (least) error value at the end of the (5000) iterations. It is seen how each of the test functions takes on successive fitness values that tend to the global minimum (zero value) in a smooth, monotonically decreasing, acceptably fast manner.

5.9. Comparison with CLPSO using unrotated benchmark functions

It is of great importance to compare a new algorithm with a powerful, recent one. Our GCO is compared with the comprehensive learning particle swarm optimizer (CLPSO) [21]. For a

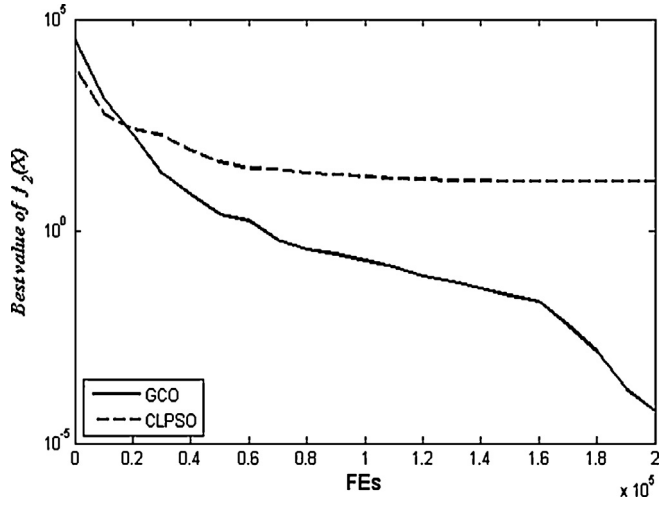

Fig. 4. Convergence for Rosenbrock function: GCO *vs.* CLPSO.

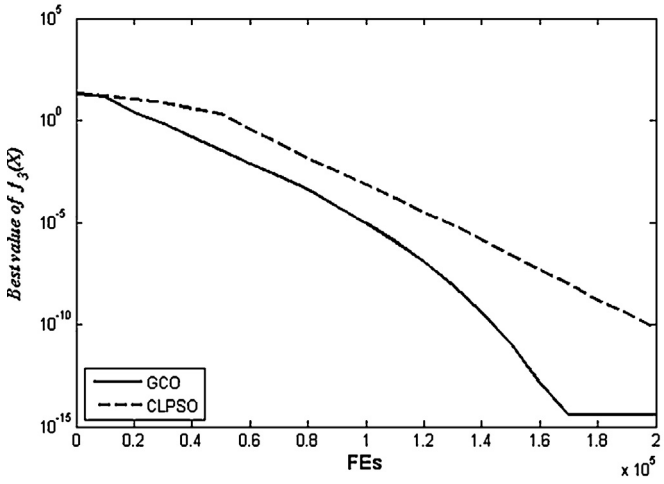

Fig. 5. Convergence for Ackley function: GCO *vs.* CLPSO.

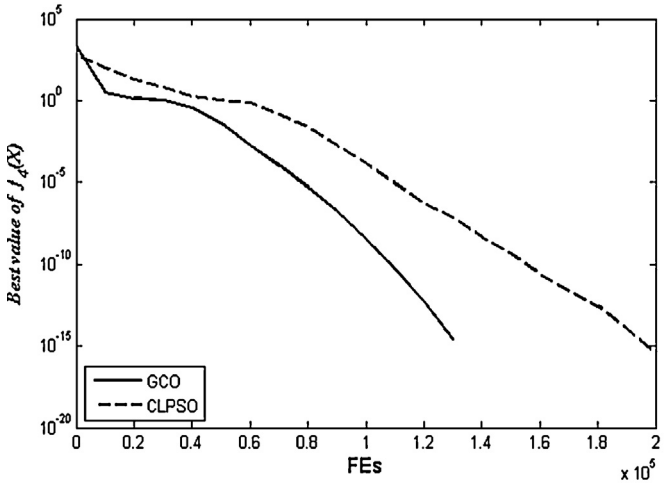

Fig. 6. Convergence for Griewank function: GCO *vs.* CLPSO.

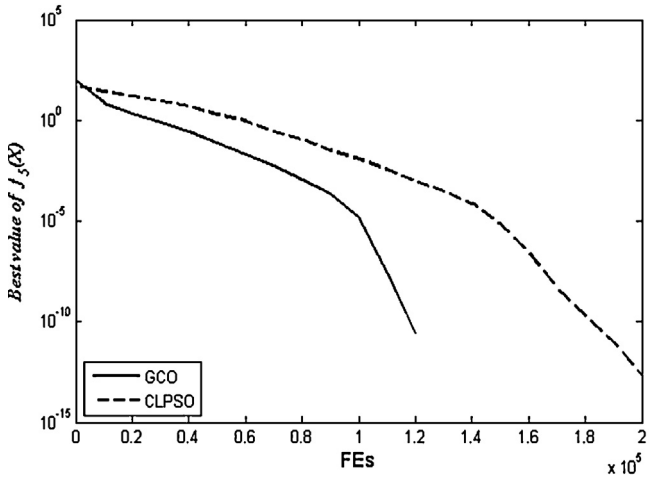

Fig. 7. Convergence for Weierstrass function: GCO *vs.* CLPSO.

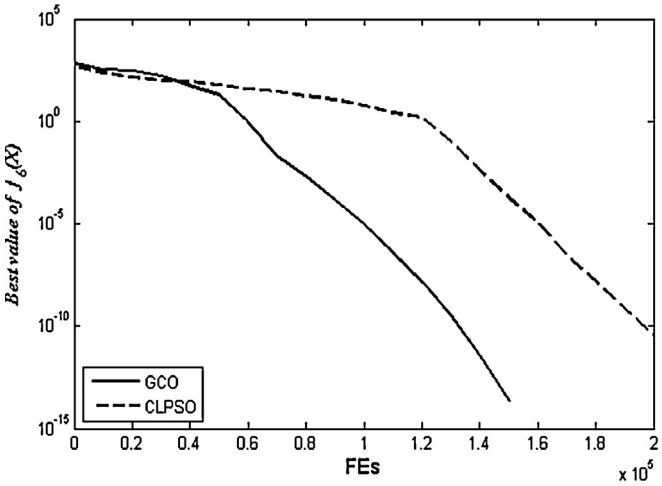

Fig. 8. Convergence for Rastrigin function: GCO *vs.* CLPSO.

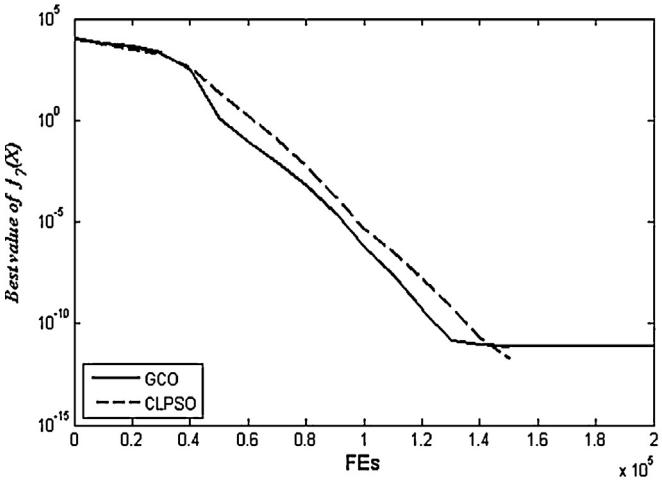

Fig. 9. Convergence for Schwefel function: GCO *vs.* CLPSO.

significant comparison, the same parameter values specified for the CLPSO in [21] are adopted for the GCO: $D = 30$, $m = 40$, FEs = 200,000 ($itr\_max = 5000$), and number of runs = 30. The remaining parameters pertinent to the CLPSO, and having no counterparts in the GCO, are taken as in [21].

Table 4 (the third column from left) gives the mean and standard deviation of the error of the 30 runs of the CLPSO on the unrotated benchmark functions $f_1(X)$ through $f_7(X)$. The result of the GCO is better (having less error) than that of the CLPSO for the five functions $f_1(X)$, $f_2(X)$, $f_4(X)$, $f_5(X)$, and $f_6(X)$, although the differences are

slight. For the two functions $f_3(X)$ and $f_7(X)$, on the other hand, the error of the GCO is above that of the CLPSO, but the error difference is hardly noticeable. In general, anyhow, we do not expect a better performance of the new algorithm on all classes of problems. An enhanced performance on one class can be offset by performance degradation on another class. This is the no free lunch theorem [37]. Figs. 3–9 (the dotted curves) demonstrate the convergence characteristics of the CLPSO for the unrotated functions $f_1(X)$ through $f_7(X)$, respectively. The variation of the best function value with the number of function evaluations, FEs, in the best run, is represented. The GCO curves are all below the CLPSO curves, indicating that the convergence of the GCO to the global minimum is faster than that of the CLPSO. Fast convergence is a noteworthy merit in the assessment of optimization algorithms.

6. Test experiments: rotated benchmark functions

In Section 5, we tested the proposed GCO algorithm using seven unrotated benchmark functions $f_1(X)$ through $f_7(X)$ defined in Eqs. (14)–(20), respectively. In this section, we test the algorithm using five *rotated* multimodal benchmark functions, $f_8(X)$ through $f_{12}(X)$, which are the respective rotation forms of $f_3(X)$ through $f_7(X)$; *cf.* Item 18 (standardization). That is, we have

- Rotated Ackley function, $f_8(X)$.
- Rotated Griewank function, $f_9(X)$.
- Rotated Weierstrass function, $f_{10}(X)$.
- Rotated Rastrigin function, $f_{11}(X)$.
- Rotated Schwefel function, $f_{12}(X)$.

The principle of rotated functions relies on the concept of vector rotation in space [38]. The D-dimensional vector X is left multiplied by an orthogonal $D \times D$ rotation matrix R to form a rotated vector Y as

$$Y = RX \tag{21}$$

where $||Y|| = ||X||$. The rotation matrix R is calculated by Salomon's method [39], which we incorporate in the GCO code; see the Appendix.

The task of using rotated functions in testing an optimization algorithm is more difficult than using unrotated functions [21]. In other words, an algorithm may succeed with unrotated functions and fail with rotated ones. Even when the algorithm succeeds with rotated functions, it is well expected that the resulting error will be much greater than that with unrotated functions. Therefore, rotated-function testing is of special significance to demonstrate the effectiveness and superiority of the algorithm.

The rotated benchmark functions are defined below.

6.1. Rotated Ackley function

$$f_8(X) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{D}\sum_{d=1}^{D} y_d^2}} - e^{\frac{1}{D}\sum_{d=1}^{D}\cos(2\pi y_d)} \tag{22}$$

where y_d represents the components of the rotated vector Y in Eq. (21).

6.2. Rotated Griewank function

$$f_9(X) = 1 + \sum_{d=1}^{D}\frac{y_d^2}{4000} - \prod_{d=1}^{D}\cos\left(\frac{y_d}{\sqrt{d}}\right) \tag{23}$$

6.3. Rotated Weierstrass function

$$f_{10}(X) = \sum_{d=1}^{D}\left(\sum_{k=0}^{k\max}[\alpha_1^k \cos(2\pi\alpha_2^k(y_d + 0.5))]\right) - D\sum_{k=0}^{k\max}[\alpha_1^k \cos(2\pi\alpha_2^k .0.5)] \quad \alpha_1 = 0.5, \alpha_2 = 3, k\max = 20 \tag{24}$$

6.4. Rotated Rastrigin function

$$f_{11}(X) = \sum_{d=1}^{D}[10 + y_d^2 - 10\cos(2\pi y_d)] \tag{25}$$

Note that Eqs. (22)–(25) have the same form as Eqs. (16)–(19) except for writing the components y_d of the rotated vector Y in place of the components x_d of the original vector X.

6.5. Rotated Schwefel function

This is a highly complex function that needs special attention. It is written in the form

$$f_{12}(X) = 418.9829D - \sum_{d=1}^{D} z_d \tag{26}$$

in which

$$z_d = \begin{cases} y_d \sin(\sqrt{|y_d|}) & \text{if } |y_d| \le 500 \quad \text{(a)} \\ -0.001(|y_d| - 500)^2 & \text{if } |y_d| > 500 \quad \text{(b)} \end{cases} \tag{27}$$

and the rotated vector Y becomes

$$Y = R(X - X^*) + X^* \tag{28}$$

where R is an orthogonal $D \times D$ matrix, as in Eq. (21), and X^* is the minimizing D-dimensional vector

$$X* = \begin{bmatrix} 420.96 \\ 420.96 \\ \vdots \\ 420.96 \end{bmatrix} \tag{29}$$

as given in Table 2 for $f_7(X)$. Eq. (28) is used for rotating the original vector X into the vector Y instead of Eq. (21). We remark that Schwefel function has rather easier-to-find minimizing solutions outside the range $[-500, 500]$. Therefore, as Eq. (27b) tells, when $|y_d|$ is greater than the upper range bound 500, z_d is set in a *portion* of the squared distance (difference) between $|y_d|$ and the value 500.

To the best of our knowledge, the optimization algorithms available in the literature are not capable of achieving very high accuracy with the rotated Schwefel function. The computed minimum is often not sufficiently close to the actual global minimum; that is, a relatively big error usually appears at the end of iterations. Therefore, apart from the CLPSO [21], this function is seldom used in algorithm testing.

The search range for the components of the vector X, the minimizing vector X^*, and the global minimum $f(X^*)$ for the rotated functions $f_8(X)$ through $f_{12}(X)$ are the *same* as those for the unrotated functions $f_3(X)$ through $f_7(X)$ specified in Table 2, respectively. That is, the optimum solution for these test functions is not affected by function rotation because the point of global minimum is considered the center of rotation of the function.

It is to be noted that a number of unrotated benchmark functions have regular grids of minima, in which the global minimum

Table 5
GCO parameter settings for rotated benchmark functions.

Function	*c*	*cp*	*red*	*tr*
$f_8(X)$	5	0.7	0.75	15
$f_9(X)$	5	0.7	0.75	13
$f_{10}(X)$	5	0.7	0.75	15
$f_{11}(X)$	5	0.7	0.75	7
$f_{12}(X)$	1	0.45	0.25	3

Table 6
Mean and standard deviation of error for rotated benchmark functions: GCO *vs.* CLPSO.

Function	GCO	CLPSO
$f_8(X)$	3.55E−15 ± 0.0	3.43E−04 ± 1.91E−04
$f_9(X)$	2.48E−16 ± 1.08E−16	7.04E−10 ± 1.25E−11
$f_{10}(X)$	0.0 ± 0.0	3.07E+00 ± 1.61E+00
$f_{11}(X)$	8.95E+00 ± 2.68E+00	3.46E+01 ± 4.59E+00
$f_{12}(X)$	2.12E+02 ± 2.92E+02	1.70E+03 ± 1.86E+02

Table 7
GCO error: Unrotated *vs.* rotated benchmark functions.

Function	Unrotated	Rotated
Ackley	3.55E−15 ± 0.0	3.55E−15 ± 0.0
Griewank	0.0 ± 0.0	2.48E−16 ± 1.08E−16
Weierstrass	0.0 ± 0.0	0.0 ± 0.0
Rastrigin	0.0 ± 0.0	8.95E+00 ± 2.68E+00
Schwefel	8.67E−12 ± 7.82E−13	2.12E+02 ± 2.92E+02

together with a group of local minima lie along the same coordinate direction. This configuration can facilitate the task of escaping from a local minimum toward the global minimum, as followed in specific approaches [40,41]. Additionally, the separability of components in some functions reduces the computational steps. A reliable algorithm should not depend on such special properties. With these arguments in mind, function rotation does lead to 'concealing' any simplifying features of this type. This is reinforced by Eq. (21) which results in strongly interconnected components of the rotated vector *Y*, by virtue of the rotation matrix *R*.

The test experiments on the rotated functions are conducted using the same values for the dimension ($D = 30$), population size ($m = 40$), maximum number of evaluations of the objective function (FEs = 200,000), shaping parameters of the beta probability distribution ($a = b = 0.1$), and number of runs (30) as those used for the unrotated functions in Section 5. The GCO algorithm parameters *c*, *cp*, *red*, and *tr* are set differently for the rotated functions as indicated in Table 5.

6.6. Results of GCO using rotated benchmark functions

The results of application of the GCO algorithm to the rotated benchmark functions $f_8(X)$ through $f_{12}(X)$ are statistically represented in Table 6 (second column) by way of the mean and standard deviation of the error of the 30 runs of the algorithm on each function. It is seen that the error for all functions is very small; a zero-value error is even obtained for $f_{10}(X)$. Such results ascertain the success of the GCO with rotated functions also. All local-minimum traps are capably avoided.

As expected, the resulting error with rotated functions is somehow greater, although quite acceptable, than that with unrotated functions. Table 7 lists error values, extracted from Tables 4 and 6, for a quantitative comparison. Note particularly that no difference in error (due to function rotation) exists for Ackley and Griewank functions, while the relatively big error occurs with the difficult-to-minimize Schwefel function, where the error amounts to 212 ± 292.

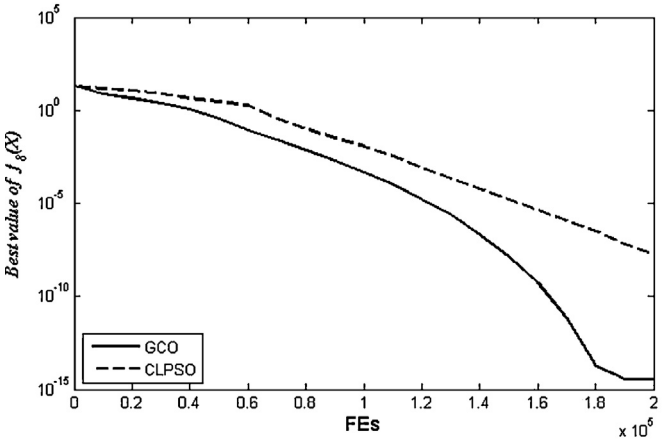

Fig. 10. Convergence for rotated Ackley function: GCO *vs.* CLPSO.

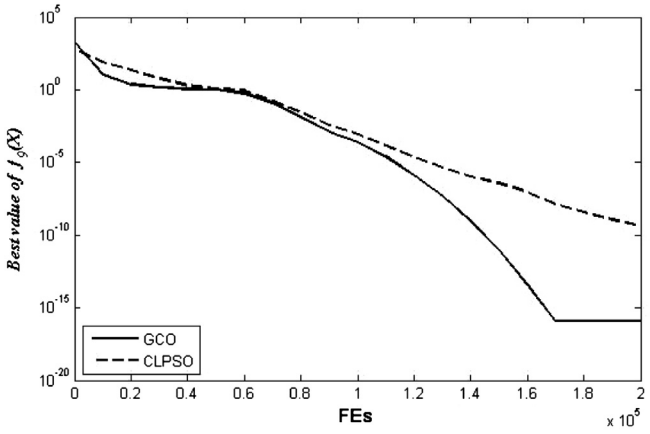

Fig. 11. Convergence for rotated Griewank function: GCO *vs.* CLPSO.

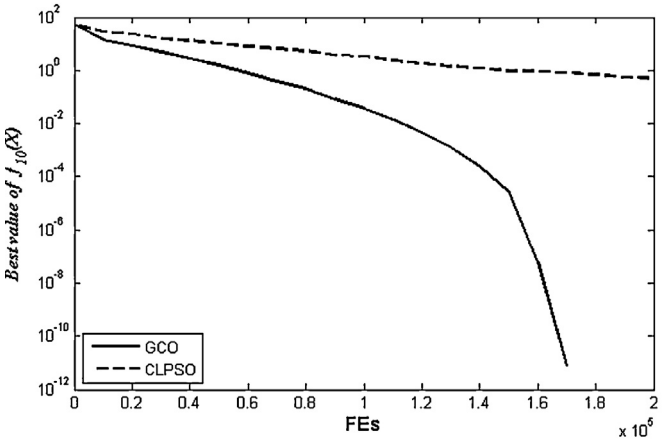

Fig. 12. Convergence for rotated Weierstrass function: GCO *vs.* CLPSO.

Figs. 10–14 illustrate the convergence characteristics of the GCO for the rotated functions $f_8(X)$ through $f_{12}(X)$, respectively. The solid curves in these figures represent the variation of the best function value with FEs in the best run. The general behavior of algorithm convergence for the rotated functions is similar to that for the unrotated functions in Figs. 3–9.

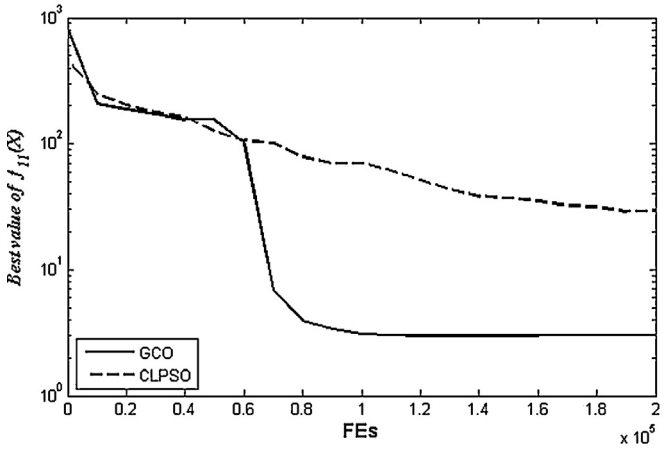

Fig. 13. Convergence for rotated Rastrigin function: GCO *vs.* CLPSO.

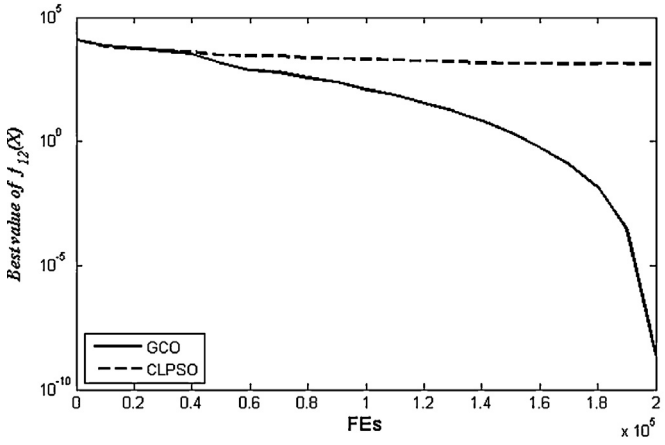

Fig. 14. Convergence for rotated Schwefel function: GCO *vs.* CLPSO.

6.7. Comparison with CLPSO using rotated benchmark functions

The GCO is again compared with the CLPSO on the basis of equal values for common particulars. Table 6 (the third column from left) gives the mean and standard deviation of the error of the 30 runs of the CLPSO on the rotated benchmark functions $f_8(X)$ through $f_{12}(X)$. A comparison between the result of the GCO and that of the CLPSO reveals that the GCO is generally better than the CLPSO for all rotated functions. For Schwefel function, in particular, the error of the GCO is 212 ± 292, as we noted in Section 6.6, whereas the error of the CLPSO is 1700 ± 186; that is, the mean value of the GCO error reduces to 12.5% of that of the CLPSO error, although the associated standard deviation is about 57% higher.

Figs. 10–14 (the dotted curves) demonstrate the convergence characteristics of the CLPSO for the rotated functions $f_8(X)$ through $f_{12}(X)$, respectively, where the variation of the best function value with FEs in the best run is represented. Like the case of unrotated functions, the GCO curves are all below the CLPSO curves, indicating faster convergence for the GCO.

7. Test experiments: composition benchmark functions

In this section, we test the algorithm using eight new proposed composition benchmark functions, $f_{13}(X)$ through $f_{20}(X)$, which are $f_{21}(X)$ through $f_{28}(X)$ in [24]; *cf.* Item 18 (standardization). Each composition function is a summation of a number (n) of basic unrotated or rotated functions with certain weights, heights, and biases. These new proposed composition functions combine the traits of the sub-functions better and retain continuous around the global/local optima. That is, we have

- Composition function 1, $f_{13}(X)$, ($n = 5$ rotated)
- Composition function 2, $f_{14}(X)$, ($n = 3$ unrotated)
- Composition function 3, $f_{15}(X)$, ($n = 3$ rotated)
- Composition function 4, $f_{16}(X)$, ($n = 3$ rotated)
- Composition function 5, $f_{17}(X)$, ($n = 3$ rotated)
- Composition function 6, $f_{18}(X)$, ($n = 5$ rotated)
- Composition function 7, $f_{19}(X)$, ($n = 5$ rotated)
- Composition function 8, $f_{20}(X)$, ($n = 5$ rotated)

The task of using composition functions in testing an optimization algorithm is extremely difficult than using unrotated and rotated ones. In other words, an algorithm may succeed with unrotated and rotated functions and fail with composition ones. Even when the algorithm succeeds with composition functions, it is well expected that the resulting error will be much greater than that with unrotated and rotated functions. Therefore, composition-function testing is of highly special significance to illustrate the robustness and superiority of the algorithm.

The definitions of the composition functions $f_{13}(X)$ through $f_{20}(X)$ and their global minimum $f(X^*)$ are shown in [24]. The search ranges for all composition functions are $[-100, 100]^D$. The test experiments on the composition functions are conducted using the values for the dimension ($D = 30$), population size ($m = 50$), maximum number of evaluations of the objective function (FEs = 300,000), shaping parameters of the beta probability distribution ($a = b = 0.1$), and number of runs (51). The GCO algorithm parameters *c*, *cp*, *red*, and *tr* are set differently for the composition functions as indicated in Table 8.

Table 8
GCO parameter settings for composition benchmark functions.

Function	*c*	*cp*	*red*	*tr*
$f_{13}(X)$	2	0.01	0.25	4
$f_{14}(X)$	2	0.03	0.25	10
$f_{15}(X)$	2	0.95	0.25	1
$f_{16}(X)$	2	0.03	0.25	5
$f_{17}(X)$	2	0.03	0.25	5
$f_{18}(X)$	2	0.03	0.25	5
$f_{19}(X)$	2	0.01	0.25	5
$f_{20}(X)$	2	0.01	0.25	3

Table 9
Mean and standard deviation of error for composition benchmark functions: GCO *vs.* NBIPOP-aCMA-ES.

Function	GCO	NBIPOP-aCMA-ES
$f_{13}(X)$	145.1942 ± 50.20826	192.157 ± 27.152
$f_{14}(X)$	68.4122 ± 33.56426	838.392 ± 459.988
$f_{15}(X)$	2579.508 ± 495.9005	667.086 ± 289.554
$f_{16}(X)$	234.3084 ± 23.73674	161.757 ± 30.045
$f_{17}(X)$	275.031 ± 3.720117	219.984 ± 11.094
$f_{18}(X)$	199.7816 ± 2.561173	158.223 ± 29.999
$f_{19}(X)$	449.08 ± 136.3417	468.925 ± 73.770
$f_{20}(X)$	154.9125 ± 90.13492	268.627 ± 73.458

7.1. Results of GCO using composition benchmark functions

The results of application of the GCO algorithm to the composition benchmark functions $f_{13}(X)$ through $f_{20}(X)$ are statistically represented in Table 9 (second column) by way of the mean and standard deviation of the error of the 51 runs of the algorithm on each function.

As expected, the resulting error with composition functions is somehow greater, although quite acceptable, than that with

unrotated and rotated functions because these composition functions are more difficult-to-minimize.

7.2. Comparison with NBIPOP-aCMA-ES using composition benchmark functions

The GCO is compared with a more advanced optimizer BI-population covariance matrix adaptation evolution strategy with alternative restart strategy (NBIPOP-$_a$CMA-ES) [22,23], which is the winner of the competition on real-parameter single objective optimization at IEEE CEC-2013[24], on the basis of equal values for common particulars. Table 9 (the third column from left) gives the mean and standard deviation of the error of the 51 runs of the NBIPOP-$_a$CMA-ES on the composition benchmark functions $f_{13}(X)$ through $f_{20}(X)$. A comparison between the result of the GCO and that of the NBIPOP-$_a$CMA-ES reveals that the GCO is generally better than the NBIPOP-$_a$CMA-ES for functions $f_{13}(X)$, $f_{14}(X)$, $f_{19}(X)$, and $f_{20}(X)$. For function $f_{14}(X)$, in particular, the error of the GCO is 68.4122 ± 33.56426, as we noted in Section 7.1, whereas the error of the NBIPOP-$_a$CMA-ES is 838.392 ± 459.988; that is, the mean value of the GCO error reduces to 8% of that of the NBIPOP-$_a$CMA-ES error, and the associated standard deviation reduces to 7%. Also for function $f_{20}(X)$, the error of the GCO is 154.9125 ± 90.13492 whereas the error of the NBIPOP-$_a$CMA-ES is 268.627 ± 73.458; that is, the mean value of the GCO error reduces to 58% of that of the NBIPOP-$_a$CMA-ES error, and the associated standard deviation is about 23% higher. For function $f_{16}(X)$ through $f_{18}(X)$, the error differences between GCO and NBIPOP-$_a$CMA-ES are relatively small. For functions $f_{15}(X)$, on the other hand, the error of the GCO is above that of the NBIPOP-$_a$CMA-ES. It is not expected a better performance of the infant algorithm on all classes of problems. An improved performance on one class leads to performance degradation on another class. This is the no free lunch theorem [37].

8. Application to spacecraft trajectory design

8.1. Optimization in spacecraft trajectory design

In recent years, aerospace researchers have realized the benefits of approaching trajectory design problems from a global optimization point of view. Many interplanetary trajectory design problems can be formalized as optimization problems [42,43]. One of the main goals to be achieved is the *maximization* of the spacecraft mass available for the pay-load or, equivalently, the *minimization* of the propellant necessary on board. The multiple gravity assist (MGA) problem has been intensively investigated. For example, Hartmann et al. [44] as well as Abdelkhalik and Mortari [45] use genetic algorithms to describe optimum trajectories. Izzo et al. [46] introduce a deterministic search space pruning algorithm which is then applied in conjunction with some heuristic global solvers, resulting in a very efficient algorithm. In the assessment and comparison of different optimization approaches, it is crucial to specify the exact problem, the underlying mathematical model, variable constraints, and all details of the solution scheme. The annual global trajectory optimization competition (GTOC), launched by the ESA ACT (European Space Agency, Advanced Concepts Team) in 2005, plays a stimulating role in this respect. Information about past editions can be found on the website http://www.esa.int/gsp/ACT/mad/op/GTOC/index.htm.

We particularly refer to the work of Vinkó and Izzo [43], where eight trajectory optimization methods, including particle swarm optimizer, genetic algorithm, simulated annealing, differential evolution, and COOP-1,2,3,4, are assessed and compared together using MGA models for a number of missions. Each problem is formulated with the aid of a black-box objective function accepting, as input, the adaptable solution vector (state vector of the model) and returning the objective function and constraint evaluation. The best known solutions (the optimizing vectors and the corresponding objective function values) are reported as well. These are obtained using DiGMO (distributed global multi-objective optimizer), a complicated cooperative optimization method which combines several population-based optimizers.

8.2. MGA missions and models

The gravity assist maneuver is the utilization of the gravity field of a celestial body and the relative motion of a spacecraft with respect to the body in order to change the spacecraft velocity [42]. During an interplanetary mission, the spacecraft encounters the massive celestial body with a specific relative velocity. Due to this close passage, there will be an exchange of momentum between the spacecraft and the body with the result that the spacecraft increases or reduces the magnitude of its inertial velocity vector or rotates the direction of this vector. The body loses an extremely small proportion of its orbital momentum owing to its enormous mass, compared to the mass of the spacecraft. However, the change in spacecraft velocity can be considerably large. Such an interesting 'interaction' between a spacecraft and a celestial body seems like that of a bike rider holding a running bus.

Multiple gravity assist (MGA) maneuvers are now extensively planned for multi-path missions among a group of planets, each planet having a certain gravity-field impact on the velocity change, denoted collectively by ΔV, of the spacecraft. If no gravity assists were made use of throughout the mission, the velocity changes needed in the various stages for the spacecraft to travel from the Earth orbit to the target orbit would be furnished by the propulsion system alone, which would be a highly sophisticated system requiring exceedingly huge budgets. MGA missions have been sent forth in space over the past forty years to reach targets having orbits of very high or very low energy levels, in comparison with Earth. They are even intended to appreciably alter the heliocentric orbit inclination. These types of orbits are all categorized as high-ΔV targets because of the high amount of ΔV necessary to reach them from Earth.

We will be concerned here with MGA models, those used for MGA missions. Before discussing the main guiding ideas of such models, the following definitions are in order [42]:

8.2.1. Leg

A leg is the trajectory followed by the spacecraft between two celestial bodies.

8.2.2. Pericenter radius

The pericenter radius, r_p, at a celestial body is the *minimum* distance between the spacecraft trajectory and the body.

8.2.3. Orbital eccentricity

The orbital eccentricity, e, of a celestial body represents the amount by which its orbit deviates from a perfect circle, where

$e = 0$ for circular orbits.
$0 < e < 1$ for elliptic orbits.
$e = 1$ for parabolic trajectories.
$e > 1$ for hyperbolic trajectories.

In a trajectory design problem, we have a sequence of $n + 1$ celestial bodies $B_0, B_1, \ldots, B_n$ (B_0 is Earth). The bodies are not necessarily distinct. It is required to visit the body sequence in such a way that the overall mass consumption of the spacecraft is minimized. The sequence is here assumed to be *fixed*, but we may also think of models where this sequence is part of the decision problem.

An MGA model has the variables:

- t_0, the starting date of the mission.
- T_i, $i = 1, 2, \ldots, n$, the time of flight along leg i, joining body B_{i-1} with body B_i.

Given the values of these variables, we are able to identify the positions p_{i-1} and p_i, respectively, of body B_{i-1} at time

$$t_0 + \sum_{j=1}^{i-1} T_j$$

and of body B_i at time

$$t_0 + \sum_{j=1}^{i} T_j$$

It is then possible to calculate the velocities at the end of each leg i and at the beginning of the next leg $i+1$, $i = 1, 2, \ldots, n-1$. In order to transfer from one leg to the next one, the spacecraft needs to provide a single impulse, a single change in velocity ΔV_i. In the initial leg, the spacecraft provides a single impulse ΔV_0 to leave the Earth orbit and attain the starting velocity at the initial leg. Similarly, in the last leg, the spacecraft provides a single impulse ΔV_n to transfer from the final velocity in the last leg to the velocity of the target body B_n.

Each impulse causes a mass consumption proportional to the modulus (absolute value of the magnitude) of the change in velocity, $|\Delta V_i|$, $i = 0, 1, \ldots, n$. In order to minimize the overall mass consumption, the following objective function is taken as a performance measure:

$$f(X) = \sum_{i=0}^{n} |\Delta V_i| \tag{30}$$

where X is the state vector. The quantities constituting the right-hand side of (30) depend implicitly on the components (state variables) of X. The optimization process consists in finding a vector $X = X^*$ which *minimizes* function (30).

MGA models usually include constraints on the pericenter radius, r_{pi}, at each intermediate body B_i, $i = 1, 2, \ldots, n-1$, which typically require r_{pi} not to fall below a certain lower threshold. The reason is that the spacecraft must be far enough from body B_i in order to be able to leave the planet orbit and avoid 'forced' landing due to the gravitational force.

Objective functions, other than (30), do exist. Of particular interest is the problem of 'asteroid deflection', suggested in the GTOC competition to avoid possible dangerous effects of an asteroid on Earth [43]. The aim of the spacecraft mission in this case is to deliver the greatest permissible push so that the asteroid is compelled to change the shape of its orbit to the farthest extent. The objective function takes the form

$$f(X) = m_f u_{\text{rel}} v_{\text{ast}} \tag{31}$$

where m_f is the final mass of the spacecraft, u_{rel} is the spacecraft velocity relative to the asteroid at arrival, and v_{ast} is the heliocentric velocity of the asteroid. Again X is the state vector, with components implicit in the quantities of the right-hand side of (31). In the optimization process, a vector $X = X^*$ is to be found such that function (31) is *maximized*.

Objective functions, like (30) and (31), are often considered as *black boxes*, accepting vectors X as inputs and returning the objective function values as outputs.

8.3. Missions used for GCO application

Our group counseling optimizer (GCO) will be applied to trajectory design problems of two spacecraft missions: Cassini1 and GTOC1. The models of both missions are of the MGA type. The solution of the Cassini1 problem requires a *minimization* process, while that of the GTOC1 problem requires a *maximization* process.

8.3.1. Cassini1

This is an MGA problem in the framework of the Cassini spacecraft trajectory design [43]. The aim of the mission is to reach Saturn and be captured by its gravity field into an orbit having a pericenter radius

$$r_p = 108,950\,\text{km} \tag{32}$$

and an eccentricity

$$e = 0.98\,(\text{elliptic orbit}) \tag{33}$$

The planetary fly-by sequence is made up of six celestial bodies

$$B_0, B_1, B_2, B_3, B_4, B_5 \quad (n = 5)$$

which are, respectively, Earth, Venus, Venus, Earth, Jupiter, Saturn.

That is, the initial body is Earth (as usual) and the target body is Saturn (as planned), and the mission has five legs, leg i, $i = 1, 2, \ldots, 5$, joining body B_{i-1} with body B_i. Note that leg 2 joins Venus with itself (the spacecraft returns to the same body) and leg 3 joins Venus with Earth (the spacecraft visits Earth once after launching during the mission). The real Cassini mission was launched from Earth on October 15, 1997 and arrived at Saturn on July 1, 2004.

The model of the Cassini1 mission is a simplified model of the MGA type. The objective function $f(X)$ is of the form (30), where the total change ΔV accumulated during the mission is calculated, including the launch ΔV and subsequent ΔV's needed at the planets and upon eventual arrival at the target to accomplish the intended orbit injection. The objective function in this situation is to be minimized in an optimization process.

The state vector, X, of the model has six components (state variables) x_i, $i = 1, 2, \ldots, 6$, which are the *times*, in days, of the various stages of the mission. These are denoted by $t_0, T_1, \ldots, T_5$, and we can write

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} t_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \end{bmatrix} \tag{34}$$

Specifically, t_0 is the starting date of the mission, expressed in the astronomical unit MJD2000 (Modified Julian Date: time in days measured from January 1, 2000), and T_i, $i = 1, 2, \ldots, 5$, is the flight time along leg i. The search ranges of these variables are given in Table 10.

The four pericenter radii $r_{p1}, \ldots, r_{p4}$ at the intermediate bodies $B_1, \ldots, B_4$, respectively, are constrained not to be less than the lower thresholds indicated in Table 11.

Table 10
Search ranges of state variables for Cassini1.

State variable	Search range	Units
$x_1 = t_0$	[−1000, 0]	MJD2000
$x_2 = T_1$	[30, 400]	Days
$x_3 = T_2$	[100, 470]	Days
$x_4 = T_3$	[30, 400]	Days
$x_5 = T_4$	[400, 2000]	Days
$x_6 = T_5$	[1000, 6000]	Days

Table 11
Pericenter radius thresholds for Cassini1.

Pericenter radius	Lower threshold (km)
r_{p1}	6351.8
r_{p2}	6351.8
r_{p3}	6778.1
r_{p4}	600,000

The best solution (X) known for the Cassini1 problem is [43]

$$X_C^{bn} = \begin{bmatrix} -789.8 \\ 158.3 \\ 449.4 \\ 54.7 \\ 1024.7 \\ 4552.8 \end{bmatrix} \tag{35}$$

where the superscript *bn* stands for 'best known' and the subscript C stands for Cassini1.

The corresponding value of the objective function (the least known value) is

$$f(X_C^{bn}) = 4.9307 \text{ km/s} \tag{36}$$

8.3.2. GTOC1

This is again an MGA problem, drawing inspiration from the first edition of the GTOC competition [43,47]. The problem is concerned with asteroid deflection, suggested to avoid expected danger of an asteroid to Earth. Asteroid TW229 is taken as the target of the GTOC1 spacecraft 'intercept' mission. Attempts to obtain generic solutions to the asteroid-deflection problems are important precautionary measures: should astronomers in the future discover a potentially dangerous asteroid, a timely and efficacious decision can be confidently taken.

The mission in this case is required to deliver the greatest possible push so that the target asteroid is compelled to largely change the shape of its original orbit, thereby preventing possible catastrophic events.

The GTOC1 planetary fly-by sequence is rather long, containing eight celestial bodies B_0, B_1, ..., B_7 which are, respectively, Earth, Venus, Earth, Venus, Earth, Jupiter, Saturn, Asteroid TW229 ($n = 7$), and there are seven legs. Note that Earth appears three times (B_0, B_2, B_4) in the sequence and Venus appears twice (B_1, B_3).

In the MGA simplified model of the GTOC1, the objective function $f(X)$, of the form (31), is to be maximized in an optimization process. This implies the maximization of the change in semi-major axis of the asteroid orbit following an elastic impact of the spacecraft with the asteroid.

Table 12
Search ranges of state variables for GTOC1.

State variable	Search range	Units
$x_1 = t_0$	[3000, 10,000]	MJD2000
$x_2 = T_1$	[14, 2000]	Days
$x_3 = T_2$	[14, 2000]	Days
$x_4 = T_3$	[14, 2000]	Days
$x_5 = T_4$	[14, 2000]	Days
$x_6 = T_5$	[100, 9000]	Days
$x_7 = T_6$	[366, 9000]	Days
$x_8 = T_7$	[300, 9000]	Days

Table 13
Pericenter radius thresholds for GTOC1.

Pericenter radius	Lower threshold (km)
r_{p1}	6351.8
r_{p2}	6778.1
r_{p3}	6351.8
r_{p4}	6778.1
r_{p5}	600,000
r_{p6}	70,000

The state vector, X, has eight state variables x_i, $i = 1, 2, \ldots, 8$, which are the times t_0, T_1, T_2, ..., T_7 of the various stages of the mission. That is,

$$X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} t_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \end{bmatrix} \tag{37}$$

The search ranges of the state variables are given in Table 12.

The six pericenter radii $r_{p1}, \ldots, r_{p6}$ at the intermediate bodies B_1, ..., B_6, respectively, are constrained not to be less than the lower thresholds indicated in Table 13.

The solution is also based on a launcher ΔV of 2.5 km/s, a specific impulse I_{sp} of duration 2500 s, and a spacecraft initial mass m_0 of 1500 kg.

The best solution known for the GTOC1 problem is [43]

$$X_G^{bn} = \begin{bmatrix} 6809.5 \\ 169.6 \\ 1079.4 \\ 56.5 \\ 1044.0 \\ 3824.2 \\ 1042.9 \\ 3393.1 \end{bmatrix} \tag{38}$$

where the subscript G stands for GTOC1.

The corresponding value of the objective function (the highest known value) is

$$f(X_G^{bn}) = 1,580,599 \text{ kg km}^2/\text{s}^2 \tag{39}$$

8.4. Application to Cassini1 trajectory design

In this section, we apply the GCO algorithm to the Cassini1 trajectory design problem. Appropriate values for the six

components of the model's state vector X of Eq. (34) are to be obtained so that the objective function $f(X)$ of Eq. (30) is minimized. Throughout the steps of the minimization process, the objective function is regarded as a black box, and its value is calculated on an input–output mapping basis. A program code in C++ is downloaded from the ESA ACT website and is used for objective function evaluation.

The resulting solution is characterized by:

- X_C^{GCO}: minimizing state vector (X^*).
- $f(X_C^{GCO})$: best (least) found value of the objective function.
- $f_{C,av}^{GCO}$: average value of the objective function on a certain number of independent runs.

The solution is assessed in light of the previously published solutions using the eight optimizers in [43] and the best known solution given by Eqs. (35) and (36). Furthermore, a separate comparison is made with a solution provided by the CLPSO, the enhanced version of the PSO. The convergence characteristics of both GCO and CLPSO, when applied to the Cassini1 problem, are studied in terms of the variation of the best found value of the objective function with the number of function evaluations, FEs.

8.4.1. Results of GCO application to Cassini1

For GCO parameters, we have the particulars:

- Dimension, $D = 6$ (This is the given dimension of the state vector).
- Population size, $m = 20$.
- Maximum number of evaluations of the objective function, FEs = 200,000, implying that the maximum number of iterations, $itr\_max = 10{,}000$ (stopping criterion).
- Shaping parameters of the beta probability distribution, $a = b = 0.1$.

The four parameters defined specifically for the GCO are:

- Number of group members acting as counselors, $c = 1$.
- Counseling probability, $cp = 0.1$.
- Search range reduction coefficient, $red = 0.25$.
- Transition rate from exploration to exploitation, $tr = 2$.

The experiments are run 20 times. This number of runs and the number 200,000 of FEs are the same as those in solutions in [43] so that a comparison can be made on a common basis.

The GCO code and the Cassini1 code for objective function evaluation are employed together (the former calls the latter) to solve the minimization problem. It is to be noted that the astronomical relationships of temporal and physical quantities constituting the various ΔV's of the black-boxed objective function of Eq. (30) are outside the realm of an optimization approach, although they are incorporated in the ready-made code of objective function evaluation. We have mentioned in passing few such quantities, including the pericenter radius (Eq. (32)) and eccentricity (Eq. (33)) of the orbit of Saturn, the target body of the mission; the lower thresholds of the pericenter radii of the four intermediate bodies (Table 11). Remember that the components of the state vector to be determined are the starting date t_0 of the mission and the respective flight times $T_1, \ldots, T_5$ along the five legs. The search ranges of these components are different from one another as Table 10 indicates.

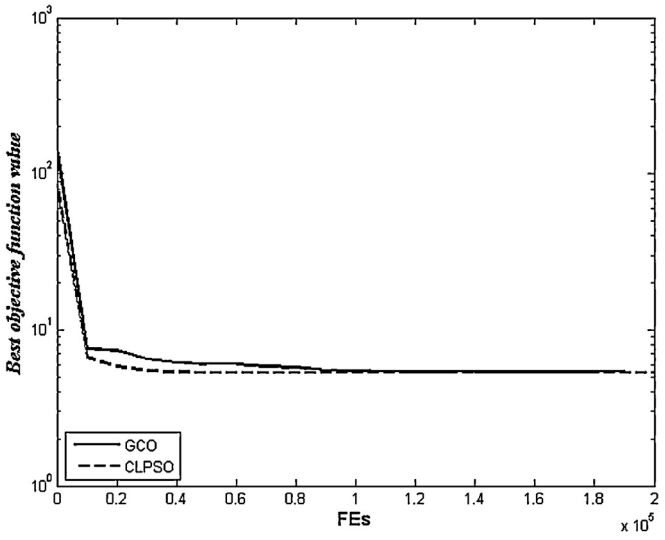

Fig. 15. Convergence for Cassini1: GCO *vs.* CLPSO, FEs = 200,000.

Table 14
Comparison of objective function values of GCO with those of other optimizers in [43] for Cassini1.

Optimizer	Objective function	
	Best found value (200,000 FEs)	Average value (20 runs)
GCO	4.9466	5.2983
PSO	5.3946	9.0650
Genetic algorithm	5.3288	10.3103
Simulated annealing	5.4162	8.4618
Differential evolution	5.3034	8.7114
COOP-1	4.9307	10.9116
COOP-2	4.9500	9.4133
COOP-3	5.3034	11.1248
COOP-4	5.0871	9.3050

The resulting state vector, at the end of iterations, turns out to be

$$X_C^{GCO} = \begin{bmatrix} -789.7 \\ 158.6 \\ 449.4 \\ 54.5 \\ 1027.2 \\ 4564.7 \end{bmatrix} \quad (FEs = 200,000) \tag{40}$$

The best found value of the objective function is

$$f(X_C^{GCO}) = 4.9466\ \text{km/s} \quad (FEs = 200,000) \tag{41}$$

On the 20 runs, the average value of the objective function is

$$f_{C,av}^{GCO} = 5.2983\ \text{km/s} \quad (FEs = 200,000) \tag{42}$$

The convergence characteristics of the GCO for the Cassini1 problem, with FEs = 200,000, are demonstrated in Fig. 15. The solid curve in the figure represents the variation of the best found objective function value with the number of FEs, in the best run. It is clear that the objective function takes on successive fitness values in a smooth, monotonically decreasing, acceptably fast manner.

In Table 14, we compare the results (41) and (42) for the Cassini1 problem with the corresponding results of the eight optimizers in [43]. It is evident from this table that, among the nine optimizers considered, the GCO is the *next-best* regarding the best found value of the objective function of the Cassini1 problem, and is truly the *best* regarding the average value of this function.

Table 15
Comparison of GCO with CLPSO objective function values for Cassini1.

Optimizer	Objective function	
	Best found value	Average value
GCO (FEs = 200,000)	4.9466	5.2983
GCO (FEs = 400,000)	4.9383	5.5510
CLPSO (FEs = 200,000)	4.9654	6.0900
CLPSO (FEs = 400,000)	4.9444	5.9264

The COOP-1 optimizer is the only one that could produce the best known value (4.9307, Eq. (36)) of the objective function. The GCO (the next-best) produces a best found value (4.9466) which is above the best known value by only 0.32%, a very slight difference.

The average value of the objective function (5.2983) obtained by the GCO (the best) is significantly lower (better) than those of all other eight optimizers. It reduces to 62.6% of that obtained by the simulated annealing optimizer (the next-best).

Having assessed and compared the GCO among a group of optimizers, we further attempt to employ it for a hopefully better solution for the Cassini1 problem. To this end, the maximum number of function evaluations is increased to FEs = 400,000 through increasing the maximum number of iterations to *itr_max* = 20,000. The other particulars and parameters of the algorithm remain unchanged. The resulting solution, under these conditions, takes the form

$$X_C^{GCO} = \begin{bmatrix} -790.8 \\ 159.9 \\ 449.4 \\ 54.4 \\ 1026.8 \\ 4557.3 \end{bmatrix} \quad (FEs = 400,000) \tag{43}$$

$$f(X_C^{GCO}) = 4.9383 \quad (FEs = 400,000) \tag{44}$$

$$f_{C,av}^{GCO} = 5.5510 \quad (FEs = 400,000) \tag{45}$$

Comparing (44) and (45) with (41) and (42), respectively, we realize that increasing FEs from 200,000 to 400,000 leads to a limited reduction (improvement) of the best found value of the objective function, from 4.9466 to 4.9383. This is only 0.17% of the best known value. However, the average value increases (worsens) from 5.2983 to 5.5510, which is 5.13% of the best known value. See Table 15, the upper two rows. The convergence characteristics of the GCO for the Cassini1 problem, with FEs = 400,000, are shown in Fig. 16.

8.4.2. Comparison with CLPSO application to Cassini1

The CLPSO [21] is applied to the Cassini1 problem for the sake of comparison. We consider two cases: the first of FEs = 200,000 and the second of FEs = 400,000.

When FEs = 200,000, the maximum number of iterations is 10,000 (population size $m = 20$). The solution in this case is

$$X_C^{CLPSO} = \begin{bmatrix} -785.4 \\ 153.6 \\ 449.4 \\ 54.9 \\ 1020.0 \\ 4540.6 \end{bmatrix} \quad (FEs = 200,000) \tag{46}$$

$$f(X_C^{CLPSO}) = 4.9654 \quad (FEs = 200,000) \tag{47}$$

$$f_{C,av}^{CLPSO} = 6.0900 \quad (FEs = 200,000) \tag{48}$$

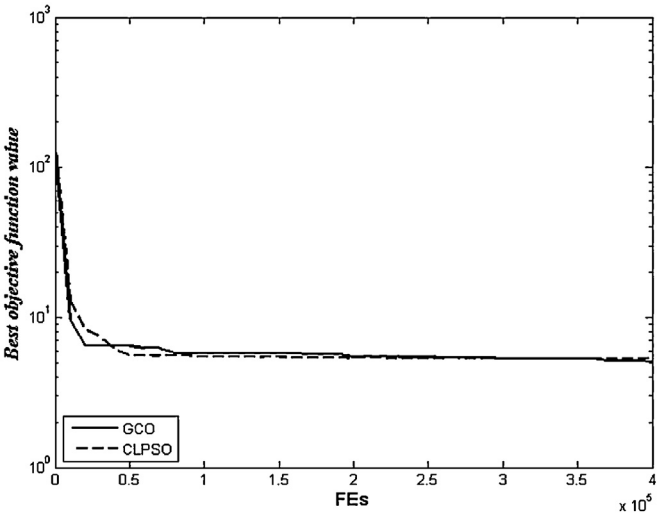

Fig. 16. Convergence for Cassini1: GCO *vs.* CLPSO, FEs = 400,000.

When FEs = 400,000, the maximum number of iterations is 20,000 ($m = 20$). The solution then is

$$X_C^{CLPSO} = \begin{bmatrix} -788.3 \\ 158.4 \\ 449.4 \\ 53.6 \\ 1033.7 \\ 4567.3 \end{bmatrix} \quad (FEs = 400,000) \tag{49}$$

$$f(X_C^{CLPSO}) = 4.9444 \quad (FEs = 400,000) \tag{50}$$

$$f_{C,av}^{CLPSO} = 5.9264 \quad (FEs = 400,000) \tag{51}$$

See Table 15, the lower two rows. It is evident from Table 15 that the objective function values obtained by the CLPSO for the Cassini1 problem are all inferior (though slightly) to those obtained by the GCO. The best found value is 4.9383 for the GCO with FEs = 400,000, and the best average value is 5.2983 for the GCO also, but with FEs = 200,000. Note, in addition, that the results of the CLPSO, like the GCO, improve when the FEs increase from 200,000 to 400,000. The convergence characteristics of the CLPSO for the Cassini1 problem are demonstrated by the dotted curves in Figs. 15 and 16 for FEs = 200,000 and FEs = 400,000, respectively; the comparison with the GCO is obvious.

8.5. Application to GTOC1 trajectory design

Here we apply the GCO algorithm to the GTOC1 trajectory design problem. The eight components of the model's state vector X of Eq. (37) are to be obtained for maximizing the objective function $f(X)$ of Eq. (31). The objective function is again regarded as a black box with available input and output ports only. A program code in C++ is also downloaded from the ESA ACT website, for objective function evaluation.

The solution of the problem, accommodating the maximizing state vector, the best (highest) found value of the objective function, and the average value of the objective function on a certain number of independent runs, will be designated as

$$X_G^{GCO}, f(X_G^{GCO}), f_{G,av}^{GCO}$$

8.5.1. Results of GCO application to GTOC1

The best parameters values of GCO are as follows:

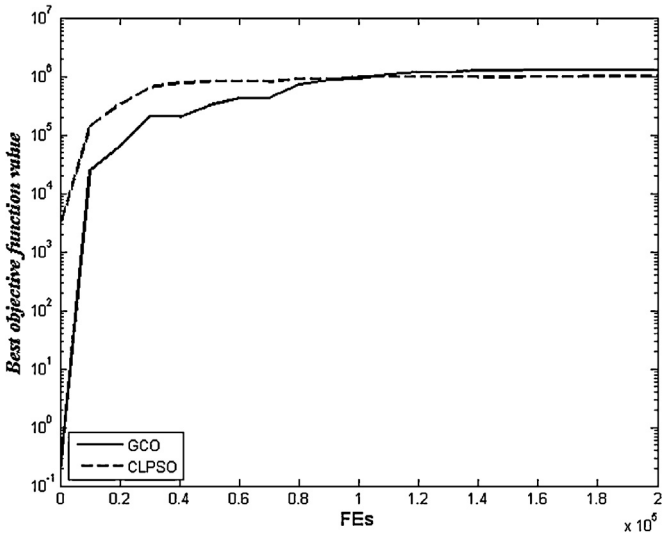

Fig. 17. Convergence for GTOC1: GCO *vs.* CLPSO, FEs = 200,000.

- $D = 8$ (The given dimension of the state vector).
- $m = 20$.
- FEs = 200,000 (itr_max = 10,000).
- $a = b = 0.1$.

and the GCO parameters:

- $c = 1$.
- $cp = 0.4$.
- $red = 0.75$.
- $tr = 5$.

The experiments are run 20 times. The GCO code and the GTOC1 code for objective function evaluation are employed together for solving the maximization problem. We emphasize that the astronomical quantities and relationships constituting the expression of m_f, u_{rel}, and v_{ast} of the objective function (31) are irrelevant from an optimization viewpoint. However, we casually referred to the lower thresholds of the pericenter radii of the six intermediate bodies of the mission (Table 13) and the values of the launcher ΔV, specific impulse I_{sp}, and spacecraft initial mass m_0. These quantities, among others, are incorporated in the encapsulated code of objective function evaluation. The components of the state vector to be determined are the starting date t_0 of the mission and the respective flight times $T_1, \ldots, T_7$ along the seven legs. The search ranges of these components are generally different from one another (Table 12).

The resulting solution is

$$X_G^{GCO} = \begin{bmatrix} 7385.7 \\ 175.9 \\ 1083.6 \\ 64.7 \\ 1416.6 \\ 526.6 \\ 6260.7 \\ 8144.9 \end{bmatrix} \quad (FEs = 200,000) \tag{52}$$

$$f(X_G^{GCO}) = 1,480,000 \quad (FEs = 200,000) \tag{53}$$

$$f_{G,av}^{GCO} = 1,131,267 \quad (FEs = 200,000) \tag{54}$$

Fig. 17 demonstrates the convergence characteristics of the GCO for the GTOC1 problem, with FEs = 200,000. The solid curve in the figure shows the variation of the best found objective function value with the number of FEs, in the best run. The objective function is seen to take on successive fitness values in a smooth, monotonically increasing, acceptably fast manner.

Table 16
Comparison of objective function values of GCO with those of other optimizers in [43] for GTOC1.

Optimizer	Objective function	
	Best found value (200,000 FEs)	Average value (20 runs)
GCO	1,480,000	1,131,267
PSO	1,003,905	694,921
Genetic algorithm	1,018,085	450,198
Simulated annealing	1,110,654	760,675
Differential evolution	1,262,748	522,346
COOP-1	1,406,033	730,602
COOP-2	1,165,523	501,770
COOP-3	1,149,909	703,132
COOP-4	1,365,642	756,360

In Table 16, the results (53) and (54) of the GCO for the GTOC1 problem are compared with the corresponding results of the eight optimizers considered in [43]. This table tells us that no optimizer, among the nine ones, is able to generate the best known value (1,580,599, Eq. (39)) of the objective function of the GTOC1 problem. However, the GCO yields comparatively excellent results. It is the *best* with respect to the best found value and also the *best* with respect to the average value.

The best found value (1,480,000) of the GCO amounts to 93.6% of the best known value. It is higher (better) than the best found value of the COOP-1 optimizer (the next-best) by 73,967 (4.7% of the best known value).

The average value (1,131,267) of the GCO is higher than that of the simulated annealing optimizer (the next-best) by 370,592 (23.4% of the best known value).

In the hope of obtaining a better solution for the GTOC1 problem, we increase the maximum number of function evaluations to FEs = 400,000 by increasing the maximum number of iterations to itr_max = 20,000. The remaining particulars and parameters of the algorithm are kept unchanged. The resulting solution then takes the form

$$X_G^{GCO} = \begin{bmatrix} 5648.5 \\ 176.2 \\ 1081.8 \\ 69.7 \\ 1043.3 \\ 4534.7 \\ 1185.9 \\ 3685.1 \end{bmatrix} \quad (FEs = 400,000) \tag{55}$$

$$f(X_G^{GCO}) = 1,497,706 \quad (FEs = 400,000) \tag{56}$$

$$f_{G,av}^{GCO} = 1,206,410 \quad (FEs = 400,000) \tag{57}$$

Comparing (56) and (57) with (53) and (54), respectively, reveals that increasing FEs from 200,000 to 400,000 leads to an improvement of the best found value of the objective function from 1,480,000 to 1,497,706, which is 1.12% of the best known value. There is likewise an improvement in the average value from 1,131,267 to 1,206,410 (4.75% of the best known value). See Table 17, the upper two rows. The convergence characteristics of the GCO for the GTOC1 problem, with FEs = 400,000, are shown by the solid curve in Fig. 18.

Table 17
Comparison of GCO with CLPSO objective function values for GTOC1.

Optimizer	Objective function	
	Best found value	Average value
GCO (FEs = 200,000)	1,480,000	1,131,267
GCO (FEs = 400,000)	1,497,706	1,206,410
CLPSO (FEs = 200,000)	1,262,100	970,820
CLPSO (FEs = 400,000)	1,496,200	1,124,100

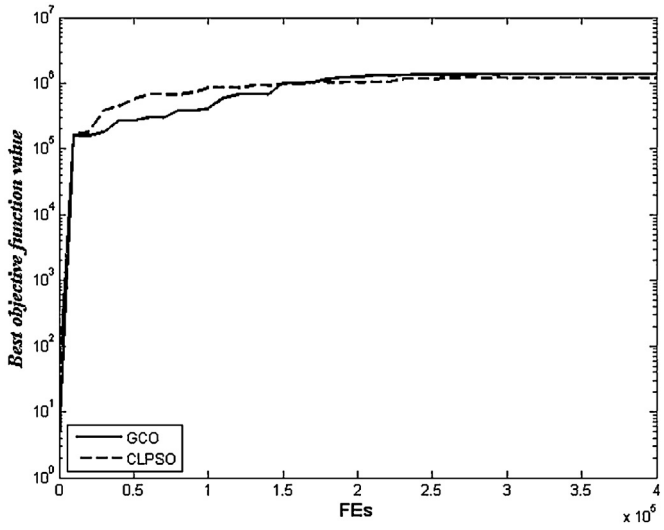

Fig. 18. Convergence for GTOC1: GCO *vs.* CLPSO, FEs = 400,000.

8.5.2. Comparison with CLPSO application to GTOC1

As we did with the Cassini1 problem, the CLPSO is here applied to the GTOC1 problem in the two cases: FEs = 200,000 ($m = 20$) and FEs = 400,000 ($m = 20$). In the first case, the solution is

$$X_G^{CLPSO} = \begin{bmatrix} 7409.3 \\ 161.8 \\ 1055.9 \\ 40.9 \\ 1062.4 \\ 597.1 \\ 6810.8 \\ 7920.0 \end{bmatrix} \quad (FEs = 200,000) \tag{58}$$

$$f(X_G^{CLPSO}) = 1,262,100 \quad (FEs = 200,000) \tag{59}$$

$$f_{G,av}^{CLPSO} = 970,820 \quad (FEs = 200,000) \tag{60}$$

In the second case, the solution is

$$X_G^{CLPSO} = \begin{bmatrix} 5640.1 \\ 189.8 \\ 1076.3 \\ 68.2 \\ 1041.5 \\ 4501.2 \\ 1201.3 \\ 2195.6 \end{bmatrix} \quad (FEs = 400,000) \tag{61}$$

$$f(X_G^{CLPSO}) = 1,496,200 \quad (FEs = 400,000) \tag{62}$$

$$f_{G,av}^{CLPSO} = 1,124,100 \quad (FEs = 400,000) \tag{63}$$

See the lower two rows of Table 17. This table indicates that the objective function values obtained by the CLPSO for the GTOC1 problem are somewhat inferior to those obtained by the GCO. The best found value is 1,497,706 (94.8% of the best known value) for the GCO with FEs = 400,000, and the best average value is 1,206,410 (76.3% of the best known value) for FEs = 400,000 also. The convergence characteristics of the CLPSO for the GTOC1 problem are shown by the dotted curves in Figs. 17 and 18 for FEs = 200,000 and FEs = 400,000, respectively; the comparison with the GCO is clear.

Table 18
Rankings of GCO among nine optimizers.

Comparison basis	Cassini1	GTOC1
Best found value of objective function	Next-best	Best
Average value of objective function	Best	Best

The rankings of the GCO among the nine optimizers, for both Cassini1 and GTOC1 missions, are summarized in Table 18.

9. Conclusion

A novel population-based, gradient-free, optimization approach for single-objective, unconstrained, continuous problems is developed. Instead of mimicking the behavior of living biological organisms, our approach adopts as *metaphor* the human social behavior in solving life problems through counseling within a group. Therefore, we call the optimization algorithm a group counseling optimizer (GCO). The inspiration radiates from the intriguing twenty points of analogy between group counseling and population-based optimization. Eighteen metaphoric items (1–12 and 15–20) of this table are invoked in devising the GCO computational scheme.

Item 13 (problem complication) for *multi-objective* optimization is utilized in [26]. The remaining item can still be utilized in other versions of the algorithm: Item 14 (inferior solution) for *suboptimum* policies.

The GCO algorithmic iterations are visualized as counseling sessions, with counselees and counselors. There are four parameters to be: *c*, *cp*, *red*, and *tr*. Solutions are progressively improved through a certain number of iterations by means of one of two strategies: *other-members counseling* and *self-counseling*. Counseling has no subtlety or sophistication and, consequently, the counseling-based GCO is free of any complicated mathematical operations.

The proposed approach is tested in minimization problems using seven unrotated benchmark functions. The resulting error of the algorithm on each function is statistically represented in terms of the *mean* and *standard deviation*. The *convergence* characteristics of the algorithm are studied *via* the variation of the best function value with the number of function evaluations. These tests are repeated for five rotated benchmark functions. Besides, GCO is tested to minimize eight highly complicated composition functions. The results for all functions, unrotated, rotated, and composition are found to be highly acceptable and promising, demonstrating the accuracy and robustness of the algorithm. Of particular interest is the successful convergence of the algorithm to the global minimum without being trapped in local minima especially for unrotated and rotated benchmark functions. Such a success is attributed mainly to an efficient compromise between exploration and exploitation due to a proper choice of the parameter *tr* and the value of maximum modification estimated by the empirical formula (10).

Furthermore, the GCO is compared with the comprehensive learning particle swarm optimizer (CLPSO) [21], using unrotated and rotated benchmark functions. The comparison reveals that the GCO is comparable to, and in many situations outperforms, the CLPSO. An additional comparison is made with a more advanced optimizer BI-population covariance matrix adaptation evolution strategy with alternative restart strategy (NBIPOP-$_a$CMA-ES)

[22,23], which is the winner of the competition on real-parameter single objective optimization at IEEE CEC-2013 [24]. The comparison reveals that the GCO is comparable to, and in some situations outperforms, the NBIPOP-$_a$CMA-ES.

With the aim of demonstrating the applicability of the GCO to real-world situations, the optimizer is used to solve optimization problems of specific MGA (multiple gravity assist) models for spacecraft trajectory design. The first model relates to the Cassini1 mission from Earth to Saturn. The objective function in this case is to be minimized. The best (least) known value of the Cassini1 objective function is 4.9307 (km/s). The GCO best found value, for FEs = 200,000 (10,000 iterations), is 4.9466. The associated average value, on 20 runs, is 5.2983. To assess the performance of the GCO for the Cassini1 problem, it is compared with the eight optimizers in: PSO, genetic algorithm, simulated annealing, differential evolution, COOP-1, 2, 3, 4. It turns out that the GCO, among the nine optimizers, is the next-best regarding the best found value, and is the best regarding the average value.

The second model relates to the GTOC1 asteroid-deflection mission from Earth to Asteroid TW229. The objective function here is to be maximized. It is again treated as a black box. The best (highest) known value of the GTOC1 objective function is 1,580,599 ($kg\,km^2/s^2$). The GCO best found value, for FEs = 200,000 (10,000 iterations), is 1,480,000. The associated average value, on 20 runs, is 1,131,267. The GCO for the GTOC1 problem is likewise compared with the eight optimizers. The result is that the GCO is the best regarding the best found value and is also the best regarding the average value.

A separate comparison is made between the GCO and CLPSO for Cassini1 and GTOC1 missions in the two cases FEs = 200,000 and FEs = 400,000. The convergence characteristics of the GCO and CLPSO, for the two missions in the two cases of FEs, are investigated.

Acknowledgments

The authors thank the anonymous reviewers for their constructive comments and suggestions.

References

[1] W. Forst, D. Hoffmann, Optimization – Theory and Practice, Springer, 2010.
[2] X.S. Yang, Engineering Optimization: An Introduction with Metaheuritics Applications, John Wiley, 2010.
[3] I. Boussaïd, J. Lepagnot, P. Siarry, A survey on optimization metaheuristics, Inf. Sci. 237 (July) (2013) 82–117.
[4] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, Boston, MA, 1989.
[5] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.
[6] R.C. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science MHS'95, IEEE Press, 1995, pp. 39–43.
[7] R.C. Eberhart, Y. Shi, J. Kennedy, Swarm Intelligence, Morgan Kaufmann, San Francisco, 2001.
[8] J. Kennedy, R.C. Eberhat, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, No. IV, IEEE Service Center, Piscataway, NJ, 1995, pp. 1942–1948.
[9] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B: Cybern. 26 (1) (1996) 29–41.
[10] B. Basturk, D. Karaboga, An artificial bee colony (ABC) algorithm for numeric function optimization, in: IEEE Swarm Intelligence Symposium, Indianapolis, IN, USA, May 12–14, 2006.
[11] D. Karaboga, An Idea Based on Honey Bee Swarm for Numerical Optimization, Technical Report-Tr06, Computer Engineering Department, Engineering Faculty, Erciyes University, Turkey, 2005.
[12] D. Karaboga, B. Akay, A survey algorithms simulating bee swarm intelligence, Art. Intell. Rev. 31 (2009) 68–85.
[13] S. Das, S. Biswas, S. Kundu, Synergizing fitness learning with proximity-based food source selection in artificial bee colony algorithm for numerical optimization, Appl. Soft Comput. 13 (2013) 4676–4694, http://dx.doi.org/10.1016/j.asoc.2013.07.009.
[14] S. Biswas, S. Das, S. Kundu, G.R. Patra, Utilizing time-linkage property in DOPs: an information sharing based artificial bee colony algorithm for tracking multiple optima in uncertain environments, Soft Comput. (2013), http://dx.doi.org/10.1007/s00500-013-1138-z.
[15] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997) 341–359, http://dx.doi.org/10.1023/A:1008202821328.
[16] S. Biswas, S. Kundu, S. Das, An improved parent-centric mutation with normalized neighborhoods for inducing niching behavior in differential evolution, IEEE Trans. Cybern. (2014), http://dx.doi.org/10.1109/TCYB.2013.2292971.
[17] S. Biswas, S. Kundu, S. Das, A.V. Vasilakos, Teaching and learning best differential evolution with self adaptation for real parameter optimization, in: IEEE Congress on Evolutionary Computation (CEC 2013), Cancun, Mexico, June 20–23, 2013, http://dx.doi.org/10.1109/CEC.2013.6557691.
[18] P. Burnard, Practical Counselling and Helping, Taylor & Francis, Routledge, 2002.
[19] D.N. Dixon, J.A. Glover, Counseling: A Problem-Solving Approach, Wiley, New York, 1984.
[20] R.C. Berg, G.L. Landreth, K.A. Fall, Group Counseling: Concepts and Procedures, 4th ed., Taylor & Francis, Routledge, 2006.
[21] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (June) (2006) 281–295.
[22] I. Loshchilov, CMA-ES with restarts for solving CEC 2013 benchmark problems, in: IEEE Congress on Evolutionary Computation (CEC 2013), Cancun, Mexico, June 20–23, 2013.
[23] I. Loshchilov, M. Schoenauer, M. Sebag, Alternative restart strategies for CMA-ES, in: Parallel Problem Solving from Nature (PPSN XII), LNCS, Springer, Taormina, Italy, 2012, pp. 296–305.
[24] J.J. Liang, B.-Y. Qu, P.N. Suganthan, A.G. Hernández-Díaz, Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session and Competition on Real-Parameter Optimization, Technical Report 201212, 2013.
[25] M.A. Eita, M.M. Fahmy, Group counseling optimization: a novel approach, in: Twenty-ninth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence (AI-2009), Cambridge, England, UK, December 15–17, 2009, pp. 195–208.
[26] H. Ali, F.A. Khan, Group counseling optimization for multi-objective functions, in: IEEE Congress on Evolutionary Computation (CEC 2013), Cancun, Mexico, June 20–23, 2013.
[27] C.B. Owen, Parameter Estimation for the Beta Distribution (M.Sc. Thesis), Department of Statistics, Brigham Young University, USA, 2008.
[28] R.C.H. Cheng, Generating beta variates with nonintegral shape parameters, Commun. ACM 21 (April (4)) (1978) 317–322.
[29] A.K. Gupta, S. Nadarajah, Handbook of Beta Distribution and Its Applications, Marcel Dekker, 2004.
[30] S.C. Esquivel, C.A. Coello Coello, On the use of particle swarm optimization with multimodal functions, Congress Evol. Comput. 2 (December) (2003) 1130–1136 (Canberra, Australia).
[31] C.Y. Lee, X. Yao, Evolutionary programming using mutations based on the levy probability distribution, IEEE Trans. Evol. Comput. 8 (February) (2004) 1–13.
[32] Z.G. Tu, L. Yong, A robust stochastic genetic algorithm (StGA) for global numerical optimization, IEEE Trans. Evol. Comput. 8 (5) (2004) 456–470.
[33] X. Yao, Y. Liu, G.M. Lin, Evolutionary programming made faster, IEEE Trans. Evol. Comput. 3 (July (2)) (1999) 82–102.
[34] D. Whitley, D. Rana, J. Dzubera, E. Mathias, Evaluating evolutionary algorithms, Art. Intell. 85 (1996) 245–276.
[35] F.V.D. Bergh, An analysis of particle swarm optimizers (Ph.D. Dissertation), Department of Computer Science, University of Pretoria, Pretoria, South Africa, 2002.
[36] A. Engelbrecht, Fundamentals of Computational Swarm Intelligence, John Wiley, 2006.
[37] D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization, IEEE Trans. Evol. Comput. 1 (April) (1997) 67–82.
[38] J.B. Kuipers, Quaternions and rotation sequences: a primer with applications to orbits, in: Aerospace and Virtual Reality, Princeton University Press, 1999.
[39] R. Salomon, Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions, BioSystems 39 (1996) 263–278.
[40] A. Ahrari, M.R. Saadatmand, M.S. Panahi, A. Ataei, On the limitations of classical benchmark functions for evaluating robustness of evolutionary algorithms, Appl. Math. Comput. 215 (2010) 3222–3229.
[41] J.J. Liang, P.N. Suganthan, K. Deb, Novel composition test functions for numerical global optimization, in: IEEE Swarm Intell. Symp., 2005, pp. 68–75.
[42] B. Addis, A. Cassioli, M. Locatelli, F. Schoen, Global optimization for the design of space trajectories, Comput. Optim. Appl. 48 (April) (2011) 635–652.
[43] T. Vinkó, D. Izzo, Global Optimisation Heuristics and Test Problems for Preliminary Spacecraft Trajectory Design, European Space Agency, Advanced Concepts Team, ACT Technical Report, 2008.
[44] J. Hartmann, V. Coverstone-Carroll, S. Williams, Generation of optimal spacecraft trajectories via a pareto genetic algorithm, J. Astronaut. Sci. 46 (1998) 267–282.
[45] O. Abdelkhalik, D. Mortari, N-impulse orbit transfer using genetic algorithms, J. Spacecraft Rockets 44 (2) (2007) 456–459.
[46] D. Izzo, V. Becerra, D. Myatt, S. Nasuto, J. Bishop, Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories, J. Global Optim. 38 (2007) 283–296.
[47] D. Izzo, First ACT global trajectory optimisation competition: problem description and summary of the results, Acta Astronaut. 61 (2007) 731–734.