Computers & Operations Research 55 (2015) 99–125

Contents lists available at ScienceDirect

Computers & Operations Research

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

A new metaheuristic for optimization: Optics inspired optimization (OIO)

Ali Husseinzadeh Kashan

Department of Industrial Engineering, Faculty of Engineering, Tarbiat Modares University, Tehran, Iran

ARTICLE INFO

Available online 23 October 2014

Keywords:
Numerical function optimization
Metaheuristics
Optics
Convex/concave mirrors

ABSTRACT

Due to the law of reflection, a concave reflecting surface/mirror causes the incident light rays to converge and a convex surface/mirror causes the light rays to reflect away so that they all appear to be diverging. These converging and diverging behaviors cause that the curved mirrors show different image types depending on the distance between the object and the mirror. We model such optical phenomena metaphorically into the searching process of numerical optimization by a new algorithm called optics inspired optimization (OIO). OIO treats the surface of the numerical function to be optimized as a reflecting surface in which each peak is assumed to reflect as a convex mirror and each valley to reflect as a concave one. Each individual is assumed to be an artificial object (or light point) that its artificially glittered ray is reflected back by the function surface, given that the surface is convex or concave, and the artificial image is formed (a candidate solution is generated within the search domain) based on the mirror equations adopted from physics of optics. Besides OIO, we introduce different variants of it, called ROIO (Rotation based OIO), and COIO (Convex combination based OIO) algorithms and conduct an extensive computational effort to find out the merit of the new algorithms. Our comparisons on benchmark test functions and a real world engineering design application (i.e., optimization of a centrifuge pump) demonstrate that the new algorithms are efficient and compete better than or similar to most of state of the art optimization algorithms with the advantage of accepting few input parameters.

1. Introduction

Since the 1970s that the idea of a general algorithmic framework (which can be applied with relatively few modifications to different optimization problems) emerged, many algorithms have been introduced which have got their source of inspiration from nature, society, culture, politics, human, etc. The term "metaheuristic" is used for such methods that combine rules and randomness while imitating natural phenomena. This field of research has attracted many researchers from various fields of science in recent years. This interest is by far in applying the existing metaheuristics for solving real word optimization problems in many fields such as business, industry, engineering, etc. However, beside all of these applications, occasionally a new metaheuristic is introduced which uses a novel metaphor as a guide for searching process in optimization. For example, the particle swarm optimization algorithm (PSO), introduced in 1995 [1], models the flocking behavior of birds; the harmony search (HS), introduced in 2001 [2], is conceptualized using the musical process of searching for a perfect state of harmony; the bacterial foraging optimization algorithm (BFOA), introduced in 2002 [3], models foraging as an optimization process where an animal seeks to maximize energy per unit time spent for foraging; the bacterial chemotaxis algorithm (BC), introduced in 2002 [28], has been proposed by analogy to the way bacteria react to chemoattractants in concentration gradients; the artificial bee colony algorithm (ABC), introduced in 2005 [4], simulates the intelligent foraging behavior of a honeybee swarm; the central force optimization algorithm (CFO), introduced in 2007 [5], makes an analogy between the process of searching a decision space for the maxima of an objective function and flying probes through physical space under the influence of gravity; the fire fly algorithm (FA), introduced in 2008 [6], performs based on the idealization of flashing characteristics of fireflies; the league championship algorithm (LCA), introduced in 2009 [7], simulates the competitions followed in league games and models the match analysis process in which coaches modify their arrangement on the basis of their own game experiences and their opponent's style of play; the group search optimizer (GSO), introduced in 2009 [8], simulates the animal searching behavior to find resources such as food, mates, oviposition, or nesting sites; The gravitational search algorithm (GSA), introduced in 2009 [29], has been constructed based on the law of gravity and the notion of mass interactions. The teaching–learning based optimization (TLBO), introduced in 2011 [9] mimics teaching–learning process in a class between the teacher and the students (learners); Krill herd algorithm (KH)

E-mail address: a.kashan@modares.ac.ir

http://dx.doi.org/10.1016/j.cor.2014.10.011
0305-0548/

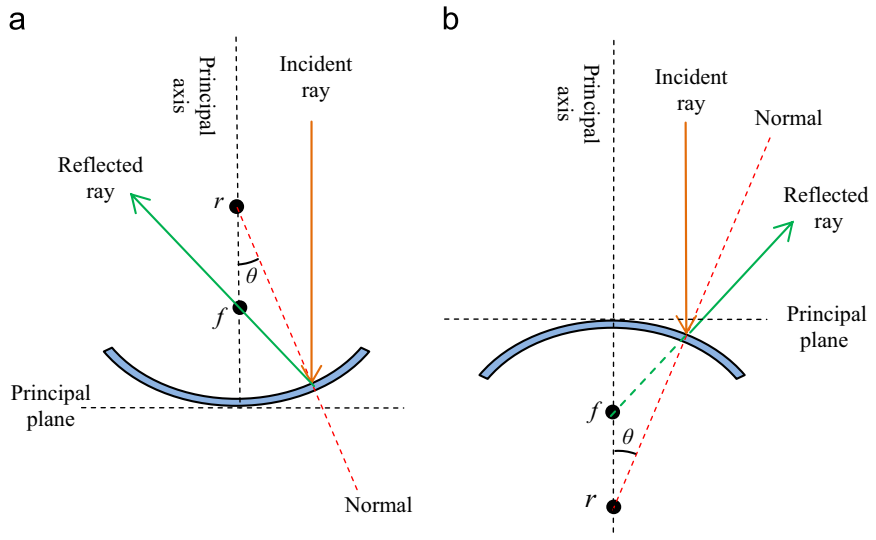

Fig. 1. Light rays traveling parallel to the axis of (a) a concave mirror are reflected so that they all pass approximately through a common focal point *f*, (b) a convex mirror are reflected so that they appear to come from a focal point *f*, located behind the mirror.

introduced in 2012 [10] works based on the simulation of the herding of the krill swarms in response to specific biological and environmental processes, etc.

There are many ways to classify metaheuristic algorithms. One way is based on the type of metaphor used to develop the algorithm. For example nature inspired algorithms (e.g., evolutionary algorithms, ant colony optimization and simulated annealing algorithm), social inspired algorithms (e.g. Tabu search, league championship algorithm [11,12]) or political inspired algorithms (e.g., Imperialist competitive algorithm [13]). Nature inspired algorithms solve the optimization problems by simulating a developing process in nature, which is considered as a new effective way for optimization. In these algorithms, the global convergence is realized by simulating the physical or ecological process in nature and the optimization mechanisms of algorithms themselves.

There are several nature inspired algorithms which adopt their source of inspiration from physics, e.g., ray optimization [24], spiral dynamics inspired optimization [23], central force optimization [5], etc. In this paper a new nature inspired algorithm is introduced based on the optical characteristics of concave and convex mirrors, called optics inspired optimization (OIO). It has been observed that concave surfaces/mirrors reflect the light rays toward the principal axis (a line passing through the center of the mirror and perpendicular to the mirror). Such a mirror causes light rays to converge (see Fig. 1a). On the other side, by a convex surface/mirror light rays coming in parallel to the principal axis are reflected away from the principal axis so that they all appear to be diverging (see Fig. 1b). We model such an optical observation metaphorically into the searching process of numerical optimization by the so called optics inspired optimization algorithm. OIO treats the surface of the numerical function to be optimized as a reflecting surface wherein each peak is assumed to reflect as a convex mirror and each valley to reflect as a concave one. Each point in the joint search/solution and objective space (a subset in R^{n+1}) which is mapped as a solution within the search space (a subset in R^n) is assumed to be an artificial light point. In this way, the artificial ray glittered from an artificial light point is reflected back artificially by the function surface, given that the reflecting surface is partially a part of a peak or a part of a valley, and the artificial image point (a new point in the joint search and objective space which is mapped as a new solution in the search/solution space) is formed upright (toward the light point position in the search space) or inverted (away from the light point position in the search space). Such a model gives us the ability to carry out both exploration and exploitation tasks during the search for optimum.

In optics, spherical mirrors (concave or convex mirrors) suffer from spherical aberration phenomenon which results in an imperfection of the produced image. Such a phenomenon can also be observed artificially in OIO which may result in a less converging behavior of algorithm. We correct the artificial spherical aberration in our algorithm as it is corrected in practice.

In the remainder of the paper we describe the algorithm in details. In Section 2, we review the optical basics and definitions. Especially, those definitions which will be used metaphorically in OIO. In Section 3 we introduce the basic steps of OIO. Section 4 introduces other versions of OIO which work based on the relaxation of the reflection law of the physics. Section 5 deals with computational experiments and comparisons. In this section we investigate the performance of OIO and its variants on several sets of numerical benchmark functions, which are used in 4 rounds of experiments. A real world application in engineering design of centrifuge pumps is also considered in Section 5. In each experiment we compare the performance of OIO based algorithms with a number of advanced comparator algorithms. Finally Section 6 concludes the paper.

2. Optics related background

The content of this section is mainly related to the theory of optics. However, we only use those materials which will be used metaphorically in OIO. For more information the interest reader may refer to the books of Jenkins and White [15], Zitzewitz et al. [14], and Griffith and Brosing [16].

Optics is a branch of physics which involves the behaviour and properties of light, including its interactions with matter and the construction of instruments that use or detect it. Practical applications of optics are found in a variety of technologies and everyday objects, including mirrors, lenses, telescopes, microscopes, etc.[1] A *curved* or *spherical mirror* is a mirror with a curved reflective

[1] http://en.wikipedia.org/wiki/Optics.

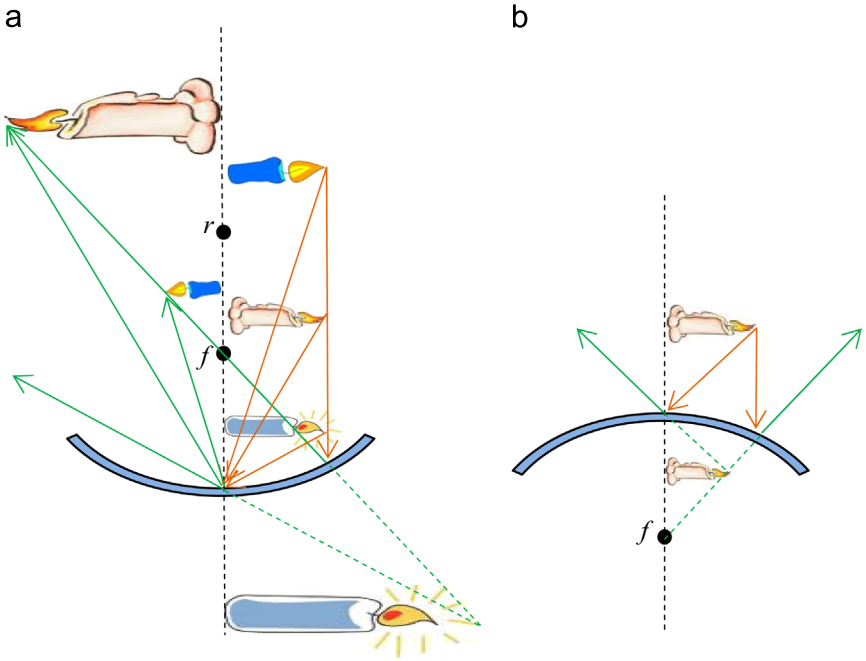

Fig. 2. Effect on image of object's position relative to mirror focal point *f*. (a) Concave mirror, (b) convex mirror.

surface, which may be either convex (bulging outward) or concave (bulging inward). Most curved mirrors have surfaces that are shaped like part of a sphere[2]. The behaviour of light reflected by a curved mirror is subject to the same laws as that of plane mirrors, which are known as the *laws of reflection*. The first law: the incident ray, the reflected ray, and the normal all lie on the same plane. The second law: the angle between the incident ray and the normal is equal to the angle between the reflected ray and the normal.

A *concave mirror* has a reflecting surface that bulges inward (away from the incident light). Concave mirrors reflect light inward to focal point. They are used to focus light. Concave mirrors show different image types depending on the distance between the object and the mirror. These mirrors are called "converging" because they tend to collect light that falls on them, refocusing parallel incoming rays toward a focus. Concave mirrors are used in some telescopes. The image on a concave mirror is virtual, upright and larger than the object if it is placed between focal point and mirror. If the object is placed beyond the focal point, its image is always real and inverted. But the image size depends on the position of the object (see Fig. 2a).

A *convex mirror* is a curved mirror in which the reflective surface bulges toward the light source. Convex mirrors reflect light outwards and always form a virtual image, since the focus (f) and the centre of curvature (r) are both imaginary points "inside" the mirror, which cannot be reached. A collimated beam of light diverges after reflection from a convex mirror, since the normal to the surface differs with each spot on the mirror. The image on a convex mirror is always virtual, upright and smaller than the object. The ray diagram in Fig. 2b represents how an image is formed by a spherical convex mirror. The figure uses two rays but remember that there are an infinite number of rays. Indeed, it is adequate to draw only two rays to locate the image of a point on an object. Therefore, we have taken just two rays coming from the top of object (candles in our case) and traced them as shown in Fig. 2. When these two rays are extended backward (or forward in case of a concave mirror), their intersection locates the image point position. In a similar way, the two rays coming from each point of the object can be traced and the corresponding image point can be obtained, accordingly. In this way we can see the whole image of the object.

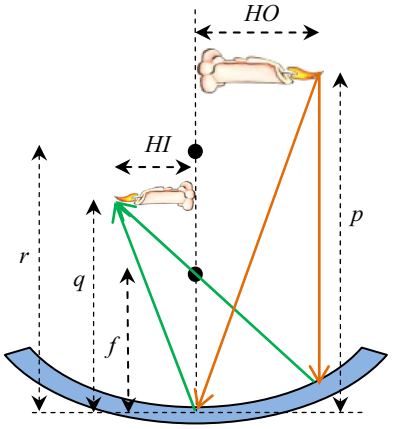

Fig. 3. When an object lies in distance p from the mirror, its image is formed in distance q from the mirror.

The spherical mirror model can be used to develop a simple equation for spherical mirrors. Using triangle relationship and the laws of reflection, it is also possible to develop a quantitative relationship between the object and image distances. Let f be the focal length, r be the radius of curvature ($r=2f$), p be the object position, and q be the image position (see Fig. 3). The mirror equation is approximately given by (see [14]):

$$\frac{2}{r}=\frac{1}{p}+\frac{1}{q}\Rightarrow q=\frac{rp}{2p-r} \tag{1}$$

All distances are measured from the vertex, the point where the axis meets the mirror. In general, distances are positive if they lie on the same side of the mirror as the light rays themselves. If they lie behind the mirror, the distances are negative. In other words, both of r (or f) and q are negative for a convex mirror and only q is negative for a concave mirror just when the object lies between the vertex and the focal point.

[2] http://en.wikipedia.org/wiki/Curved_mirror.

Magnification (m) is another property of a spherical mirror, which determines how much larger or smaller the image is relative to the object. In practice, this is a simple ratio of the image height (HI) to the object height (HO). Using triangle geometry, this ratio can be written in terms of image position and object position as follows [14]:

$$m = -\frac{q}{p} = \frac{HI}{HO} \Rightarrow HI = -HO\frac{q}{p} \tag{2}$$

Table 1 summarizes the properties of different mirrors with objects located on the principal axis of the mirror [14]. Virtual images are always behind the mirror, which means that the image position is negative. When the absolute value of a magnification is less than one, the image is smaller than the object and when the absolute value of the magnification is greater than one, the image is larger than the object. A negative magnification means that the image is inverted relative to the object.

It should be noted that Eq. (1) is approximately correct. To form an image, this equation uses only rays that are close to and almost parallel with the principal axis. Such a situation is physically imposed by assuming that the angle the incident beam makes with the axis is small enough to make the approximation $\sin(\theta) \approx \theta$ for rays coming from the axis. In reality, the ray coming from an object toward a mirror is diverging, so not the entire ray is close to or parallel to the axis. Rays that are far from principal axis do not converge to a single point. The fact that a spherical mirror does not bring all parallel rays to a single point is known as *spherical aberration* (see Fig. 4). This defect is most noticeable for light rays striking the outer edges of the mirror. Rays that strike the outer edges of the mirror fail to focus in the same precise location as light rays that strike the inner portions of the mirror. The result is that the images of objects as seen in spherical mirrors are often blurry.

The extent of the ray divergence (κ) from the focus (see Fig. 4b), which is called the *lateral aberration*, can be quantized in terms of the distance HO of the light ray from the principal axis of a concave mirror with the radius of curvature r. The lateral aberration κ is thus given by [17]:

$$\kappa = \frac{r^2}{2\sqrt{r^2 - HO^2}} - \frac{r}{2} \tag{3}$$

For example, a beam of width 26 cm will focus within 1.7 cm of the focal point on a mirror of radius of curvature 1 m. Spherical aberration can be minimized by using a mirror whose height is small compared with the radius of curvature. Eq. (3) predicts that when HO is kept constant and the radius of curvature gets larger, the lateral aberration κ decreases. We use such a mechanism for correcting artificial spherical aberration in OIO.

3. Optics inspired optimization algorithm (OIO)

In this section we show how the above physical perceptions can be artificially modeled to develop an efficient algorithm for optimization, called OIO. OIO is optics inspired population based evolutionary algorithm in which it is assumed that a number of artificial light points (points in R^{n+1} whose mapping in R^n are potential solutions to the problem) are sitting in front of an artificial wavy mirror (function surface) reflecting their images. OIO treats the surface of the function to be optimized as the reflecting mirror composed of peaks and valleys. (In the remainder of paper we use both "mirror" and "function surface", equivalently.) Each peak is treated as a convex reflective surface and each valley is treated as a concave reflective surface. In this way, the artificial ray glittered from an artificial light point is reflected back artificially by the function surface, given that the reflecting surface is partially a part of a peak or a part of a valley, and the artificial image point (a new point in R^{n+1} which is mapped in R^n as a new solution in the search domain) is formed upright (toward the light point position in the search space) or inverted (outward the light point position in the search space).

Fig. 5 demonstrates the idea behind OIO to generate new solutions in the one dimensional search space. In this figure it is assumed that an artificial light point (object) in the joint search and objective space is in front of the function surface (mirror) in a particular distance from the vertex (values on the X-axis form the search/solution space and values on the $f(X)$-axis form the objective space. The set of all points in the $X-f(X)$-coordinate system forms the joint search and objective space). The artificial image is formed in the joint search and objective space and its position and height is determined through mirror (Eq. (1)) and magnification (Eq. (2)) equations. Finally, mapping the artificial image position into the search space results the position of a new solution in the search space. Depending on the type of the reflecting part of the function surface (convex or concave) and depending on the position of the artificial light point (object) in the joint search and objective space, there are four different situations under which new solutions are generated (see Fig. 5).

The idea behind OIO is thus simple. Given an individual solution O in the population, a different solution F (vertex point) is picked randomly from the population. If F is worse than O, in terms of function/objective value, then it is assumed that the surface is convex and a new solution is generated upright somewhere toward O, on the line connecting O and F (see Fig. 5a). If F is better than O then it is assumed that the surface is concave and the

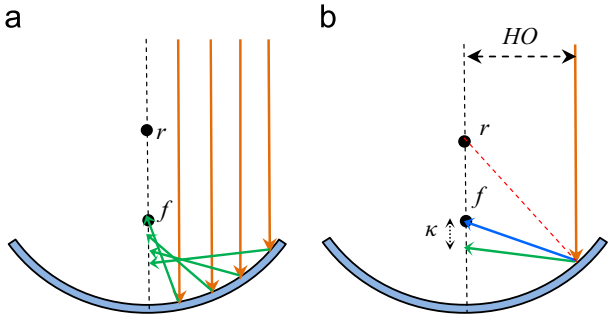

Fig. 4. (a) Spherical mirrors reflect rays such that they converge at points other than the focus f (b) lateral aberration (κ) is the extent of ray divergence from the focus.

Table 1
Properties of different mirrors.

Mirror	**Concave**			**Convex**
f	Positive			Negative
p	$p > r$	$r > p > f$	$f > p > 0$	$p > 0$
q	$r > q > f$	$q > r$	$\|q\| > p$ & Negative	$\|f\| > \|q\| > 0$ & Negative
Image size (HI)	Smaller	Larger	Larger	Smaller
Image direction	Inverted	Inverted	Upright	Upright
Image type	Real	Real	Virtual	Virtual

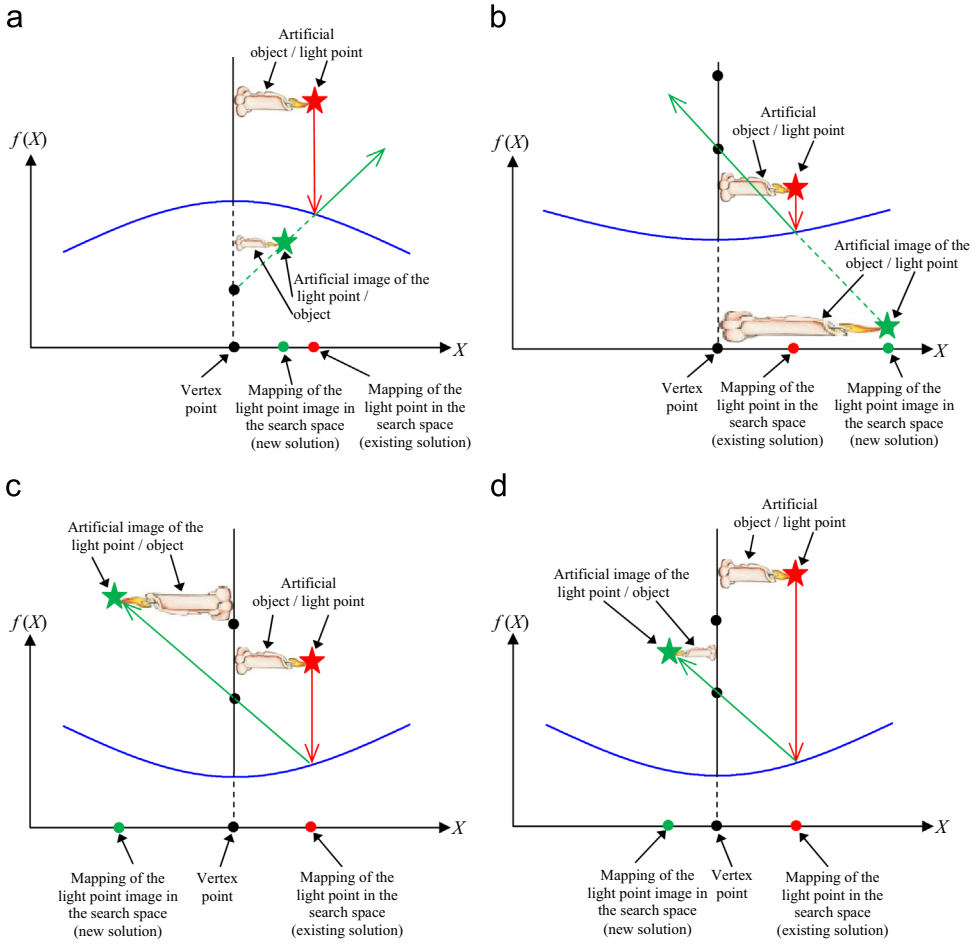

Fig. 5. The idea behind generation of the new solutions in OIO, (a) When the function surface serves as a convex surface. (b) When the function surface serves as a concave surface and the artificial object/light point is between artificial focal point and the function surface. (c) When the function surface serves as a concave surface and the artificial object/light point is between artificial focal point and centre of curvature. (d) When the function surface serves as a concave surface and the artificial object/light point is beyond the centre of curvature.

new solution is generated upright toward (see Fig. 5b) or inverted outward (see Fig. 5c and d) O, on the line connecting O and F in the search space (later we will see that OIO may uses from the weighted linier combination of these new solutions).

As can be seen from the conceptual model of Fig. 5, the logic of OIO to generate new solutions is able to serve exploration and exploitation during the search for an optimum. Exploration can be controlled relatively via allowing a larger jump in the solution space (see Fig. 5b and c) while exploitation can be carried out by allowing a relatively smaller jump over the base solutions (see Fig. 5a and d). In the remainder of this section we first introduce the notations used to develop OIO and then present its algorithmic steps.

3.1. Notations and formulation

Let $f(\vec{X}=[x_1x_2\cdots x_n])$ be an n variable numerical function that should be minimized over the n-dimensional decision space defined by $l_d \le x_d \le u_d, d=1,..,n$. Recall that $f: R^n \to R$ and we are seeking the global minimum of f that is, finding $\vec{X}^* \in R^n$ such that $f(\vec{X}^*) \le f(\vec{X}), \forall \vec{X} \in R^n$. It is worth to note that the joint search and objective space is a subset of R^{n+1} where $[x_1x_2\cdots x_n f(\vec{X})]_{1\times(n+1)}$ is a vector in this subset. In our notations, we define (see Figs. 6 and 7 for the case of $n=1$):

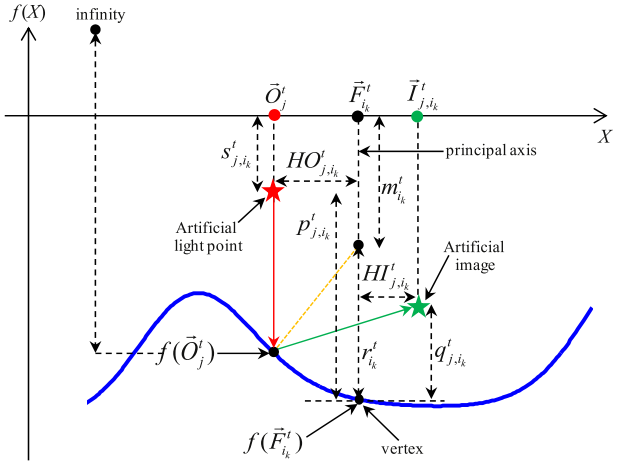

Fig. 6. When the surface of the function serves as a concave mirror.

- $\vec{O}_j^t = [o_{j1}^t o_{j2}^t \cdots o_{jn}^t]_{1\times n}$ to address the position of artificial light point j in the n dimensional search space in iteration t (i.e., the jth solution in the population),
- $\vec{F}_{i_k}^t = [f_{i_k1}^t f_{i_k2}^t \cdots f_{i_kn}^t]_{1\times n}$ to address a different point in the search space (i.e., an individual in the population) which passes the artificial principal axis through itself. The artificial mirror vertex position is thus determined by the vector $[f_{i_k1}^t f_{i_k2}^t \cdots f_{i_kn}^t f$

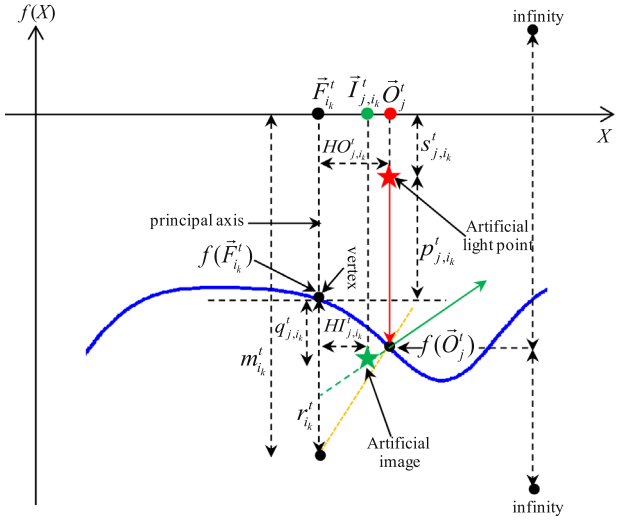

Fig. 7. When the surface of the function serves as a convex mirror.

$(\overrightarrow{F}^t_{i_k})]_{1\times(n+1)}$. Index i_k is drawn randomly from {1,…, *NO*}. *NO* is the number of artificial light points (i.e., the population size).

- $\overrightarrow{I}^t_{j,i_k}=[i^t_{j1}i^t_{j2}\cdots i^t_{jn}]_{1\times n}$ to address an image position of the artificial light point j in the search space in iteration t. The artificial image is formed by the artificial mirror whose principal axis passes through $\overrightarrow{F}^t_{i_k}$.
- s^t_{j,i_k} as the position of the artificial light point j (whose image is formed by the artificial mirror) on the function/objective axis (objective space) in iteration t. The position of artificial light point j in the joint search and objective space is thus given by the vector $[o^t_{j1}o^t_{j2}\ldots o^t_{jn}s^t_{j,i_k}]_{1\times(n+1)}$,
- p^t_{j,i_k} as the distance between the position of artificial light point j on the function/objective axis and the position of artificial mirror vertex on the function/objective axis (i.e., $f(\overrightarrow{F}^t_{i_k})$) in iteration t,
- q^t_{j,i_k} as the distance between the image position of the artificial light point j on the function/objective axis and the position of artificial mirror vertex on the function/objective axis (i.e., $f(\overrightarrow{F}^t_{i_k})$)in iteration t,
- $r^t_{i_k}$ as the radius of curvature of the artificial mirror whose center of curvature is on the principal axis which passes through $\overrightarrow{F}^t_{i_k}$,
- $m^t_{i_k}$ as the position of the centre of curvature on the function/objective axis (objective space),
- HO^t_{j,i_k} as the height of the artificial light point j from artificial principal axis in iteration t,
- HI^t_{j,i_k} as the image height of the artificial light point j from artificial principal axis in iteration t,
- κ^t_{j,i_k} as the value of lateral aberration relevant to the artificial mirror which is reflecting the image of the artificial light point j in iteration t,

The overall mechanism of OIO can be formulated as follows. At first, *NO* number of individuals are generated randomly to form the initial position of the artificial light points in the search space. Thereafter, in iteration t, each artificial light point j with position $\overrightarrow{O}^t_j=[o^t_{j1}o^t_{j2}\cdots o^t_{jn}]$ (j=1,…, *NO*) in the search space sits (in the joint search and objective space in position $[o^t_{j1}o^t_{j2}\cdots o^t_{jn}s^t_{j,i_k}]$) in front of the artificial mirror (function surface) in distance p^t_{j,i_k} from the mirror vertex and its artificial image is formed in the joint search and objective space in distance q^t_{j,i_k} (on the function/objective axis) from the vertex, given that the principal axis passes through point $\overrightarrow{F}^t_{i_k}$ in the search space ($\overrightarrow{F}^t_{i_k}$ is selected randomly from the current population with this condition that $f(\overrightarrow{F}^t_{i_k})$ must differ from$f(\overrightarrow{O}^t_j)$) and the artificial mirror radius of curvature is $r^t_{i_k}$. Mapping the artificial image position into the solution space produces an artificial image position $\overrightarrow{I}^t_{j,i_k}$ in the search space which can be treated as a new solution to the problem.

Just similar to spherical mirrors which suffer from spherical aberration phenomenon, OIO may not reflect satisfactory performance on some problems due to such an artificial phenomenon. We correct the spherical aberration in our algorithm as it is corrected in practice

3.2. Generation of a new solution in OIO

Following our discussions in Section 2 about the light reflection from spherical mirrors and the overall idea behind OIO algorithm which was presented in Sections 3 and 3.1, in this section we are going to develop OIO's equation to generate a new solution in the search space. As was already mentioned, the function surface serves as a reflecting mirror in OIO. Different segments of the function surface will serve as concave or convex mirrors.

Let us consider the artificial light point j with position $\overrightarrow{O}^t_j=[o^t_{j1}o^t_{j2}\cdots o^t_{jn}]$ in the search space and position $[o^t_{j1}o^t_{j2}\cdots o^t_{jn}s^t_{j,i_k}]$ in the joint search and objective space, for which we are interested to find the image characteristics (e.g., position and height). We first determine whether the light point is sitting in front of a convex segment or a concave segment of the function surface. To decide, we pick randomly an individual $\overrightarrow{F}^t_{i_k}$ from the current population which differs from $\overrightarrow{O}^t_j$ (that is, $i_k\in$ {1,…, *NO*} such that $i_k\neq j$ and $f(\overrightarrow{F}^t_{i_k})\neq f(\overrightarrow{O}^t_j)$) and assume that it is a vertex point in the search space which passes the principal axis of the artificial mirror through itself. Now, if $f(\overrightarrow{O}^t_j)>f(\overrightarrow{F}^t_{i_k})$ we conclude that the function surface is concave (see Fig. 6) and if $f(\overrightarrow{O}^t_j)<f(\overrightarrow{F}^t_{i_k})$ we conclude that the function surface is convex (see Fig. 7). Figs. 6 and 7 demonstrate such conclusions graphically for the case of n=1.

In optics, the mirror equations (Eqs. (1) and (2)) have been extracted in the presence of a one dimensional (n=1) object (the mirror is thus a concave/convex curve). It is not difficult to see that they work for 2d objects (n=2), too (here, the mirror is a concave/convex hull). We therefore generalize these equations so that they work for higher (say n) dimensional objects hypothetically.

Back to the one dimensional case, let us assume that the function surface in front of the artificial light point j is concave (see Fig. 6). The position s^t_{j,i_k} of the artificial light point j on the function/objective axis (objective space) is determined somewhere between $f(\overrightarrow{O}^t_j)$ and positive infinity. We set the value of s^t_{j,i_k} equal to a random value $U[f(\overrightarrow{O}^t_j),f(\overrightarrow{O}^t_j)+d_\infty]$, where it is assumed that $f(\overrightarrow{O}^t_j)+d_\infty$ is the physical infinity (we use $U[a,b]$,to denote a random value distributed uniformly between a and b). d_∞ is itself the physical infinity and can accept any positive value. Starting with $d_\infty=|\max_{j=1,\ldots,NO}\{f(\overrightarrow{O}^1_j)\}|$, we update d_∞ whenever we correct the artificial spherical aberration in OIO (see Section 3.3). Based on our definition of p^t_{j,i_k} we can write

$$p^t_{j,i_k}=s^t_{j,i_k}-f(\overrightarrow{F}^t_{i_k}) \tag{4}$$

In a similar way we can assume that the position $m_{i_k}^t$ of the centre of curvature of the artificial concave mirror on the function/objective axis (objective space) is somewhere between $f(\vec{O}_j^t)$ and positive infinity. Therefore, we determine the value of $m_{i_k}^t$ randomly as $U[f(\vec{O}_j^t), f(\vec{O}_j^t)+d_\infty]$, where we assume that $f(\vec{O}_j^t)+d_\infty$ is the physical infinity. Based on our definition of $r_{i_k}^t$ we can write:

$$r_{i_k}^t = m_{i_k}^t - f(\vec{F}_{i_k}^t) \tag{5}$$

Now, let us assume that the function surface in front of the artificial light point j is convex (see Fig. 7). The position s_{j,i_k}^t of the artificial light point j on the function/objective axis (objective space) is somewhere between $f(\vec{F}_{i_k}^t)$ and positive infinity. We set the value of s_{j,i_k}^t randomly as $U[f(\vec{F}_{i_k}^t), f(\vec{F}_{i_k}^t)+d_\infty]$, where it is assumed that $f(\vec{F}_{i_k}^t)+d_\infty$ is the physical infinity. Similar to the case of artificial concave mirror, in case of artificial convex mirror, p_{j,i_k}^t is obtained by Eq. (4), too.

From Fig. 7 we can assume that the position $m_{i_k}^t$ of the centre of curvature of the artificial convex mirror on the function/objective axis (objective space) is somewhere between negative infinity and $f(\vec{O}_j^t)$.Therefore, we set the value of $m_{i_k}^t$ randomly as $U[f(\vec{O}_j^t) - d_\infty, f(\vec{O}_j^t)]$, where we assume that $f(\vec{O}_j^t)-d_\infty$ is the physical negative infinity. Just similar to the case of artificial concave mirror, in case of a convex mirror, $r_{i_k}^t$ is also obtained by Eq. (5). However, unlike the case of an artificial concave mirror, here $r_{i_k}^t$ is negative (which is in accordance with the contents of Table 1).

Once the value of p_{j,i_k}^t and $r_{i_k}^t$ were determined, we can use the mirror equation to obtain q_{j,i_k}^t as follows:

$$\frac{2}{r_{i_k}^t} = \frac{1}{p_{j,i_k}^t} + \frac{1}{q_{j,i_k}^t} \Rightarrow q_{j,i_k}^t = \frac{r_{i_k}^t p_{j,i_k}^t}{2p_{j,i_k}^t - r_{i_k}^t}. \tag{6}$$

Depending on the type of the function surface (convex or concave) and the relative position of artificial light point j on the function/objective axis to the position of the center of curvature, q_{j,i_k}^t can be either positive or negative. Given the fact that:

$$HO_{j,i_k}^t = \|\vec{O}_j^t - \vec{F}_{i_k}^t\|, \tag{7}$$

Based on the magnification equation (Eq. (2)), we obtain the image height of the artificial light point j as follows:

$$HI_{j,i_k}^t = -HO_{j,i_k}^t \frac{q_{j,i_k}^t}{p_{j,i_k}^t}, \tag{8}$$

which we treat it as a step length to generate a new solution. Using vector algebra, we can now generate the image position of the artificial light point j in the search space in iteration t as follows:

$$\begin{aligned}
\vec{I}_{j,i_k}^t &= \vec{F}_{i_k}^t + HI_{j,i_k}^t \frac{(\vec{O}_j^t - \vec{F}_{i_k}^t)}{\|\vec{O}_j^t - \vec{F}_{i_k}^t\|} \\
&= \vec{F}_{i_k}^t - HO_{j,i_k}^t \frac{q_{j,i_k}^t}{p_{j,i_k}^t} \frac{(\vec{O}_j^t - \vec{F}_{i_k}^t)}{\|\vec{O}_j^t - \vec{F}_{i_k}^t\|} \\
&= \vec{F}_{i_k}^t - \frac{q_{j,i_k}^t}{p_{j,i_k}^t}(\vec{O}_j^t - \vec{F}_{i_k}^t) \\
&= \vec{F}_{i_k}^t - \frac{r_{i_k}^t}{2p_{j,i_k}^t - r_{i_k}^t}(\vec{O}_j^t - \vec{F}_{i_k}^t)
\end{aligned} \tag{9}$$

In this way, $\vec{I}_{j,i_k}^t$ may be entered as a new solution (search direction) in the population if it produces a better function value than $\vec{O}_j^t$. However, to preserve more diversity in the population it seems beneficial to use a consequent of multiple search directions instead of a single search direction generated by Eq. (9) in OIO. Let $\vec{I}_{j,i_k}^t$ be a typical search direction obtained by Eq. (9). Based on the notion of using multiple search directions to generate a new consequent image position (solution) we write:

$$\vec{I}_j^t = \sum_{k=1}^{K} w_k^t \vec{I}_{j,i_k}^t \tag{10}$$

where $\vec{I}_j^t$ is the weighted linear combination of different search directions $\vec{I}_{j,i_k}^t, \forall k$ and is treated as the consequent of different images of the artificial light point j in different mirrors. The random weights w_k^t are generated such that $\sum_{k=1}^{K} w_k^t = 1$, $0 < w_k^t \leq 1$. Generally $K=1$ or $K=2$ would produce satisfactory results (for all experiments of Section 5 we set $K=1$ except for Experiment 4 for which we use from $K=2$. Usually, $K=2$ provides the same pattern of performance achieved by using $K=1$ for the rest of experiments, at the expense of increasing computation times). For each value of k, i_k is selected via the roulette wheel approach. Eq. (10) says that to generate a new solution in OIO we may generate multiple artificial image points $\vec{I}_{j,i_k}^t, \forall k$ and introduce their weighted mean as the new solution $\vec{I}_j^t$.

Once a complete solution was generated by Eq. (10), and before any evaluation being carried out on the quality of the solution, its feasibility is checked with respect to the range constraints and values outside of the ranges are inserted back into their range randomly.

The feasible solution $\vec{I}_j^t$ generated by Eq. (10), differs from $\vec{O}_j^t$ in all dimensions. However on many functions, due to the early convergence of the algorithm to local optima, it may not be a good choice to make changes in all dimensions. For some functions, preserving a change only in one dimension of $\vec{O}_j^t$ may produce more satisfactory results (for example, while the classic PSO algorithm do changes in all dimensions, ABC changes only in one dimension). Let c denotes the number of changes made in $\vec{O}_j^t$. Our investigations indicate that, depending on the problem, either very small (e.g., $c=1$) or very high (e.g., $c=n-1$ or $c=n$) values of c produce satisfactory results. When $c=n$, the new solution $\vec{I}_j^t$ is generated and a greedy selection is carried out between $\vec{O}_j^t$ and $\vec{I}_j^t$. If $f(\vec{I}_j^t)$ is better than $f(\vec{O}_j^t)$, then we replace the jth member of the population with $\vec{I}_j^t$. When $c<n$, we first generate the solution $\vec{I}_j^t$ and set $\vec{U}_j^t \leftarrow \vec{O}_j^t$. c components are selected randomly from $\vec{I}_j^t$ and their values are assigned to their corresponding components in $\vec{U}_j^t$. Finally, a greedy selection is carried out between $\vec{O}_j^t$ and $\vec{U}_j^t$. If $f(\vec{U}_j^t)$ is better than $f(\vec{O}_j^t)$, then we replace the jth member of the population with $\vec{U}_j^t$.

To remedy the problem of setting a proper value for c, we can tune its value for each problem separately. Although this approach is effective, it is not efficient and is not reasonable for evaluation purpose. Instead, it is preferred to solve each set of functions with a single parameter value. We know that while for some functions a small value of c is preferred to be used for generation of all

individuals in the population, for some other functions a large value of c is suitable. In our implementation of OIO for optimizing all functions included in all experiments, we divide the population into two subpopulations. Now the value of c is kept low (e.g., $c=1$) for the first subpopulation and high (e.g., $c=n-1$ or $c=n$) for the second subpopulation. It must be noted that the basic model of OIO has been introduced based on a single population of individuals.

Our initial investigations show that when $c=n$ for all individuals in the population (when there are no subpopulations) or for all individuals in the second subpopulation, the algorithm converges too early, resulting in being suboptimal. To heal the premature convergence, when the value of c is considered equal to n for object j, to generate $\vec{I}_j^t$ we add a random perturbation term to the right side of Eq. (9) to improve the exploration ability of OIO and its variants (see Section 4). The perturbation term is equal to $\delta_j^t \times brand_j^t \times (rand_j^t - 1/2)$, where δ_j^t controls the perturbation magnitude and its value can be set equal to HO_{j,i_k}^t. As the population evolves, we expect that δ_j^t gets smaller and therefore the perturbation magnitude gets smaller, too. $rand_j^t$ is a random number with value equal to $U[0,1]$. $brand_j^t$ is a Bernoulli random number which is equal to 1 with probability $1-\frac{NO\times t+j}{FE_{max}}$, where FE_{max} is the maximum allowable number of function evaluations. As can be seen, the probability value decreases as the search proceeds, and this makes the effect of random perturbation less and less during the evolution process.

Before leaving this section, we should point out that in our implementation of OIO and its variants (see Section 4), we use from the roulette wheel selection strategy in "Mirror type determination" step of OIO to select the individual solution $\vec{F}_{i_k}^t$ from the current population (see the flowchart of OIO). However, since the function to be optimized is a minimization problem, we have to convert the objective function values to the fitness function values of maximization form. One way is to use the simplest procedure to get the fitness value relevant to solution j ($fit(\vec{O}_j^t)$) from its function value ($f(\vec{O}_j^t)$) and the maximum function value of the population as follows: $fit(\vec{O}_j^t) = \max_{i=1,\ldots,NO}\{f(\vec{O}_i^t)\} - f(\vec{O}_j^t)$. The other way, which is used in our implementation of OIO and its variants, is to use the fitness assignment method proposed by Karaboga [4] as follows:

$$fit(\vec{O}_j^t) = \begin{cases} \frac{1}{1+f(\vec{O}_j^t)} & f(\vec{O}_j^t) \geq 0 \\ 1+abs(f(\vec{O}_j^t)) & f(\vec{O}_j^t) < 0 \end{cases} \tag{11}$$

It is worth mentioning that the roulette wheel selection is every time conducted between the members of the same subpopulation. However, the global best solution always participates in the roulette wheel selection.

3.3. Correcting the artificial spherical aberration in OIO

As was already mentioned, spherical aberration occurs in a spherical mirror when the rays are far from the principal axis relative to the radius of curvature. Large values of lateral aberration κ indicates that the width (HO) of incoming ray is large compared to the radius of curvature (r).

In analogy with what happens in optics, spherical aberration may occur artificially in OIO, which results in a weak convergence of algorithm. Later in Section 5.6, we will give an example in which correcting the spherical aberration has a significant effect on the quality of the final results obtained by OIO algorithm. Correction of spherical aberration (or decreasing the lateral aberration) in OIO is closely related to modify the value of $r_{i_k}^t$.

In course of search for the optimum by OIO, it is possible (especially at the start of the search) to repeatedly obtain $HO_{j,i_k}^t > \left|r_{i_k}^t\right|$ (since $r_{i_k}^t$ is negative for an artificial convex mirror, we use its absolute value), which means that the artificial light point j is at the height that cannot be seen by the artificial mirror (and the lateral aberration amount is undefined). To be diagnosed in the mirror, the mirror absolute radius of curvature ($|r_{i_k}^t|$) must be greater than HO_{j,i_k}^t. Therefore, when computing $r_{i_k}^t$ in Eq. (5), if we get $HO_{j,i_k}^t > \left|r_{i_k}^t\right|$, we must correct $r_{i_k}^t$. We correct $r_{i_k}^t$ via increasing the length of the artificial mirror radius of curvature same as the case of correcting spherical aberration as follows.

Now we must correct the artificial spherical aberration which may occur in OIO. Given an artificial light point j, with position $[o_{j1}^t o_{j2}^t \ldots o_{jn}^t s_{j,i_k}^t]$ in the joint search and objective space sitting in front of the function surface, we say aberration has occurred if the value of lateral aberration κ_{j,i_k}^t (see Eq. (12)) be greater than a given threshold. We assume that the threshold value is 0.01 (in optics, for a mirror with radius of curvature of 1 and a light ray of $\theta = 30^\circ$, HO comes out to be 0.26 and Eq. (3) yields the value of 0.017 for the lateral aberration (i.e., 1.76% aberration). Deviation of the reflected ray from focal point starts becoming noticeable around $\theta = 30^\circ$, [17]). Using Eq. (3) as the basis to calculate the value of lateral aberration relevant to an artificial mirror which is reflecting the image of the artificial light point j in iteration t, we can write:

$$\kappa_{j,i_k}^t = \frac{(r_{i_k}^t)^2}{2\sqrt{(r_{i_k}^t)^2 - (HO_{j,i_k}^t)^2}} - \frac{\left|r_{i_k}^t\right|}{2} \tag{12}$$

If for an artificial light point j we come out to $\kappa_{j,i_k}^t > 0.01$, we correct the occurred aberration via increasing the length of the artificial mirror radius of curvature. To correct the occurred aberration, we repeatedly do the following steps until getting $\kappa_{j,i_k}^t \leq 0.01$. We first increase the absolute length of the physical infinity and set $d_\infty \leftarrow 2d_\infty$. Then, the new value of the radius of curvature $r_{i_k}^t$ is calculated via Eq. (5), given that $m_{i_k}^t = m_{i_k}^t + d_\infty$ (if the artificial mirror is concave) or $m_{i_k}^t = m_{i_k}^t - d_\infty$ (if the artificial mirror is convex). The value of κ_{j,i_k}^t is then computed based on the new value of $r_{i_k}^t$ and the whole process is repeated until getting $\kappa_{j,i_k}^t < 0.01$ (Fig. 8 represents such an idea to correct the artificial aberration occurred in OIO). The final value of $r_{i_k}^t$ is then used to calculate q_{j,i_k}^t via Eq. (6).

Now we are ready to put forward the entire flowchart of basic OIO algorithm for minimizing an unconstrained numerical function (see Fig. 9).

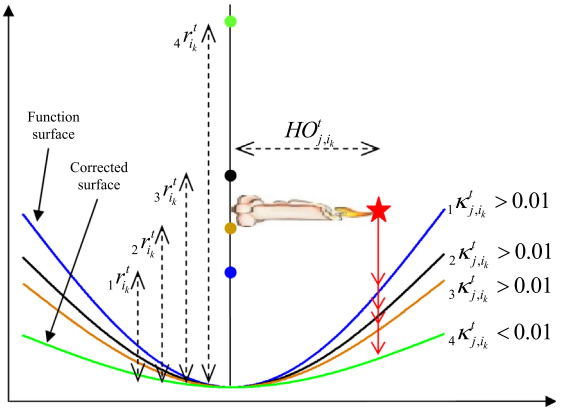

Fig. 8. The artificial spherical aberration is corrected via increasing the length of the radius of curvature.

4. Other variants of OIO algorithm

One can see that the search direction in OIO algorithm is determined based on employing the first law of reflection, which says: the incident ray, the reflected ray, and the normal all lie on the same plane. Due to this law, a new image position $\vec{I}^{t}_{j,i_k}$ in OIO is formed somewhere on the line connecting points $\vec{O}^{t}_{j}$ and $\vec{F}^{t}_{i_k}$ in the search space. In this section we relax this law artificially in such a way that the image can now rotate around the principal axis and has not to lie on the same line with the object (such a situation can be imagined for a 2d object/image). The consequence of this relaxation in OIO is that now, $\vec{I}^{t}_{j,i_k}$ is not forced to lie on the line connecting

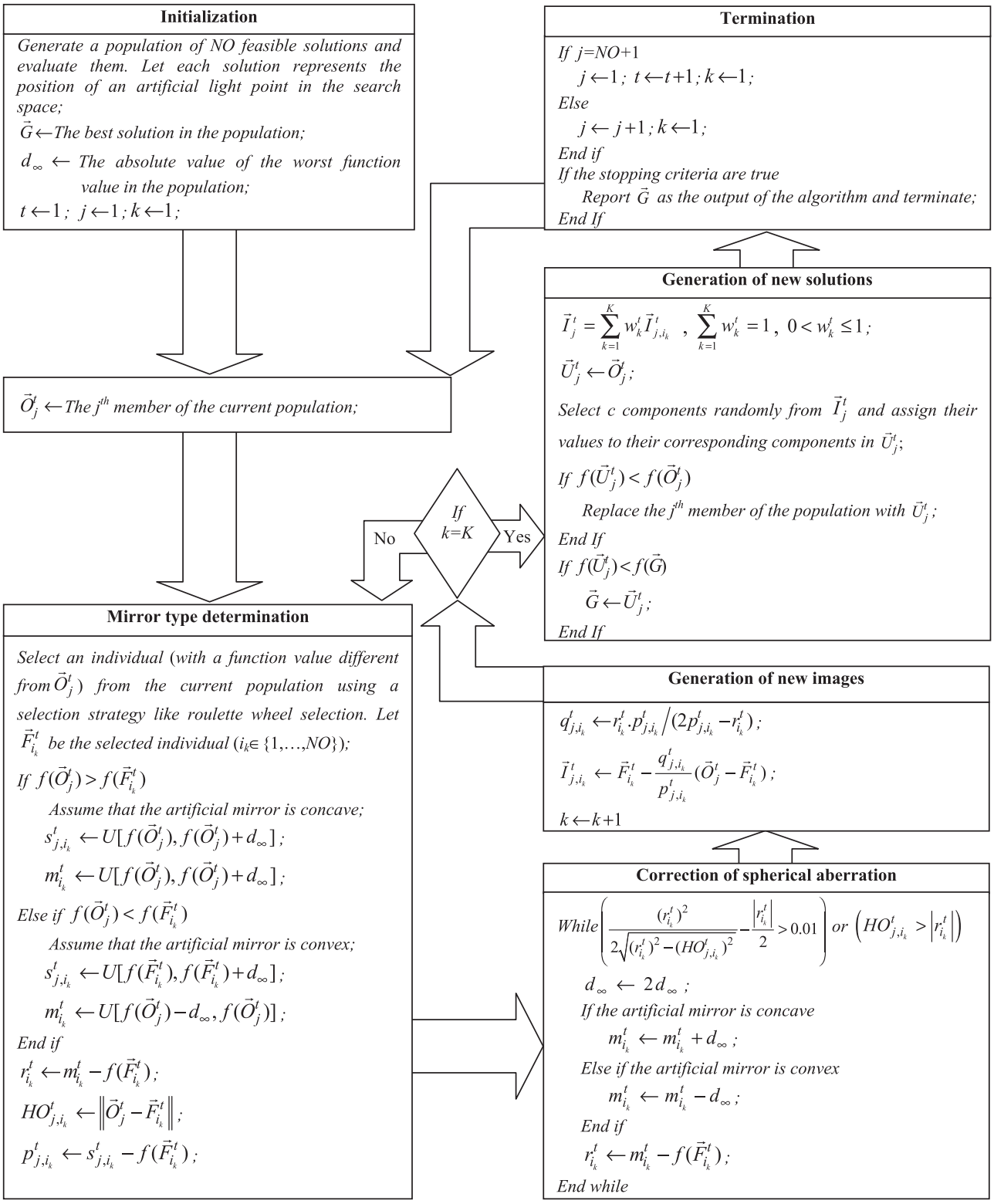

Fig. 9. Flowchart of basic OIO algorithm.

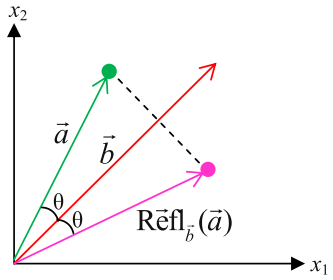

Fig. 10. (Color online) Vector reflection on x_1-x_2 plane.

points $\vec{O}_j^t$ and $\vec{F}_{i_k}^t$ in the search space. To model such a relaxation we need a technique for vector rotation. Among many possible rotations, we consider a special one which is based on the vector reflection defined in linear algebra [18]. Let us consider the geometry of Fig. 10 in which the point $\vec{a}$ (the green point) is reflected across the vector $\vec{b}$ (a mirror). Then, the reflection of $\vec{a}$ across $\vec{b}$ which is given by $\text{Refl}_{\vec{b}}(\vec{a})$ (the pink point), lies on the other side of $\vec{b}$ from $\vec{a}$, exactly at the same distance from $\vec{b}$ as is $\vec{a}$. The distance between $\vec{a}$ and its reflection is exactly twice the distance of $\vec{a}$ to $\vec{b}$.

Definition:. *Vector reflection*—The vector reflection of a vector $\vec{a}$ across the nonzero vector $\vec{b}$ in R^n is $\text{Refl}_{\vec{b}}(\vec{a})=2\frac{\vec{a}\cdot\vec{b}}{\vec{b}\cdot\vec{b}}\vec{b}-\vec{a}$, where $\vec{a}\cdot\vec{b}$ is the inner product of $\vec{a}$ and $\vec{b}$.

Abbreviated by ROIO (rotation based optics inspired optimization), we address a different version of OIO algorithm which uses the notion of image rotation to generate new solutions within the search space. Fig. 11 demonstrates the idea to generate a new rotated image position vector in ROIO and the one typically generated in OIO, for a two dimensional search space.

Since the image rotation is allowed in ROIO, we assume that the artificial image is formed in the search space on the line connecting $\text{Refl}_{\vec{O}_j^t}(\vec{F}_{i_k}^t)$ and $\vec{F}_{i_k}^t$ (see Fig. 11). The image position of the artificial light point j in the search space in iteration t is generated in ROIO as follows:

$$\vec{I}_{j,i_k}^t=\vec{F}_{i_k}^t-HO_{j,i_k}^t\frac{q_{j,i_k}^t}{p_{j,i_k}^t}\frac{\vec{F}_{i_k}^t-\text{REfl}_{\vec{O}_j^t}(\vec{F}_{i_k}^t)}{\|\vec{F}_{i_k}^t-\text{REfl}_{\vec{O}_j^t}(\vec{F}_{i_k}^t)\|}. \tag{13}$$

Let us assume that the image rotation is still allowed, as in ROIO, but now the image position of the artificial light point j in the search space ($\vec{I}_{j,i_k}^t$) can be formed somewhere between what is typically generated in OIO and what is generated in ROIO. We use the concept of convex combination to model this situation. Let us refer to an algorithm which uses such a model as COIO (Convex combination based optics inspired optimization). The image position of the artificial light point j in the search space in iteration t is thus generated in COIO by

$$\vec{I}_{j,i_k}^t=\vec{F}_{i_k}^t-HO_{j,i_k}^t\frac{q_{j,i_k}^t}{p_{j,i_k}^t}\left(\lambda_j^t\frac{\vec{O}_j^t-\vec{F}_{i_k}^t}{\|\vec{O}_j^t-\vec{F}_{i_k}^t\|}+(1-\lambda_j^t)\frac{\vec{F}_{i_k}^t-\text{REfl}_{\vec{O}_j^t}(\vec{F}_{i_k}^t)}{\|\vec{F}_{i_k}^t-\text{REfl}_{\vec{O}_j^t}(\vec{F}_{i_k}^t)\|}\right) \tag{14}$$

where $0\le\lambda_j^t\le 1$. The value of λ_j^t is set randomly as $U[0,1]$.

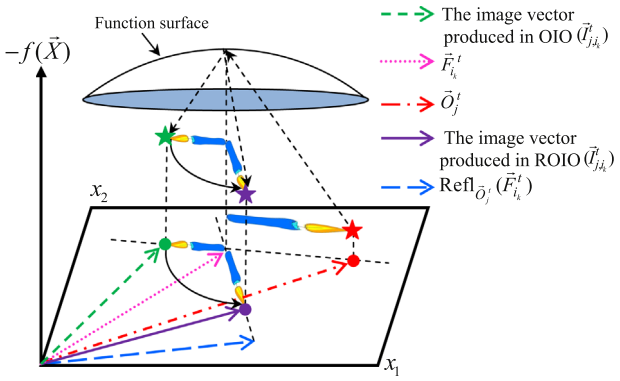

Fig. 11. Artificial image formation in a 2d search space by OIO and ROIO (for the sake of better visualization the vertical axis has been inverted).

The entire flowchart of ROIO and COIO is kept the same as that of OIO except for Eq. (9) which is replaced by Eq. (13) and Eq. (14), respectively.

5. Experiments on benchmark functions

In this section we investigate the performance of the proposed algorithms on various unimodal/multimodal, separable/non-separable and regular/non-regular functions, against different rivals. We consider 5 experiments for comparisons. The first experiment consists of 23 classic functions. This set of benchmark functions have been widely adopted by other researchers and we conduct such an experiment to compare algorithms in terms of their convergence speed. In the second experiment we compare the performance of algorithms in the presence of large scale optimizations on three numerical functions. The third experiment consists of 50 functions. This set is large enough to include many different kinds of problems. The aim of such an experiment is to measure the final quality of the outputs when the algorithms have enough opportunity to do their best. Experiment 4 aims at optimizing all benchmark functions of the special session on real-parameter optimization of the 2005 IEEE Congress on Evolutionary Computation (CEC2005). Finally in experiment 5 we investigate the performance of the new algorithms on a real world application of engineering design optimization which is the bi-objective optimization of a centrifuge pump. All of OIO, ROIO and COIO algorithms are implemented in MATLAB environment and their codes are available online at drkashan.ir.

5.1. Experiment 1

In our first experiment we compare the performance of OIO based algorithms with different evolutionary and swarm intelligence algorithms like evolution strategies learned with automatic termination (ESLAT), covariance matrix adaptation evolution strategies (CMA-ES), ABC and TLBO. In this experiment there are 23 benchmark functions which have been used in previous research of Karaboga and Akay [19], He et al. [8] and Rao and Patel [20]. These problems are identified as P1 to P23 and can be grouped into unimodal functions (P1 to P7), multimodal functions (P8 to P13), and low-dimensional multimodal functions (P14 to P23). Details on the benchmark functions considered in this experiment are shown in Table 2.

The maximum allowed number of evaluations is 100,000 for all algorithms. All OIO based algorithms are run for 50 times on each benchmark function. The algorithms terminate when they complete the maximum number of evaluations or when they reach the global minima within the gap of 0.001. Results obtained

Table 2
Benchmark functions used in Experiments 1.

Function		Formulation	*n*	Range	*Fmin*
P1	Sphere	$f(X)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]^n$	0
P2	Schwefel 2.22	$f(X)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]^n$	0
P3	Schwefel 1.2	$f(X)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^n$	0
P4	Schwefel 2.21	$f(X)=\max\{\lvert x_i\rvert, i=1,\ldots,n\}$	30	$[-100,100]^n$	0
P5	Rosenbrock	$f(X)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	30	$[-30,30]^n$	0
P6	Step	$f(X)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	30	$[-100,100]^n$	0
P7	Quartic	$f(X)=\sum_{i=1}^{n}ix_i^4+\text{random}[0,1)$	30	$[-1.28,1.28]^n$	0
P8	Schwefel 2.26	$f(X)=-\sum_{i=1}^{n}\left(x_i\ \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	30	$[-500,500]^n$	−12569.5
P9	Rastrigin	$f(X)=\sum_{i=1}^{n}(x_i^2-10\ \cos(2\pi x_i)+10)$	30	$[-5.12,5.12]^n$	0
P10	Ackley	$f(X)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos\ 2\pi x_i\right)+20+e$	30	$[-32,32]^n$	0
P11	Griewank	$f(X)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(x_i/\sqrt{i}\right)+1$	30	$[-600,600]^n$	0
P12	Penalized	$f(X)=\frac{\pi}{n}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$, $y_i=1+\frac{1}{4}(x_i+1)$, $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0 & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	30	$[-50,50]^n$	0
P13	Penalized 2	$f(X)=0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	$[-50,50]^n$	0
P14	Foxholes	$f(X)=\left[\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right]^{-1}$	2	$[-65.536,65.536]^n$	0.9980
P15	Kowalik	$f(X)=\sum_{i=1}^{11}\left(a_i-\frac{x_1(1+x_2b_i)}{(1+x_3b_i+x_4b_i^2)}\right)^2$	4	$[-5,5]^n$	0.0003075
P16	Six-hump camel-back	$f(X)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]^n$	−1.0316285
P17	Branin	$f(X)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(x_1)+10$	2	$[-5,10]\times[0,15]$	0.398
P18	Goldstein−Price	$f(X)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-2,2]^n$	3
P19	Hartman 3	$f(X)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{n}a_{ij}(x_j-p_{ij})^2\right]$	3	$[0,1]^n$	−3.86
P20	Hartman 6	$f(X)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{n}a_{ij}(x_j-p_{ij})^2\right]$	6	$[0,1]^n$	−3.32
P21	Shekel 5	$f(X)=-\sum_{j=1}^{5}\frac{1}{\sum_{i=1}^{4}(x_j-a_{ij})^2+c_i}$	4	$[0,10]^n$	−10.1499
P22	Shekel 7	$f(X)=-\sum_{j=1}^{7}\frac{1}{\sum_{i=1}^{4}(x_j-a_{ij})^2+c_i}$	4	$[0,10]^n$	−10.3999
P23	Shekel 10	$f(X)=-\sum_{j=1}^{10}\frac{1}{\sum_{i=1}^{4}(x_j-a_{ij})^2+c_i}$	4	$[0,10]^n$	−10.5319

by OIO based algorithms are compared with the results of other rivals in terms of the mean and standard deviation of the number of evaluations before termination (see Table 3), the rate of success in completing the search before reaching the maximum allowed number of evaluations (see Table 4) and the level of significance in performance (see Table 5). Results on the performance of comparator algorithms have been taken directly from Karaboga and Akay [19] and Rao and Patel [20].

To optimize the benchmark functions of Experiment 1, the number of artificial light points (*NO*) or the population size is set equal to 40 for OIO, ROIO and COIO algorithms. The first subpopulation in ROIO and COIO constitute one third of the whole population and the second subpopulation constitute two third of the whole population. In OIO, the size of each subpopulation is equal to half of the whole population. The number of changes (c) is set equal to 1 for the first subpopulation and is set equal to n for the second subpopulation. Results in terms of the mean and standard deviation of the number of function evaluations have been summarized in Table 3. Results show that COIO achieves the global minima (in all runs) of 12 out of 23 problems faster than ESLAT, CMA-ES, ABC and TLBO algorithms (see Table 5 for statistical significance, too). This record is 12 out of 23 for ROIO and 1 out of 23 for OIO. This means that both of ROIO and COIO exhibit the fastest convergence on half of test functions considered in Experiment 1. It is observed from the results of all algorithms that on 5 functions (P2, P3, P6, P9, and P11) the computational effort made by ROIO, to reach the global minima in all runs, is less than the rest of algorithms. This record is 7 (P1, P4, P5, P7, P8, P12, P13), 9 (P10, P14, P15, P17, P18, P19, P21, P22, P23), 2 (P16, P20), 0, 0 and 0, for COIO, TLBO, ABC, OIO, ESLAT and CMA-ES algorithms, respectively.

It is worth to note that the TLBO algorithm as presented in Rao et al. [9] and Rao and Patel [20] has some problems with correctly counting the number of fitness evaluations in duplicate removal phase [22]. Hence, the exact number of fitness evaluations in TLBO using publicly available code of Rao and Patel [20] is always only approximated and greater than those reported here which were adopted directly from Rao and Patel [20].

We abstain to report the best value found by algorithms, since they almost find the global optimum of all problems in all runs.

Table 3
Mean ± St.dev of the number of evaluations carried out by different algorithms for the functions considered in Experiment 1.

Function	ESLAT	CMA-ES	ABC	TLBO	OIO	ROIO	COIO
P1-Sphere	69724	10721	9264 ± 1481	4648 ± 148	26982 ± 952	2653 ± 407	2621 ± 402
P2-Schwefel 2.22	60859	12145	12991 ± 673	7395 ± 163	32212 ± 798	4921 ± 414	5221 ± 529
P3-Schwefel 1.2	72141	21248	12255 ± 1390	12218 ± 1305	1E+5 (6104.1) ± 0 (1895.2)	9037 ± 1593	9901 ± 1461
P4-Schwefel 2.21	69821	20813	1E+5 (2.712) ± 0 (1.18)	9563 ± 715	1E+5 (5.61) ± 0 (0.85)	4977 ± 506	4270 ± 478
P5-Rosenbrock	66609	55821	1E+5 (0.936) ± 0 (1.76)	1E+5 (16.321) ± 0 (1.356)	1E+5 (4.38) ± 0 (11.26)	26082 ± 15779	11950 ± 5375
P6-Step	57064	2184	4853 ± 1044	13778 ± 1491	15782 ± 1298	880 ± 207	1040 ± 192
P7-Quartic	50962	667131	1E+5 (0.0906) ± 0 (0.0189)	1E+5 (0.00625) ± 0 (0.00386)	1E+5 (0.029) ± 0 (0.012)	21439 ± 14010	19202 ± 10453
P8-Schwefel 2.26	61704	6621	64632 ± 23897	1E+5 ± 0	51102 ± 5863	63231 ± 12589	37778 ± 6027
P9-Rastrigin	53880	10079	26731 ± 9311	34317 ± 13866	50277 ± 5405	3869 ± 1005	4462 ± 909
P10-Ackley	58909	10654	16616 ± 1201	3868 ± 2634	36138 ± 1319	4670 ± 489	4905 ± 551
P11-Griewank	71044	10522	36151 ± 17128	10090 ± 16237	31773 ± 6713	3567 ± 491	3635 ± 453
P12-Penalized	63030	13981	73440 ± 2020	10815 ± 1430	22750 ± 1838	2666 ± 749	2264 ± 609
P13-Penalized 2	65655	13756	8454 ± 1719	30985 ± 12937	26700 ± 1301	5177 ± 1501	3884 ± 1775
P14-Foxholes	1305	540	1046 ± 637	524 ± 150	1199 ± 755	1531 ± 816	1359 ± 1064
P15-Kowalik	2869	13434	6120 ± 4564	2488 ± 2700	5944 ± 3382	5356 ± 3080	4856 ± 3031
P16-Six – Hump	1306	619	342 ± 109	447 ± 175	1083 ± 196	1199 ± 273	974 ± 274
P17-Branin	1257	594	530 ± 284	362 ± 88	1381 ± 523	1730 ± 498	1203 ± 358
P18-Goldstein – Price	1201	2052	15186 ± 13500	452 ± 244	796 ± 165	868 ± 220	647 ± 137
P19-Hartman 3	1734	996	4747 ± 16011	547 ± 135	987 ± 258	1291 ± 420	996 ± 274
P20-Hartman 6	3816	2293	1583 ± 457	24847 ± 29465	4104 ± 957	4513 ± 1130	4558 ± 1853
P21-Shekel 5	2338	1246	6069 ± 13477	1245 ± 114	4024 ± 2370	4043 ± 2026	3317 ± 2572
P22-Shekel 7	2468	1267	7173 ± 9022	1272 ± 99	5090 ± 4265	4077 ± 823	3062 ± 734
P23-Shekel 10	2410	1275	15392 ± 24413	1270 ± 135	4004 ± 969	3736 ± 759	3150 ± 1009

Table 4
Success rate of different algorithms for the benchmark functions considered in Experiment 1.

Function	ESLAT	CMA-ES	ABC	TLBO	OIO	ROIO	COIO
P1-Sphere	100	100	100	100	100	100	100
P2-Schwefel 2.22	100	100	100	100	100	100	100
P3-Schwefel 1.2	100	100	100	100	0	100	100
P4-Schwefel 2.21	0	100	0	100	0	100	100
P5-Rosenbrock	70	90	0	0	0	100	100
P6-Step	98	36	100	100	100	100	100
P7-Quartic	0	0	0	0	0	100	100
P8-Schwefel 2.26	0	0	86	40	100	100	100
P9-Rastrigin	40	0	100	100	100	100	100
P10-Ackley	100	100	100	100	100	100	100
P11-Griewank	90	92	96	100	100	100	100
P12-Penalized	100	88	100	100	100	100	100
P13-Penalized 2	60	86	100	100	100	100	100
P14-Foxholes	60	0	100	100	100	100	100
P15-Kowalik	94	88	100	100	100	100	100
P16-Six – Hump	100	100	100	100	100	100	100
P17-Branin	100	100	100	100	100	100	100
P18-Goldstein – Price	100	78	100	100	100	100	100
P19-Hartman 3	100	100	100	100	100	100	100
P20-Hartman 6	94	48	100	96	100	100	100
P21-Shekel 5	72	40	98	100	100	100	100
P22-Shekel 7	72	48	100	100	100	100	100
P23-Shekel 10	84	52	96	100	100	100	100
Total	**9**	**8**	**16**	**19**	**19**	**23**	**23**

The only exceptions are functions P3, P4, P5 and P7 for which OIO is not able to reach the global minimum in any run. For these problems, values in parenthesis denote the mean and standard deviation among the best function values found in 50 runs. As can be seen from these values, except for function P3 on which OIO performs unsatisfactory, on the other functions it performs acceptable.

The success rate of all algorithms is shown in Table 4. From results of this table we can see that COIO and ROIO perform surprisingly well and are able to find the global minima of all of 23 functions in all of 50 runs. Such a performance for example on a function such as P5, which is the Rosenbrock function, is valuable because the global minimum in this function is inside a long narrow valley and this makes the convergence towards the global minimum very difficult for many algorithms such as ABC or TLBO. The performance of OIO in terms of finding the global minima of all runs is also acceptable and comparable with TLBO. Both of OIO and TLBO find the global minima of 19 out of 23 functions in all of 50 runs.

The remarkable performance of ROIO and COIO algorithms is on function P7 which is known as Quartic function. All of ESLAT, CMA-ES, ABC and TLBO algorithms, together with OIO algorithm, fail to find the global optimum of this function. Quartic function is a simple unimodal function padded with noise. The noise component makes sure that the algorithm never gets the same value on the same point. The algorithm that does not perform well on this function will probably work poorly on noisy data. As can be seen, unlike all comparator algorithms, both of ROIO and COIO can hit the global minimum with few evaluations.

Another function on which none of the comparator algorithms could reach the global minimum in all runs is Schwefel function (P8) in which is the global minimum is geometrically distant, over the variable space, from the next best local minima. Therefore, the search algorithms are potentially prone to converge in the wrong direction. Fortunately, all of OIO, ROIO and COIO algorithms are able to find the true minimum in all runs. Among rivals, ABC exhibits the best performance with success rate of 86%. Both of ESLAT and CMA-ES are not able to hit the global minimum in any run, and TLBO is successful only in 40% of its runs. Here, the pure OIO algorithm performs much faster than others in reaching the global optimum in each run.

Adopting a similar statistical analysis followed by Zhang et al. [21] we conduct an approximate two-sample t-test between the mean number of function evaluations made by OIO, ROIO and COIO versus ABC and TLBO algorithms according to following statistics:

$$t_0 = \frac{\bar{y}_1 - \bar{y}_2}{\sqrt{S_1^2/n_1 + S_2^2/n_2}} \tag{15}$$

where $\bar{y}_1$ and $\bar{y}_2$are mean values and S_1 and S_2 denote the standard deviations of the results obtained by two different approaches. n_1 and n_2 are the number of independent runs of

two approaches. The degree of freedom is calculated as follows:

$$df = \left\lfloor 1/\left(\left(\left(\frac{S_1^2/n_1}{S_1^2/n_1+S_2^2/n_2}\right)^2/n_1\right)+\left(\left(\frac{S_2^2/n_2}{S_1^2/n_1+S_2^2/n_2}\right)^2/n_2\right)\right)\right\rfloor. \quad (16)$$

Results have been summarized in Table 5. For example, when comparing OIO versus ABC on problem P1, it is resulted by the t-test that the performance of ABC in terms of the mean number of evaluations is significantly different (better) at $\alpha=0.05$ from OIO. Therefore, the content of the relevant cell is filled by "ABC". From the results of Table 5 it can be seen that both of ROIO and COIO perform faster than others. However, comparisons indicate that COIO is faster than ROIO, since it performs faster than ROIO on 11 problems and performs slower only on 5 problems (see the last row in Table 5). On 7 problems there is no significant difference. OIO performs as the weakest, too.

It should be noted that the weak performance of OIO is due to the use of subpopulations. As we investigated, on 22 out of 23 functions, OIO performs much better with one population (where no subpopulation exists) under setting of $c=1$. Only on one function it performs very well under $c=n$. Bearing the force of using a same setting for all functions via the notion of subpopulations, is at the expense of a lower convergence rate.

Results of Table 5 demonstrate that both of COIO and ROIO perform faster than TLBO on 13 problems, but perform slower on 10 problems. The statistical difference between the convergence speed of both COIO and ROIO in comparison with ABC is even more significant. While COIO and ROIO converge faster than ABC on 15 and 16 problems respectively, each of them performs slower only on 4 functions. In general, the final ranking in terms of the performance and convergence speed of algorithms would be as COIO > ROIO > TLBO > OIO=ABC, (where " > " denotes "generally better/faster than" relationship and "=" denotes "almost equal" relationship).

In conclusion, from the results summarized over 23 benchmark functions we can see that both of ROIO and COIO are very efficient and provide the best results faster than others. However, COIO performs generally faster than ROIO.

5.2. Experiment 2

This experiment is conducted for very high dimension of $n=500$ for Rosenbrock (P5), Schwefel 2.26 (P8) and Penalized (P12) functions to measure the potentials of our algorithms for large scale optimization (we do not investigate the other functions considered by Rao et al. [9] since they are relatively easy). The search domain for the Rosenbrock function is modified as $[-100, 100]^n$. Results in terms of the mean values for 30 independent runs are compared with those obtained by PSO, DE, ABC, and TLBO algorithms and details are summarized in Table 6. Following Rao et al. [9], the maximum number of function evaluations is set to 100000. All parameter settings are similar to that were used in Experiment 1 except for the population size which is set equal to 9 for OIO, ROIO and COIO. It can be seen from the results of Table 6 that all rivals are inferior to ROIO and COIO on all functions. On the large scale Rosenbrock function, COIO performs as the best and converges truly to the global minimum in all runs. The performance of ROIO is also remarkable in comparison with rivals, but is inferior to that given by COIO. On 28 out of 30 runs, ROIO converges to the global minimum truly, but on two runs it converges to a local solution with function value equal to 493.958. The pure OIO algorithm is not able to

Table 5
Significance test for OIO, ROIO and COIO algorithms versus ABC and TLBO algorithms on the benchmark functions considered in Experiment 1

Function	OIO vs ABC	OIO vs TLBO	ROIO vs ABC	ROIO vs TLBO	COIO vs ABC	COIO vs TLBO	OIO vs ROIO	OIO vs COIO	ROIO vs COIO
P1-Sphere	ABC	TLBO	ROIO	ROIO	COIO	COIO	ROIO	COIO	–
P2-Schwefel 2.22	ABC	TLBO	ROIO	ROIO	COIO	COIO	ROIO	COIO	ROIO
P3-Schwefel 1.2	ABC	TLBO	ROIO	ROIO	COIO	COIO	ROIO	COIO	ROIO
P4-Schwefel 2.21	–	TLBO	ROIO	ROIO	COIO	COIO	ROIO	COIO	COIO
P5-Rosenbrock	–	–	ROIO	ROIO	COIO	COIO	ROIO	COIO	COIO
P6-Step	ABC	TLBO	ROIO	ROIO	COIO	COIO	ROIO	COIO	ROIO
P7-Quartic	–	–	ROIO	ROIO	COIO	COIO	ROIO	COIO	–
P8-Schwefel 2.26	OIO	OIO	–	ROIO	COIO	COIO	OIO	COIO	COIO
P9-Rastrigin	ABC	TLBO	ROIO	ROIO	COIO	COIO	ROIO	COIO	ROIO
P10-Ackley	ABC	TLBO	ROIO	TLBO	COIO	TLBO	ROIO	COIO	ROIO
P11-Griewank	OIO	TLBO	ROIO	ROIO	COIO	COIO	ROIO	COIO	–
P12-Penalized	OIO	TLBO	ROIO	ROIO	COIO	COIO	ROIO	COIO	COIO
P13-Penalized 2	ABC	OIO	ROIO	ROIO	COIO	COIO	ROIO	COIO	COIO
P14-Foxholes	–	TLBO	ABC	TLBO	ABC	TLBO	OIO	–	–
P15-Kowalik	–	TLBO	–	TLBO	–	TLBO	–	COIO	–
P16-Six – Hump	ABC	TLBO	ABC	TLBO	ABC	TLBO	OIO	COIO	COIO
P17-Branin	ABC	TLBO	ABC	TLBO	ABC	TLBO	OIO	COIO	COIO
P18-Goldstein – Price	OIO	TLBO	ROIO	TLBO	COIO	TLBO	OIO	COIO	COIO
P19-Hartman 3	–	TLBO	–	TLBO	–	TLBO	OIO	–	COIO
P20-Hartman 6	ABC	OIO	ABC	ROIO	ABC	COIO	OIO	–	–
P21-Shekel 5	–	TLBO	–	TLBO	–	TLBO	–	–	–
P22-Shekel 7	–	TLBO	ROIO	TLBO	COIO	TLBO	–	COIO	COIO
P23-Shekel 10	OIO	TLBO	ROIO	TLBO	COIO	TLBO	–	COIO	COIO
Total	**5 vs 10**	**3vs 18**	**15 vs 4**	**13 vs 10**	**16 vs 4**	**13 vs 10**	**7 vs 12**	**0 vs 19**	**5 vs 11**

Table 6
Comparative results on the mean performance of different algorithms for the benchmark functions considered in Experiment 2.

Function	PSO	DE	ABC	TLBO	OIO	ROIO	COIO
Rosenbrock	1.09E+6	8.72E+10	1007.87	497.91	403305.63	32.93	**2.38995E−08**
Scwefel 2.26	−98168.1	−138152.03	−190906.66	−184297.381	−183298.393	−203579.490	**−204013.866**
Penalized	5.29	1.48E+10	3.46E−8	0.06292	0.184650	2.81854E−13	**2.49533E−14**

Table 7
Benchmark functions considered in Experiments 3.

Function	Formulation	*n*	Range	*Fmin*
Q1-Stepint	$f(X)=25+\sum_{i=1}^{n}\lfloor x_i\rfloor$	5	$[-5.12,5.12]^n$	0
Q2-Step	$f(X)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	30	$[-100,100]^n$	0
Q3-Sphere	$f(X)=\sum_{i=1}^{n}x_i^2$	30	$[-100,100]^n$	0
Q4-Sum Squares	$f(X)=\sum_{i=1}^{n}ix_i^2$	30	$[-10,10]^n$	0
Q5-Quartic	$f(X)=\sum_{i=1}^{n}ix_i^4+\text{random}[0,1)$	30	$[-1.28,1.28]^n$	0
Q6-Beable	$f(X)=(1.5-x_1+x_1x_2)^2+(2.25-x_1+x_1x_2^2)^2+(2.625-x_1+x_1x_2^3)^2$	2	$[-4.5,4.5]^n$	0
Q7-Easom	$f(X)=-\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$	2	$[-100,100]^n$	−1
Q8-Matyas	$f(X)=0.26(x_1^2+x_2^2)-0.48x_1x_2$	2	$[-10,10]^n$	0
Q9-Colville	$f(X)=100(x_1^2-x_2)^2+(x_1-1)^2+(x_3-1)^2+90(x_3^2-x_4)^2+10.1((x_2-1)^2+(x_4-1)^2)+19.8(x_2-1)(x_4-1)$	4	$[-10,10]^n$	0
Q10-Trid6	$f(X)=\sum_{i=1}^{n}(x_i-1)^2-\sum_{i=2}^{n}x_ix_{i-1}$	6	$[-n^2,n^2]^n$	−50
Q11-Trid10	$f(X)=\sum_{i=1}^{n}(x_i-1)^2-\sum_{i=2}^{n}x_ix_{i-1}$	10	$[-n^2,n^2]^n$	−210
Q12-Zakharov	$f(X)=\sum_{i=1}^{n}x_i^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4$	10	$[-5,10]^n$	0
Q13-Powell	$f(X)=\sum_{i=1}^{n/k}(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-x_{4i-1})^4+10(x_{4i-3}-x_{4i})^4$	24	$[-4,5]^n$	0
Q14-Schwefel 2.22	$f(X)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	30	$[-10,10]^n$	0
Q15-Schwefel 1.2	$f(X)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^n$	0
Q16-Rosenbrock	$f(X)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	30	$[-30,30]^n$	0
Q17-Dixon – Price	$f(X)=(x_1-1)^2+\sum_{i=2}^{n}i(2x_i^2-x_{i-1})^2$	30	$[-10,10]^n$	0
Q18-Foxholes	$f(X)=\left(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6}\right)^{-1}$	2	$[-65.536,65.536]^n$	0.998
Q19-Branin	$f(X)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(x_1)+10$	2	$[-5,10]\times[0,15]$	0.3979
Q20-Bohachevsky1	$f(X)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)-0.4\cos(4\pi x_2)+0.7$	2	$[-100,100]^n$	0
Q21-Booth	$f(X)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	2	$[-10,10]^n$	0
Q22-Rastrigin	$f(X)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	30	$[-5.12,5.12]^n$	0
Q23-Schwefel 2.26	$f(X)=-\sum_{i=1}^{n}x_i\sin(\sqrt{\lvert x_i\rvert})$	30	$[-500,500]^n$	−12569.48
Q24-Michalewicz 2	$f(X)=-\sum_{i=1}^{n}\sin(x_i)(\sin(ix_i^2/\pi))^{2m}, m=10$	2	$[0,\pi]^n$	−1.8013
Q25-Michalewicz 5	$f(X)=-\sum_{i=1}^{n}\sin(x_i)(\sin(ix_i^2/\pi))^{2m}, m=10$	5	$[0,\pi]^n$	−4.6876
Q26-Michalewicz10	$f(X)=-\sum_{i=1}^{n}\sin(x_i)(\sin(ix_i^2/\pi))^{2m}, m=10$	10	$[0,\pi]^n$	−9.6601
Q27-Schaffer	$f(X)=0.5+\frac{\sin^2\left(\sqrt{x_1^2+x_2^2}\right)-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	2	$[-100,100]^n$	0
Q28-Six – Hump	$f(X)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	2	$[-5,5]^n$	−1.0316
Q29-Bohachevsky2	$f(X)=x_1^2+2x_2^2-0.3\cos(3\pi x_1)\cos(4\pi x_2)+0.3$	2	$[-100,100]^n$	0
Q30-Bohachevsky3	$f(X)=x_1^2+2x_2^2-0.3\cos(3\pi x_1+4\pi x_2)+0.3$	2	$[-100,100]^n$	0
Q31-Shubert	$f(X)=\left(\sum_{i=1}^{5}i\cos((i+1)x_1+i)\right)\left(\sum_{i=1}^{5}i\cos((i+1)x_2+i)\right)$	2	$[-10,10]^n$	−186.7309
Q32-Goldstein-Price	$f(X)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]\times[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$	2	$[-2,2]^n$	3
Q33-Kowalik	$f(X)=\sum_{i=1}^{11}\left(a_i-\frac{x_1(1+x_2b_i)}{(1+x_3b_i+x_4b_i^2)}\right)^2$	4	$[-5,5]^n$	0.0003075
Q34-Shekel 5	$f(X)=-\sum_{j=1}^{5}\frac{1}{\sum_{i=1}^{4}(x_j-a_{ij})^2+c_i}$	4	$[0,10]^n$	−10.1531
Q35-Shekel 7	$f(X)=-\sum_{j=1}^{7}\frac{1}{\sum_{i=1}^{4}(x_j-a_{ij})^2+c_i}$	4	$[0,10]^n$	−10.4029
Q36-Shekel 10	$f(X)=-\sum_{j=1}^{10}\frac{1}{\sum_{i=1}^{4}(x_j-a_{ij})^2+c_i}$	4	$[0,10]^n$	−10.5364
Q37-Perm	$f(X)=\sum_{k=1}^{n}\left[\sum_{i=1}^{n}(i^k+\beta)((x_i/i)^k-1)\right]^2$	4	$[-n,n]^n$	0
Q38-PowerSum	$f(X)=\sum_{k=1}^{n}\left[\left(\sum_{i=1}^{n}x_i^k\right)-b_k\right]^2$	4	$[0,n]^n$	0
Q39-Hartman 3	$f(X)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{n}a_{ij}(x_j-p_{ij})^2\right]$	3	$[0,1]^n$	−3.8627
Q40-Hartman 6	$f(X)=-\sum_{i=1}^{4}c_i\exp\left[-\sum_{j=1}^{n}a_{ij}(x_j-p_{ij})^2\right]$	6	$[0,1]^n$	−3.3219
Q41-Griewank	$f(X)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(x_i/\sqrt{i}\right)+1$	30	$[-600,600]^n$	0
Q42-Ackley	$f(X)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos 2\pi x_i\right)+20+e$	30	$[-32,32]^n$	0
Q43-Penalized	$f(X)=\frac{\pi}{n}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	30	$[-50,50]^n$	0

Table 7 (*continued*)

Function	Formulation	*n*	Range	*Fmin*
	$y_i = 1+\frac{1}{4}(x_i+1), \quad u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$			
Q44-Penalized 2	$f(X) = 0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{n-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})] + (x_n-1)^2[1+\sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	30	$[-50,50]^n$	0
Q45-Langerman 2	$f(X) = -\sum_{i=1}^{m} c_i \exp\left(\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$	2	$[0,10]^n$	−1.08
Q46-Langerman 5	$f(X) = -\sum_{i=1}^{m} c_i \exp\left(\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$	5	$[0,10]^n$	−1.5
Q47-Langerman10	$f(X) = -\sum_{i=1}^{m} c_i \exp\left(\left(-\frac{1}{\pi}\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\cos\left(\pi\sum_{j=1}^{n}(x_j-a_{ij})^2\right)\right)$	10	$[0,10]^n$	NA
Q48-FletcherPowell 2	$f(X) = \sum_{i=1}^{n}(A_i-B_i)^2, A_i = \sum_{j=1}^{n}(a_{ij}\ \sin\ \alpha_j + b_{ij}\ \cos\ \alpha_j)$ $B_i = \sum_{j=1}^{n}(a_{ij}\ \sin\ x_j + b_{ij}\ \cos\ x_j)$	2	$[-\pi,\pi]^n$	0
Q49-FletcherPowell 5	$f(X) = \sum_{i=1}^{n}(A_i-B_i)^2, A_i = \sum_{j=1}^{n}(a_{ij}\ \sin\ \alpha_j + b_{ij}\ \cos\ \alpha_j)$ $B_i = \sum_{j=1}^{n}(a_{ij}\ \sin\ x_j + b_{ij}\ \cos\ x_j)$	5	$[-\pi,\pi]^n$	0
Q50-FletcherPowell 10	$f(X) = \sum_{i=1}^{n}(A_i-B_i)^2, A_i = \sum_{j=1}^{n}(a_{ij}\ \sin\ \alpha_j + b_{ij}\ \cos\ \alpha_j)$ $B_i = \sum_{j=1}^{n}(a_{ij}\ \sin\ x_j + b_{ij}\ \cos\ x_j)$	10	$[-\pi,\pi]^n$	0

achieve satisfactory results on this function but it is still superior to PSO and DE. On the large scale Schwefel 2.26 function, ROIO and COIO perform very well and their mean output is close to the global optimum (with minimum value equal to −209491.44). They are able to repeatedly hit the global optimum. OIO performs better than PSO and DE and close to TLBO. On the large scale Penalized function (with minimum value equal to 0) ROIO and COIO perform remarkably better than others. OIO shows inferior result compared to that of ABC and TLBO, but still better than PSO and DE. The final ranking in terms of the quality of results provided by algorithms would be as COIO > ROIO > ABC > TLBO > OIO > PSO > DE (where " > " denotes the "generally better than" relationship).

5.3. Experiment 3

In this experiment we investigate the performance of OIO based algorithms on a large set of problems including 50 benchmark functions taken from Karaboga and Akay [19]. Details of the benchmark functions considered in this experiment are shown in Table 7 (for information on the value of parameters used for different functions, please refer to Karaboga and Akay [19]).

To conduct a fair comparison between different algorithms, all algorithms are run for 30 times on each benchmark function. The maximum allowed number of evaluations in each run is set as 500000. Results obtained by OIO, ROIO and COIO algorithms are compared with the results given by GA, PSO, DE, ABC and TLBO for the same number of function evaluations. Results reported here for these algorithms have been taken directly from Karaboga and Akay [19] and, Rao and Patel [20].

Again, all parameter settings are similar to that were used in Experiment 1 except for the population size which is set equal to 300 for OIO, ROIO and COIO. In order to make the comparisons more clearly, values below 10^{-12} are assumed to be 0. The mean and standard deviation of the best values are given in Table 8 for each function. Values in parenthesis determine the rank of algorithms. A smaller value denotes a better rank for the algorithm among others. The last two rows in Table 8 show the total points gathered by each algorithm (through summation over all rank values) and the number of times out of 50, that an algorithm is successful to find the global minimum in all of 30 runs, respectively. The best (minimum) possible points awarded to an algorithm would be equal to 50 and is attained when an algorithm takes the first rank for all of 50 functions.

From the results of Table 8 it can be seen that all of OIO, ROIO and COIO are the most effective algorithms among rivals and outperform GA, PSO, DE, ABC and TLBO algorithms on many functions.

On 44 out of 50 (i.e. 88%) benchmark functions COIO is ranked first. Such a record is 82% for ROIO, 78% for OIO, 76% for ABC, 72% for TLBO, 64% for DE, 48% for PSO and 32% for GA.

These conclusions sound that OIO based algorithms are among the most reliable and effective algorithms. On PowerSum function COIO performs the best and takes the first rank singly, though its difference with OIO and ROIO is not significant. ROIO shows the best performance on Quartic and Langerman 10 functions on which COIO performs a bit worse. On these problems the performance of ROIO and COIO is significantly better than other rivals. On FletcherPowel 10 OIO performs significantly better than other algorithms where some of them show a highly varying performance. On Rosenbrock function the performance of OIO is comparable to COIO and is best among other rivals. Here the performance of OIO and COIO is always optimal.

There are only three functions on which the first rank is not dedicated to one of OIO based algorithms. These functions are Trid 10, Dixon – Price and Perm. On Trid 10, OIO based algorithms are not able to converge to the global optimum with the desired precision. On Dixon – Price function, this is ABC algorithm which obtains the true optimum. Unlike PSO, DE and TLBO algorithms which always end up with local optima, all of OIO algorithms are able to escape from these optima. Here the performance of OIO and COIO is much better than ROIO. On Perm function, the performance of all OIO based algorithms are rather the same and better than all rivals except TLBO. On Langerman 5 all of OIO based algorithms together with DE perform optimal and converge to the global optimum acceptably in all runs. On FletcherPowel 5 function, the performance of OIO and COIO is always optimal. While the performance of some rivals is far distant from optimal, the performance of ROIO is acceptable. On FletcherPowel 10 function, OIO based algorithms takes the first three rank. Though the performance of ABC is not too bad, other rivals exhibit noncompetitive performance.

While on 7 functions (Quartic, Zakharov, Powell, Schwefel 2.22, Schwefel 1.2, Ackley and Langerman 10) ROIO beats OIO, on 9 functions (Trid 10, Rosenbrock, Dixon-Price, Schwefel 2.26,

Table 8
Comparative results of different algorithms over 30 independent runs on the functions considered in Experiments 3.

Function		GA	PSO	DE	ABC	TLBO	OIO	ROIO	COIO
Q1-Stepint	Mean	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0	0	0	0	0	0	0	0
Q2-Step	Mean	1.17E+3 (2)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	76.561450	0	0	0	0	0	0	0
Q3-Sphere	Mean	1.11E+3 (2)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	74.214474	0	0	0	0	0	0	0
Q4-Sum Squares	Mean	1.48E+2 (2)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	12.4092893	0	0	0	0	0	0	0
Q5-Quartic	Mean	0.1807 (8)	0.00115659(3)	0.0013633 (4)	0.0300166 (7)	0.0043519 (5)	0.0231883 (6)	4.8880E−05 (1)	6.0008E−05 (2)
	St.dev	0.027116	0.000276	0.000417	0.004866	1.99E−3	0.0076709	3.3277E−05	3.7710E−05
Q6-Beable	Mean	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0	0	0	0	0	0	0	0
Q7-Easom	Mean	−1 (1)	−1 (1)	−1 (1)	−1 (1)	−1 (1)	−1 (1)	−1 (1)	−1 (1)
	St.dev	0	0	0	0	0	0	0	0
Q8-Matyas	Mean	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0	0	0	0	0	0	0	0
Q9-Colville	Mean	0.014938 (2)	0 (1)	0.0409122 (3)	0.0929674 (4)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0.007364	0	0.081979	0.066277	0	0	0	0
Q10-Trid6	Mean	−49.9999 (1)	−50 (1)	−50 (1)	−50 (1)	−50 (1)	−50 (1)	−50 (1)	−50 (1)
	St.dev	2.25E−5	0	0	0	0	0	0	0
Q11-Trid 10	Mean	−209.476 (5)	−210 (1)	−210 (1)	−210 (1)	−210 (1)	−209.9818 (2)	−209.6001 (4)	−209.8299 (3)
	St.dev	0.193417	0	0	0	0	0.0368	0.4589	0.2553
Q12-Zakharov	Mean	0.013355 (4)	0 (1)	0 (1)	0.0002476 (2)	0 (1)	0.0016747 (3)	0 (1)	0 (1)
	St.dev	0.004532	0	0	0.000183	0	0.007782	0	0
Q13-Powell	Mean	9.703771 (7)	0.00011004 (4)	2.17E−7 (3)	0.0031344 (6)	5.86E−8 (2)	0.0029349 (5)	0 (1)	0 (1)
	St.dev	1.547983	0.000160	1.36E−7	0.000503	8.13E−8	0.001609	0	0
Q14-Schwefel 2.22	Mean	11.0214 (3)	0 (1)	0 (1)	0 (1)	0 (1)	1.17538E−09 (2)	0 (1)	0 (1)
	St.dev	1.386856	0	0	0	0	2.32123E−10	0	0
Q15-Schwefel 1.2	Mean	7.40E+3 (3)	0 (1)	0 (1)	0 (1)	0 (1)	889.9567 (2)	0 (1)	0 (1)
	St.dev	1.14E+3	0	0	0	0	271.2836	0	0
Q16-Rosenbrock	Mean	1.96E+5 (7)	15.088617 (5)	18.203938 (6)	0.0887707 (4)	1.62 E−5 (3)	0 (1)	1.26989E−08 (2)	0 (1)
	St.dev	3.85E+4	24.170196	5.036187	0.077390	3.64 E−5	0	3.40519E−08	0
Q17-Dixon−Price	Mean	1.22E+3 (6)	0.66666667 (5)	0.6666667 (5)	0 (1)	0. 6666667(4)	0.05645285 (3)	0.24473359 (4)	0.00980867 (2)
	St.dev	2.66E+2	1E−8	1E−9	0	0	0.03045534	0.00235027	0.00495085
Q18-Foxholes	Mean	0.998004 (1)	0.99800393 (1)	0.9980039 (1)	0.9980039 (1)	0.9980039 (1)	0.9980039 (1)	0.9980039 (1)	0.9980039 (1)
	St.dev	0	0	0	0	0	0	0	0
Q19-Branin	Mean	0.397887 (1)	0.39788736 (1)	0.3978874 (1)	0.3978874 (1)	0.3978874 (1)	0.3978874 (1)	0.3978874 (1)	0.3978874 (1)
	St.dev	0	0	0	0	0	0	0	0
Q20-Bohachevsky1	Mean	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0	0	0	0	0	0	0	0
Q21-Booth	Mean	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0	0	0	0	0	0	0	0
Q22-Rastrigin	Mean	52.92259 (4)	43.9771369 (3)	11.716728 (2)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	4.564860	11.728676	2.538172	0	0	0	0	0
Q23-Schwefel 2.26	Mean	−11593.4 (4)	−6909.1359 (6)	−10266 (5)	−12569.487 (1)	−12414.884 (3)	−12569.487 (1)	−12569.483 (2)	−12569.487 (1)
	St.dev	93.254240	457.957783	521.849292	0	1.18E+2	4.9528E−12	0.0031058	1.6548E−12
Q24-Michalewicz 2	Mean	−1.8013 (1)	−1.5728692 (2)	−1.801303 (1)	−1.8013034 (1)	−1.801303 (1)	−1.8013034 (1)	−1.8013034 (1)	−1.8013034 (1)
	St.dev	0	0.119860	0	0	0	0	0	0
Q25-Michalewicz 5	Mean	−4.64483 (4)	−2.4908728 (5)	−4.683482 (2)	−4.6876582 (1)	−4.672658 (3)	−4.6876582 (1)	−4.6876582 (1)	−4.6876582 (1)
	St.dev	0.097850	0.256952	0.012529	0	4.74E−2	0	0	0
Q26-Michalewicz 10	Mean	−9.49683 (4)	−4.0071803 (5)	−9.591151 (3)	−9.6601517 (1)	−9.6172 (2)	−9.6601517 (1)	−9.6601517 (1)	−9.6601517 (1)
	St.dev	0.141116	0.502628	0.064205	0	4.52E−2	2.28542E−09	2.28542E−09	0
Q27-Schaffer	Mean	0.004239 (2)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0.004763	0	0	0	0	0	0	0
Q28-Six−Hump	Mean	−1.03163 (1)	−1.0316285 (1)	−1.031628 (1)	−1.0316285 (1)	−1.03163 (1)	−1.0316285 (1)	−1.0316285 (1)	−1.0316285 (1)

	St.dev	0	0	0	0	0	0	0	0
Q29-Bohachevsky 2	Mean	0.06829 (2)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0.078216	0	0	0	0	0	0	0
Q30-Bohachevsky 3	Mean	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0	0	0	0	0	0	0	0
Q31-Shubert	Mean	−186.731 (1)	−186.73091 (1)	−186.7309 (1)	−186.73091 (1)	−186.731 (1)	−186.73091 (1)	−186.73091 (1)	−186.73091 (1)
	St.dev	0	0	0	0	0	0	0	0
Q32-Goldstein-Price	Mean	5.250611 (2)	3 (1)	3 (1)	3 (1)	3 (1)	3 (1)	3 (1)	3 (1)
	St.dev	5.870093	0	0	0	0	0	0	0
Q33-Kowalik	Mean	0.005615 (5)	0.00049062 (4)	0.0004266 (2)	0.0004266 (2)	0.0003076 (1)	0.000307486 (1)	0.000468939 (3)	0.000307486 (1)
	St.dev	0.008171	0.000366	0.000273	6.04E−5	0	0	0.000105866	3.0418E−09
Q34-Shekel 5	Mean	−5.66052 (2)	−2.0870079 (3)	−10.1532 (1)	−10.1532 (1)	−10.1532 (1)	−10.1532 (1)	−10.1532 (1)	−10.1532 (1)
	St.dev	3.866737	1.178460	0	0	0	0	0	0
Q35-Shekel 7	Mean	−5.34409 (2)	−1.9898713 (3)	−10.40294 (1)	−10.402941 (1)	−10.402941 (1)	−10.402941 (1)	−10.402941 (1)	−10.402941 (1)
	St.dev	3.517134	1.420602	0	0	0	0	0	0
Q36-Shekel 10	Mean	−3.82984 (2)	−1.8796753 (3)	−10.53641 (1)	−10.53641 (1)	−10.53641 (1)	−10.53641 (1)	−10.53641 (1)	−10.53641 (1)
	St.dev	2.451956	0.432476	0	0	0	0	0	0
Q37-Perm	Mean	0.302671 (8)	0.03605158 (6)	0.0240069 (5)	0.0411052 (7)	6.77E−4 (1)	0.0021156 (2)	0.0025935 (3)	0.0096613 (4)
	St.dev	0.193254	0.048927	0.046032	0.023056	7.45E−4	0.0023670	0.0021013	0.0290010
Q38-PowerSum	Mean	0.010405 (7)	11.3904479 (8)	0.0001425 (5)	0.0029468 (6)	7.43E−5 (4)	7.37137E−06 (2)	2.72054E−05 (3)	2.8563E−06 (1)
	St.dev	0.009077	7.355800	0.000145	0.002289	1.11E−4	1.03282E−05	3.68986E−05	1.4924E−05
Q39-Hartman 3	Mean	−3.86278 (1)	−3.6333523 (2)	−3.862782 (1)	−3.8627821 (1)	−3.862782 (1)	−3.8627821 (1)	−3.8627821 (1)	−3.8627821 (1)
	St.dev	0	0.116937	0	0	0	0	0	0
Q40-Hartman 6	Mean	−3.29822 (2)	−1.8591298 (4)	−3.226881 (3)	−3.3219952 (1)	−3.322368 (1)	−3.322368 (1)	−3.322368 (1)	−3.322368 (1)
	St.dev	0.050130	0.439958	0.047557	0	0	0	0	0
Q41-Griewank	Mean	10.63346 (4)	0.01739118 (3)	0.0014792 (2)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	1.161455	0.020808	0.002958	0	0	0	0	0
Q42-Ackley	Mean	14.67178 (4)	0.16462236 (3)	0 (1)	0 (1)	0 (1)	4.33560E−09 (2)	0 (1)	0 (1)
	St.dev	0.178141	0.493867	0	0	0	1.26948E−09	0	0
Q43-Penalized	Mean	13.3772 (4)	0.0207338 (3)	0 (1)	0 (1)	2.67 E−8 (2)	0 (1)	0 (1)	0 (1)
	St.dev	1.448726	0.041468	0	0	0	0	0	0
Q44-Penalized 2	Mean	125.0613 (5)	0.00767535 (4)	0.0021975 (3)	0 (1)	2.34 E−8 (2)	0 (1)	0 (1)	0 (1)
	St.dev	12.001204	0.016288	0.004395	0	0	0	0	0
Q45-Langerman 2	Mean	−1.08094 (1)	−0.679268 (2)	−1.080938 (1)	−1.0809384 (1)	−1.080938 (1)	−1.0809384 (1)	−1.0809384 (1)	−1.0809384 (1)
	St.dev	0	0.274621	0	0	0	0	0	0
Q46-Langerman 5	Mean	−0.96842 (2)	−0.5048579 (5)	−1.499999 (1)	−0.938150 (4)	−0.939702 (3)	−1.499999 (1)	−1.499999 (1)	−1.499999 (1)
	St.dev	0.287548	0.213626	0	0.000208	1.55E−5	4.6777E−08	4.0704E−08	4.7624E−08
Q47-Langerman 10	Mean	−0.63644 (6)	−0.0025656 (8)	−1.0528 (2)	−0.4460925(7)	−0.64906 (5)	−0.9182184 (3)	−1.0629210 (1)	−0.8456012 (4)
	St.dev	0.374682	0.003523	0.302257	0.133958	1.73E−1	0.2653010	0.3373867	0.1636252
Q48-FletcherPowell 2	Mean	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)	0 (1)
	St.dev	0	0	0	0	0	0	0	0
Q49-FletcherPowell 5	Mean	0.004303 (3)	1457.88344 (7)	5.988783 (6)	0.1735495 (4)	2.2038134 (5)	0 (1)	3.86905E−05 (2)	0 (1)
	St.dev	0.009469	1269.362389	7.334731	0.068175	4.39	0	0.0001104	0
Q50-FletcherPowell10	Mean	29.57348 (5)	1364.45555 (8)	781.55028 (7)	8.2334401 (4)	35.971004 (6)	0.8778067 (1)	5.1605302 (3)	3.6795318 (2)
	St.dev	16.021078	1325.379655	1048.813487	8.092742	7.13E+1	1.5771227	4.9103663	4.4047507
Point (final Rank)		**150 (8)**	**138 (7)**	**100 (6)**	**95 (5)**	**85 (4)**	**71 (3)**	**67 (2)**	**61 (1)**
# of first ranks out of 50		**16 (8)**	**24 (7)**	**32 (6)**	**38 (4)**	**36 (5)**	**39 (3)**	**41 (2)**	**44 (1)**

Kowalik, Perm, PowerSum, FletcherPowell 5 and FletcherPowell 10) OIO beats ROIO. On 8 functions (Quartic, Zakharov, Dixon–Price, Powell, Schwefel 2.22, Schwefel 1.2, PowerSum and Ackley) COIO beats OIO, but on 4 functions OIO beats COIO (Trid 10, Perm, Langerman 10 and FletcherPowell 10). On 3 functions (Quartic, Perm and Langerman 10) ROIO beats COIO, but on 8 functions (Trid 10, Rosenbrock, Dixon-Price, Schwefel 2.26, Kowalik, PowerSum, FletcherPowell 5 and FletcherPowell 10) COIO beats ROIO.

While both of ABC and TLBO algorithms are inferior to OIO, ROIO and COIO in terms of the criteria listed in the last two rows of Table 8, there is no surpassing between ABC and TLBO. Because the total point gathered by TLBO is less (better) than ABC, while the number of times that ABC takes the first rank is greater (better) than TLBO. The final ranking in terms of the quality of results provided by the algorithms would be as COIO > ROIO = OIO > TLBO = ABC > DE > PSO > GA (where ">" denotes "generally better than" relationship and "=" denotes "almost equal" relationship).

5.4. Experiment 4: Evaluation on CEC benchmark functions

In this section we compare the performance of OIO and COIO algorithms on all benchmark functions of the special session on real-parameter optimization of the 2005 IEEE Congress on Evolutionary Computation (CEC2005) [25]. These functions are introduced by F1 to F25. Five of these functions are unimodal and other twenty are multimodal functions. Experiments are conducted on all 25 functions in 10D and 30D. The replication length is set to 25 runs for each algorithm.

We abstain to evaluate the performance of ROIO because in our previous experiments, it performed almost similar to COIO which its performance is testing in this experiment. Since we evaluate the performance of OIO and COIO on CEC2005 benchmark functions, we can made an evaluation on the performance of ROIO, too.

To optimize CEC2005 functions, the number of artificial light points is set equal to 75 for 10D and 225 for 30D problems. The first subpopulation constitute one third of the whole population and the second subpopulation constitute two third of the whole population. The number of changes (c) is set equal to 1 for the first subpopulation and is set equal to $n-1$ for the second subpopulation.

To make the performance of OIO and COIO more competitive on a few benchmark functions, we have adopted the transfer operator of the League Championship Algorithm [12] by which, after a certain number of iterations (at the end of every *NO* number of successive iterations), a number of dimensions (i.e., 10% on average) are selected randomly from each artificial light point j ($\vec{O}_j^t$) and their relevant value is replaced with the corresponding value taken from another artificial light point m ($\vec{O}_m^t$) where m is selected randomly from the set of indices $M_j = \{m | m \neq j, f(\vec{O}_m^t) < f(\vec{O}_j^t)\}$. It is worth to note that on many functions the performance of OIO without using the transfer operator is the same as or even slightly better than the case of hybridizing it with this operator. However, there are several functions for which the performance of OIO and COIO with transfer operator (even with a small transfer probability of 10%) is better than the pure algorithm.

Best functions error values achieved when FEs = lE+3, FEs = lE+4 and FEs = lE+5 for the 10D functions are listed in Tables 9–12. Best functions error values achieved when FEs = lE+3, FEs = lE+4, FEs = lE+5 and FEs = 3E+5 for the 30D functions are listed in Tables 13–16. Comparisons versus an advance DMS-L-PSO algorithm are made and results are provided in Table 17. In their DMS-L-PSO algorithm, Liang and Suganthan [26] used Quasi-Newton method in the end of the search by PSO. Following Liang and Suganthan [26], we also allow in the end of the search (when 0.95*Max_FEs proceed) by OIO

Table 9
Results obtained by OIO and COIO on CEC2005 benchmark functions (F1–F7) with $n=10$.

FES		F1		F2		F3		F4		F5		F6		F7	
		OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO
1E+3	Best	3.0899E+02	5.3103E+01	2.2170E+03	1.3139E+03	1.8846E+06	2.9772E+06	3.1655E+03	5.4536E+02	8.2146E+03	6.0825E+03	8.3008E+04	8.4571E+04	1.0911E+01	9.1341E+00
	Median	1.5843E+03	7.9691E+02	4.2128E+03	6.2835E+03	1.5553E+07	1.6875E+07	5.1057E+03	6.0227E+03	1.1003E+04	9.2564E+03	2.8729E+06	5.0377E+06	1.2862E+02	5.0850E+01
	Worst	3.1925E+03	1.9254E+03	9.2456E+03	8.2526E+03	5.9355E+07	6.0357E+07	1.1851E+04	1.2984E+04	1.3625E+04	1.2226E+04	4.3204E+07	1.4964E+08	1.0371E+03	2.8781E+02
	Mean	1.5962E+03	8.3656E+02	5.0624E+03	5.7135E+03	1.8970E+07	2.0442E+07	6.1542E+03	6.3671E+03	1.0828E+04	9.3927E+03	6.8720E+06	1.5359E+07	2.7132E+02	7.2095E+01
	St.dev	7.3900E+02	5.5155E+02	1.9467E+03	1.8411E+03	1.4041E+07	1.6416E+07	2.6869E+03	3.0629E+03	1.3449E+03	1.4526E+03	9.9618E+06	3.1970E+07	2.6840E+02	6.2947E+01
1E+4	Best	7.5061E−06	1.0022E−04	2.6291E−01	7.4170E−01	2.1553E+04	7.3226E+04	3.8964E+00	6.8435E+00	1.2874E+02	5.9516E+02	3.8142E+00	4.1481E+00	6.2621E−02	1.3389E−01
	Median	1.2086E−04	8.0558E−04	4.8217E+00	3.0422E+01	1.7848E+05	3.5946E+05	2.9845E+01	1.4272E+02	7.0601E+02	1.1958E+03	1.0512E+01	6.0139E+01	1.0633E+00	1.1354E+00
	Worst	1.1360E−03	4.6095E−03	4.0261E+01	4.9616E+02	1.9574E+06	1.9167E+06	6.3820E+02	2.1993E+03	2.3579E+03	2.9910E+03	4.7949E+02	3.2264E+02	5.3011E+00	5.4788E+00
	Mean	2.2773E−04	1.0545E−03	8.9022E+00	7.5029E+01	3.4824E+05	5.2489E+05	8.1272E+01	3.8877E+02	7.7234E+02	1.4187E+03	6.5330E+01	8.3079E+01	1.4213E+00	1.4339E+00
	St.dev	2.9620E−04	9.5590E−04	1.0350E+01	1.1300E+02	4.9857E+05	4.3841E+05	1.4307E+02	5.3685E+02	4.7598E+02	6.9966E+02	1.0905E+02	9.1136E+01	1.3381E+00	1.1411E+00
1E+5	Best	0.0000E+00	0.0000E+00	0.0000E+00	4.5475E−13	2.6567E−04	3.0263E−04	0.0000E+00	1.0475E−09	7.2323E−08	2.2375E−02	2.8082E−09	2.9269E−09	2.9518E−02	1.1358E−08
	Median	0.0000E+00	0.0000E+00	0.0000E+00	6.6564E−11	4.4354E−04	4.3801E−04	0.0000E+00	1.4326E−06	5.7462E−06	3.3382E+00	4.5703E−09	4.4992E−09	7.1376E−02	1.6985E−01
	Worst	0.0000E+00	0.0000E+00	0.0000E+00	1.4259E−08	1.7951E+01	7.9888E+01	5.9040E−09	1.9549E−03	1.1910E−04	1.3917E+02	2.8950E−05	5.6626E−09	1.7710E−01	4.0884E−01
	Mean	0.0000E+00	0.0000E+00	0.0000E+00	1.0603E−09	1.2365E+00	6.2145E+00	2.8462E−10	1.0798E−04	1.4801E−05	1.3554E+01	1.6738E−06	4.3841E−09	8.2112E−02	1.9422E−01
	St.dev	0.0000E+00	0.0000E+00	0.0000E+00	3.0502E−09	3.6749E+00	1.7393E+01	1.1932E−09	3.9186E−04	2.6005E−05	2.9080E+01	6.2281E−06	5.4563E−10	4.5783E−02	1.2914E−01

Table 10
Results obtained by OIO and COIO on CEC2005 benchmark functions (F8–F13) with $n=10$.

FES		F8		F9		F10		F11		F12		F13	
		OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO
	Best	2.0339E+01	2.0440E+01	3.3112E+01	3.3858E+01	4.2325E+01	4.0347E+01	8.8975E+00	1.0149E+01	1.7011E+03	4.2085E+03	4.8938E+00	3.7655E+00
	Median	2.0758E+01	2.0724E+01	5.9664E+01	6.0749E+01	6.7523E+01	6.8219E+01	1.0964E+01	1.1402E+01	1.5343E+04	1.9614E+04	9.0805E+00	6.1380E+00
1E+3	Worst	2.0940E+01	2.0893E+01	7.2630E+01	7.5304E+01	8.7732E+01	8.2973E+01	1.2241E+01	1.2322E+01	3.1288E+04	4.8021E+04	4.0680E+01	1.8722E+01
	Mean	2.0745E+01	2.0718E+01	5.7314E+01	5.8907E+01	6.6176E+01	6.6997E+01	1.0832E+01	1.1321E+01	1.5781E+04	1.9364E+04	1.1680E+01	6.5587E+00
	St.dev	1.4739E−01	1.1859E−01	9.4439E+00	9.0168E+00	9.6750E+00	1.0290E+01	8.5669E−01	5.8308E−01	8.4629E+03	1.0580E+04	8.7327E+00	3.0759E+00
	Best	2.0334E+01	2.0357E+01	1.6724E+00	4.5345E−01	3.5926E+00	5.9779E+00	4.1445E+00	2.6342E+00	3.5082E+00	1.9751E+01	8.9794E−01	2.3314E−01
	Median	2.0504E+01	2.0578E+01	3.7106E+00	2.4361E+00	1.5289E+01	1.5095E+01	7.6252E+00	5.8393E+00	6.2030E+01	2.1304E+02	1.5257E+00	9.3826E−01
1E+4	Worst	2.0683E+01	2.0764E+01	6.1368E+00	3.9399E+00	3.4660E+01	2.7926E+01	8.9926E+00	9.8121E+00	3.2927E+03	3.2873E+03	2.4106E+00	1.4922E+00
	Mean	2.0511E+01	2.0542E+01	3.7936E+00	2.2989E+00	1.7474E+01	1.4846E+01	7.4694E+00	6.1163E+00	4.4783E+02	6.6533E+02	1.4832E+00	9.3234E−01
	St.dev	9.8949E−02	9.9622E−02	1.2637E+00	1.0075E+00	8.6849E+00	6.6949E+00	1.2427E+00	2.2253E+00	8.5056E+02	9.1229E+02	3.5764E−01	3.1470E−01
	Best	2.0000E+01	2.0000E+01	0.0000E+00	0.0000E+00	2.9849E+00	1.9899E+00	7.1017E−01	1.0622E+00	2.4158E−10	1.6223E−10	3.9623E−02	0.0000E+00
	Median	2.0000E+01	2.0000E+01	0.0000E+00	0.0000E+00	5.9698E+00	6.9647E+00	2.0515E+00	3.6712E+00	1.0003E+01	1.0003E+01	2.6524E−01	9.0406E−02
1E+5	Worst	2.0000E+01	2.0000E+01	0.0000E+00	0.0000E+00	8.9546E+00	1.2934E+01	3.7138E+00	5.5483E+00	1.8835E+01	2.0949E+01	4.2755E−01	2.8063E−01
	Mean	2.0000E+01	2.0000E+01	0.0000E+00	0.0000E+00	5.6912E+00	6.6065E+00	2.0814E+00	3.6384E+00	7.7213E+00	1.1266E+01	2.4798E−01	1.0884E−01
	St.dev	2.5658E−06	7.7677E−06	0.0000E+00	0.0000E+00	1.7593E+00	2.7078E+00	8.4384E−01	1.0212E+00	7.7419E+00	7.0834E+00	1.0938E−01	8.4099E−02

Table 11
Results obtained by OIO and COIO on CEC2005 benchmark functions (F14–F19) with $n=10$.

FES		F14		F15		F16		F17		F18		F19	
		OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO
	Best	3.5987E+00	4.1466E+00	4.6644E+02	5.4574E+02	2.2688E+02	1.9209E+02	2.5225E+02	2.3673E+02	1.0984E+03	1.0879E+03	1.0283E+03	1.0903E+03
	Median	4.1976E+00	4.3425E+00	6.6668E+02	6.6268E+02	2.8279E+02	2.6880E+02	3.1557E+02	3.1145E+02	1.1535E+03	1.1379E+03	1.1473E+03	1.1399E+03
1E+3	Worst	4.4861E+00	4.4359E+00	7.5086E+02	7.2207E+02	3.3574E+02	4.1265E+02	4.0987E+02	3.8154E+02	1.2218E+03	1.1742E+03	1.2041E+03	1.1916E+03
	Mean	4.1786E+00	4.3149E+00	6.5720E+02	6.5793E+02	2.8549E+02	2.6769E+02	3.1484E+02	3.0760E+02	1.1548E+03	1.1317E+03	1.1419E+03	1.1392E+03
	St.dev	2.0582E−01	9.3818E−02	6.5826E+01	4.3247E+01	3.1659E+01	4.8809E+01	3.4825E+01	3.6257E+01	3.4411E+01	2.3744E+01	3.9572E+01	2.8566E+01
	Best	3.3563E+00	2.7415E+00	6.3718E+01	6.9084E+01	9.5038E+01	8.7780E+01	9.5918E+01	1.0117E+02	4.9176E+02	5.9974E+02	5.2608E+02	5.4579E+02
	Median	3.7197E+00	3.6696E+00	1.3811E+02	1.6752E+02	1.1248E+02	1.2096E+02	1.3019E+02	1.2472E+02	9.4692E+02	9.6962E+02	8.1325E+02	9.7766E+02
1E+4	Worst	3.8928E+00	3.9049E+00	2.7867E+02	2.9281E+02	1.5277E+02	1.4981E+02	2.0771E+02	1.6133E+02	1.0247E+03	1.0132E+03	1.0230E+03	1.0447E+03
	Mean	3.6795E+00	3.6329E+00	1.3975E+02	1.7392E+02	1.1571E+02	1.1812E+02	1.3807E+02	1.2772E+02	8.9421E+02	9.3350E+02	8.7124E+02	9.4811E+02
	St.dev	1.5526E−01	2.5947E−01	6.0763E+01	5.2367E+01	1.4511E+01	1.3873E+01	2.8236E+01	1.6754E+01	1.3580E+02	9.5141E+01	1.1551E+02	1.0856E+02
	Best	2.6221E−01	1.1875E+00	0.0000E+00	0.0000E+00	7.7705E+01	8.7540E+01	9.1439E+01	9.6944E+01	3.0000E+02	3.0000E+02	3.0000E+02	4.0956E+02
	Median	2.1456E+00	2.4123E+00	0.0000E+00	0.0000E+00	1.0212E+02	1.0965E+02	1.0838E+02	1.1147E+02	8.0000E+02	8.0363E+02	8.0000E+02	8.1066E+02
1E+5	Worst	2.7097E+00	3.4902E+00	4.0075E−12	6.6317E+01	1.1534E+02	1.4980E+02	1.3510E+02	1.4825E+02	9.4694E+02	9.5928E+02	9.6755E+02	9.8845E+02
	Mean	2.0486E+00	2.3379E+00	1.6030E−−3	2.6527E+00	1.0156E+02	1.1014E+02	1.0857E+02	1.1312E+02	7.4086E+02	7.9728E+02	7.3444E+02	8.4749E+02
	St.dev	4.9643E−01	6.6551E−01	8.0149E−13	1.3263E+01	9.6673E+00	1.1952E+01	1.0619E+01	1.0785E+01	1.9672E+02	1.8685E+02	1.9122E+02	1.1447E+02

Table 12
Results obtained by OIO and COIO on CEC2005 benchmark functions (F20–F25) with $n=10$.

FES		F20		F21		F22		F23		F24		F25	
		OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO
	Best	1.0923E+03	9.9483E+02	1.0481E+03	1.0599E+03	8.6323E+02	9.2933E+02	1.1437E+03	1.2367E+03	1.0063E+03	9.0590E+02	6.3371E+02	7.4113E+02
	Median	1.1403E+03	1.1557E+03	1.3036E+03	1.2999E+03	1.0369E+03	1.0223E+03	1.3185E+03	1.3048E+03	1.3012E+03	1.2950E+03	1.3104E+03	1.3775E+03
1E+3	Worst	1.2178E+03	1.2199E+03	1.3721E+03	1.3498E+03	1.1204E+03	1.0852E+03	1.3634E+03	1.3590E+03	1.3625E+03	1.4087E+03	1.4522E+03	1.4752E+03
	Mean	1.1464E+03	1.1509E+03	1.2934E+03	1.2873E+03	1.0201E+03	1.0184E+03	1.2979E+03	1.3037E+03	1.2823E+03	1.2606E+03	1.1646E+03	1.3068E+03
	St.dev	3.3189E+01	4.9455E+01	7.0105E+01	5.9251E+01	6.5480E+01	4.3258E+01	5.4973E+01	3.0855E+01	8.7260E+01	1.1411E+02	2.4709E+02	1.8687E+02
	Best	7.9403E+02	4.4927E+02	3.4539E+02	3.0012E+02	3.0009E+02	3.0093E+02	5.5947E+02	5.5947E+02	2.0000E+02	2.0000E+02	2.0000E+02	3.9004E+02
	Median	9.7450E+02	9.9582E+02	8.3130E+02	9.8156E+02	7.8416E+02	7.9508E+02	7.2150E+02	9.1909E+02	2.0001E+02	2.0018E+02	4.0209E+02	4.1523E+02
1E+4	Worst	1.0317E+03	1.0292E+03	1.0936E+03	1.1999E+03	8.0007E+02	8.4519E+02	1.2083E+03	1.1916E+03	5.0762E+02	4.9770E+02	6.5478E+02	5.3122E+02
	Mean	9.3930E+02	9.3171E+02	7.6895E+02	9.0621E+02	7.1101E+02	7.1613E+02	7.8599E+02	8.8583E+02	2.2447E+02	2.2005E+02	4.0812E+02	4.1938E+02
	St.dev	8.0410E+01	1.4295E+02	2.3388E+02	2.2856E+02	1.8327E+02	1.8486E+02	2.3832E+02	2.5295E+02	8.4479E+01	6.8812E+01	6.9737E+01	2.7481E+01
	Best	3.0000E+02	3.0000E+02	3.0000E+02	3.0000E+02	3.0000E+02	3.0000E+02	5.5947E+02	5.4759E+02	2.0000E+02	2.0000E+02	2.0000E+02	3.2390E+02
	Median	8.0000E+02	8.0000E+02	5.0000E+02	8.0000E+02	7.2740E+02	7.4614E+02	5.5947E+02	5.5947E+02	2.0000E+02	2.0000E+02	3.7665E+02	3.8422E+02
1E+5	Worst	9.6030E+02	9.5779E+02	8.0000E+02	9.8700E+02	8.0000E+02	8.0000E+02	9.7050E+02	1.1436E+03	2.0000E+02	2.0000E+02	3.8546E+02	3.9392E+02
	Mean	7.2114E+02	7.5377E+02	5.5242E+02	6.5845E+02	5.6078E+02	6.3766E+02	6.1824E+02	7.2224E+02	2.0000E+02	2.0000E+02	3.6306E+02	3.8109E+02
	St.dev	2.3224E+02	2.2026E+02	1.5266E+02	2.1932E+02	2.3698E+02	2.1616E+02	1.2183E+02	2.2574E+02	0.0000E+00	0.0000E+00	4.9220E+01	1.3223E+01

Table 13
Results obtained by OIO and COIO on CEC2005 benchmark functions (F1–F7) with $n=30$.

FES		F1		F2		F3		F4		F5		F6		F7	
		OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO
	Best	4.6416E+04	3.6874E+04	3.2875E+04	4.1219E+04	5.7504E+08	3.7518E+08	4.4322E+04	4.4051E+04	3.1223E+04	2.8043E+04	1.0990E+10	1.2294E+10	4.5458E+03	7.8625E+02
	Median	5.5663E+04	5.4111E+04	5.7557E+04	5.8835E+04	9.6738E+08	7.1172E+08	6.4503E+04	6.9434E+04	3.7203E+04	3.5498E+04	1.9130E+10	2.0196E+10	8.8667E+03	1.3865E+03
1E+3	Worst	6.8317E+04	7.1294E+04	9.5197E+04	6.9235E+04	1.3574E+09	1.4712E+09	1.1625E+05	1.0559E+05	4.1263E+04	4.1627E+04	2.9319E+10	3.0526E+10	9.9891E+03	2.2053E+03
	Mean	5.6273E+04	5.5421E+04	5.8141E+04	5.6939E+04	9.5580E+08	8.3323E+08	6.5777E+04	6.7943E+04	3.6576E+04	3.5544E+04	1.9734E+10	2.0701E+10	8.7188E+03	1.4123E+03
	St.dev	6.6951E+03	8.2076E+03	1.3570E+04	8.1441E+03	2.2712E+08	3.0454E+08	1.4677E+04	1.3195E+04	2.7278E+03	3.4320E+03	4.4479E+09	4.9248E+09	1.1770E+03	3.9382E+02
	Best	3.9148E+03	2.3057E+03	1.2917E+04	1.7529E+04	1.5971E+07	2.2137E+07	1.6768E+04	1.9905E+04	1.4056E+04	9.8418E+03	8.7708E+07	8.6310E+07	7.7790E+02	6.6484E+01
	Median	8.5002E+03	6.6686E+03	1.9265E+04	2.3609E+04	4.1901E+07	6.6063E+07	2.3423E+04	2.9246E+04	1.7005E+04	1.4450E+04	5.8287E+08	5.2026E+08	1.2404E+03	1.2800E+02
1E+4	Worst	2.2677E+04	1.3373E+04	2.8439E+04	2.8198E+04	9.5009E+07	1.1160E+08	3.0479E+04	3.7671E+04	2.0279E+04	1.7145E+04	1.5353E+09	1.2420E+09	1.8522E+03	2.5351E+02
	Mean	8.9566E+03	6.7301E+03	1.9596E+04	2.3673E+04	4.7464E+07	6.1457E+07	2.3412E+04	2.8524E+04	1.6704E+04	1.4134E+04	6.5565E+08	5.6210E+08	1.2878E+03	1.4146E+02
	St.dev	3.4804E+03	2.8296E+03	3.9379E+03	2.7708E+03	2.0374E+07	2.2688E+07	4.3927E+03	4.8781E+03	1.6104E+03	1.9642E+03	4.1560E+08	2.9558E+08	3.2153E+02	5.6096E+01
	Best	3.0481E−03	3.2688E−03	7.5898E+01	2.1732E+03	1.0976E+06	1.9006E+06	3.3328E+03	4.7182E+03	3.8253E+03	4.9900E+03	1.5754E+02	2.5184E+02	1.1810E+00	1.4494E+00
	Median	1.5218E−02	6.3268E−03	2.7214E+02	4.1199E+03	2.1594E+06	7.7905E+06	7.9243E+03	1.5712E+04	4.0789E+03	5.8953E+03	5.4603E+02	3.9169E+02	1.3430E+00	1.6299E+00
1E+5	Worst	4.8142E−02	8.2674E−03	4.9896E+02	9.3548E+03	3.4204E+06	1.4718E+07	9.9837E+03	2.4244E+04	4.4658E+03	6.4138E+03	3.2730E+03	5.0348E+02	3.5791E+00	3.3780E+00
	Mean	2.0582E−02	6.1237E−03	2.8798E+02	4.7515E+03	2.2658E+06	7.5727E+06	7.1149E+03	1.5790E+04	4.1306E+03	5.7161E+03	9.9482E+02	3.9209E+02	1.9252E+00	2.0489E+00
	St.dev	1.8741E−02	1.8071E−03	1.5466E+02	2.7028E+03	8.4438E+05	4.9019E+06	2.5616E+03	7.8039E+03	3.1210E+02	5.8129E+02	1.2958E+03	1.0681E+02	1.0278E+00	7.9138E−01
	Best	0.0000E+00	0.0000E+00	1.0831E−08	7.3965E−09	1.3131E−03	1.5288E−03	9.1116E+01	2.8246E+03	1.7122E+03	3.4480E+03	5.5275E−08	5.8822E−08	5.2125E−09	3.0786E−10
	Median	0.0000E+00	0.0000E+00	7.0317E−08	5.5347E−08	3.0721E−03	3.0180E−03	1.0389E+03	7.1775E+03	2.9896E+03	4.5833E+03	9.8661E−08	1.6931E−07	1.4772E−02	2.4573E−02
3E+5	Worst	1.0232E−12	0.0000E+00	5.0274E−07	7.5383E−07	3.5616E+01	4.1803E+01	2.8539E+03	1.5909E+04	3.8686E+03	6.2775E+03	3.9866E+00	3.9866E+00	4.8906E−02	1.0271E−01
	Mean	5.9117E−14	0.0000E+00	1.2172E−07	1.2611E−07	2.7980E+00	2.1602E+00	1.0383E+03	7.8323E+03	2.8958E+03	4.5967E+03	9.5679E−01	6.3786E−01	1.7407E−02	2.5446E−02
	St.dev	2.2045E−13	0.0000E+00	1.3765E−07	1.9255E−07	7.4333E+00	8.3788E+00	8.0004E+02	3.3759E+03	6.1070E+02	6.7805E+02	1.7377E+00	1.4917E+00	1.2999E−02	2.2452E−02

Table 14
Results obtained by OIO and COIO on CEC2005 benchmark functions (F8–F13) with $n=30$.

FES		F8		F9		F10		F11		F12		F13	
		OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO
	Best	2.0970E+01	2.1000E+01	3.4752E+02	3.1011E+02	5.1170E+02	4.9647E+02	3.9592E+01	4.3790E+01	8.4432E+05	1.1483E+06	8.2785E+02	5.3978E+01
	Median	2.1237E+01	2.1221E+01	3.9700E+02	3.9287E+02	6.1663E+02	6.0687E+02	4.4895E+01	4.6061E+01	1.4025E+06	1.4725E+06	2.7895E+03	1.9797E+02
1E+3	Worst	2.1335E+01	2.1294E+01	4.4171E+02	4.3440E+02	6.9443E+02	7.0696E+02	4.7874E+01	4.7585E+01	1.7655E+06	1.7545E+06	2.1503E+04	3.2668E+02
	Mean	2.1226E+01	2.1200E+01	3.9739E+02	3.9094E+02	6.1512E+02	6.1367E+02	4.4917E+01	4.6048E+01	1.3822E+06	1.4499E+06	4.5414E+03	1.8534E+02
	St.dev	6.7872E−02	7.9651E−02	2.5902E+01	2.6298E+01	4.2637E+01	5.1096E+01	2.1146E+00	1.1927E+00	2.0970E+05	1.6010E+05	4.5830E+03	8.3551E+01
	Best	2.0970E+01	2.0959E+01	1.3861E+02	1.2876E+02	1.8307E+02	1.3659E+02	3.6861E+01	3.6118E+01	1.0384E+05	1.3126E+05	2.0604E+01	1.7128E+01
	Median	2.1096E+01	2.1103E+01	1.6922E+02	1.6661E+02	2.2797E+02	1.8220E+02	3.9520E+01	4.0032E+01	2.1429E+05	2.3929E+05	2.6148E+01	2.5555E+01
1E+4	Worst	2.1207E+01	2.1183E+01	2.1773E+02	2.0750E+02	3.4801E+02	2.2525E+02	4.2473E+01	4.2014E+01	3.2472E+05	3.8030E+05	4.6805E+01	3.7426E+01
	Mean	2.1095E+01	2.1089E+01	1.6917E+02	1.6590E+02	2.3519E+02	1.8280E+02	3.9724E+01	3.9837E+01	2.1873E+05	2.4737E+05	2.7317E+01	2.6201E+01
	St.dev	5.3944E−02	6.3097E−02	1.6725E+01	2.0136E+01	4.9161E+01	2.1935E+01	1.2329E+00	1.3207E+00	5.3661E+04	6.1286E+04	5.9761E+00	4.7078E+00
	Best	2.0950E+01	2.0900E+01	1.2043E+01	4.2767E+00	1.1044E+02	7.7885E+01	1.7745E+01	2.1934E+01	7.7039E+03	1.2547E+04	3.9129E+00	3.5392E+00
	Median	2.1021E+01	2.0997E+01	1.3640E+01	5.6982E+00	1.4925E+02	9.7803E+01	1.8695E+01	2.2998E+01	1.3396E+04	4.4150E+04	4.9180E+00	3.5993E+00
1E+5	Worst	2.1073E+01	2.1055E+01	1.6457E+01	6.9912E+00	1.9103E+02	1.5376E+02	2.2087E+01	2.4270E+01	5.4168E+04	7.8894E+04	6.4520E+00	3.9033E+00
	Mean	2.1013E+01	2.0986E+01	1.3676E+01	5.5200E+00	1.4905E+02	1.0932E+02	1.9228E+01	2.3082E+01	2.0589E+04	4.9087E+04	4.9286E+00	3.6433E+00
	St.dev	4.7777E−02	5.9081E−02	1.7907E+00	1.1304E+00	3.2813E+01	3.2164E+01	1.7559E+00	8.5211E−01	1.9089E+04	2.5669E+04	1.0261E+00	1.5058E−01
	Best	2.0000E+01	2.0000E+01	1.7621E−12	2.1600E−12	8.9546E+01	8.5566E+01	1.4170E+01	1.7186E+01	1.5693E−08	1.5246E−08	6.6327E−01	1.5841E−01
	Median	2.0000E+01	2.0000E+01	2.3874E−12	2.5011E−12	1.3233E+02	1.1442E+02	1.8630E+01	2.2155E+01	4.9032E+02	5.1146E+02	1.0762E+00	4.0601E−01
3E+5	Worst	2.0000E+01	2.0000E+01	9.9496E−01	2.8990E−12	1.7810E+02	1.5223E+02	2.5652E+01	2.6041E+01	1.0802E+04	5.2514E+03	1.8786E+00	9.4906E−01
	Mean	2.0000E+01	2.0000E+01	3.9798E−02	2.4852E−12	1.3384E+02	1.1534E+02	1.9122E+01	2.2108E+01	1.4058E+03	1.6264E+03	1.1044E+00	4.4599E−01
	St.dev	7.7690E−07	5.0809E−06	1.9899E−01	2.0919E−13	2.2765E+01	1.6757E+01	3.3239E+00	2.3074E+00	2.3863E+03	1.8141E+03	2.7700E−01	2.1734E−01

Table 15
Results obtained by OIO and COIO on CEC2005 benchmark functions (F14–F19) with $n=30$.

FES		F14		F15		F16		F17		F18		F19	
		OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO
	Best	1.3931E+01	1.3842E+01	9.6041E+02	9.7641E+02	8.2952E+02	8.2236E+02	6.7451E+02	8.4128E+02	1.2574E+03	9.8544E+02	1.2456E+03	9.5145E+02
	Median	1.4206E+01	1.4124E+01	1.1216E+03	1.1530E+03	9.7157E+02	9.9114E+02	1.0714E+03	1.0700E+03	1.3211E+03	1.1223E+03	1.3347E+03	1.1461E+03
1E+3	Worst	1.4297E+01	1.4447E+01	1.2225E+03	1.2342E+03	1.1510E+03	1.1071E+03	1.2716E+03	1.3081E+03	1.3800E+03	1.3415E+03	1.3823E+03	1.2768E+03
	Mean	1.4196E+01	1.4127E+01	1.1232E+03	1.1496E+03	9.9000E+02	9.7754E+02	1.0483E+03	1.0762E+03	1.3184E+03	1.1703E+03	1.3287E+03	1.1303E+03
	St.dev	8.1362E−02	1.5738E−01	7.0029E+01	5.6477E+01	8.1128E+01	7.9627E+01	1.4003E+02	1.1198E+02	2.9930E+01	9.4697E+01	3.3142E+01	8.9915E+01
	Best	1.3120E+01	1.3301E+01	5.2968E+02	5.6178E+02	2.7271E+02	2.6019E+02	2.9576E+02	3.0943E+02	1.0493E+03	9.0000E+02	1.0235E+03	9.0000E+02
	Median	1.3733E+01	1.3680E+01	6.6882E+02	6.9072E+02	3.5140E+02	3.4402E+02	4.0820E+02	3.7818E+02	1.0958E+03	9.0000E+02	1.0768E+03	9.0000E+02
1E+4	Worst	1.3930E+01	1.3986E+01	7.7483E+02	7.8161E+02	6.1014E+02	6.1038E+02	6.8526E+02	7.0843E+02	1.1253E+03	9.0002E+02	1.1403E+03	9.0001E+02
	Mean	1.3693E+01	1.3681E+01	6.6207E+02	6.8104E+02	3.7003E+02	3.8467E+02	4.2409E+02	4.4673E+02	1.0888E+03	9.0000E+02	1.0819E+03	9.0000E+02
	St.dev	1.8596E−01	1.6041E−01	6.9457E+01	7.2438E+01	7.2478E+01	1.0451E+02	8.0258E+01	1.3563E+02	2.1636E+01	3.9755E−03	2.7527E+01	2.2546E−03
	Best	1.2262E+01	1.2077E+01	1.5255E+02	1.3885E+02	8.8894E+01	1.1915E+02	1.2857E+02	9.4859E+01	8.0007E+02	9.0000E+02	8.7779E+02	9.0000E+02
	Median	1.2571E+01	1.2472E+01	1.7223E+02	1.9702E+02	1.3007E+02	1.3899E+02	1.4015E+02	1.7105E+02	9.0001E+02	9.0000E+02	9.4320E+02	9.0000E+02
1E+5	Worst	1.3136E+01	1.2837E+01	1.9417E+02	2.2930E+02	1.7936E+02	1.6392E+02	1.8524E+02	1.8630E+02	9.6207E+02	9.0000E+02	9.6680E+02	9.0000E+02
	Mean	1.2704E+01	1.2492E+01	1.7209E+02	1.9071E+02	1.3557E+02	1.4275E+02	1.4879E+02	1.5586E+02	9.0326E+02	9.0000E+02	9.3594E+02	9.0000E+02
	St.dev	3.6012E−01	3.0396E−01	1.5602E+01	3.3028E+01	3.3969E+01	1.8570E+01	2.4017E+01	3.5820E+01	6.4645E+01	0.0000E+00	3.4412E+01	0.0000E+00
	Best	1.0118E+01	9.4222E+00	1.4497E+01	6.7088E+00	8.6183E+01	9.9978E+01	9.9792E+01	8.1488E+01	8.0000E+02	9.0000E+02	8.0000E+02	8.0000E+02
	Median	1.0866E+01	1.0960E+01	5.1060E+01	5.1623E+01	1.3641E+02	1.5199E+02	1.3597E+02	1.3294E+02	9.2207E+02	9.0000E+02	9.1906E+02	9.0000E+02
3E+5	Worst	1.1695E+01	1.1750E+01	8.0278E+01	1.3003E+02	2.8191E+02	2.1404E+02	3.4096E+02	2.6284E+02	9.2808E+02	9.0000E+02	9.3189E+02	9.0000E+02
	Mean	1.0889E+01	1.0977E+01	5.1459E+01	5.3960E+01	1.4320E+02	1.5412E+02	1.4251E+02	1.4073E+02	8.7381E+02	9.0000E+02	8.8783E+02	8.9600E+02
	St.dev	4.1727E−01	5.2329E−01	1.6115E+01	2.6119E+01	3.9571E+01	3.2605E+01	4.7125E+01	3.7937E+01	6.1555E+01	0.0000E+00	5.4991E+01	2.0000E+01

Table 16
Results obtained by OIO and COIO on CEC2005 benchmark functions (F20–F25) with $n=30$.

FES		F20		F21		F22		F23		F24		F25	
		OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO	OIO	COIO
1E+3	Best	1.2588E+03	9.3451E+02	1.3471E+03	1.3651E+03	1.4209E+03	1.4524E+03	1.3597E+03	1.3601E+03	1.4162E+03	1.4203E+03	1.7175E+03	1.4993E+03
	Median	1.3314E+03	1.1648E+03	1.3944E+03	1.4073E+03	1.5773E+03	1.5392E+03	1.3953E+03	1.4011E+03	1.4503E+03	1.4528E+03	1.8006E+03	1.5506E+03
	Worst	1.3775E+03	1.3102E+03	1.4459E+03	1.4410E+03	1.7499E+03	1.6716E+03	1.4453E+03	1.4380E+03	1.4787E+03	1.4910E+03	1.8966E+03	1.6739E+03
	Mean	1.3250E+03	1.1438E+03	1.3914E+03	1.4087E+03	1.5723E+03	1.5552E+03	1.3966E+03	1.4008E+03	1.4505E+03	1.4561E+03	1.8002E+03	1.5583E+03
	St.dev	3.2492E+01	1.0859E+02	2.2609E+01	2.2270E+01	8.1255E+01	6.6403E+01	1.9046E+01	2.0163E+01	1.5431E+01	1.9783E+01	5.2984E+01	3.9727E+01
1E+4	Best	1.0229E+03	9.0000E+02	1.1778E+03	1.1019E+03	1.0951E+03	1.1011E+03	1.1880E+03	1.0648E+03	1.0535E+03	1.0399E+03	7.5755E+02	9.7800E+02
	Median	1.0849E+03	9.0000E+02	1.2420E+03	1.2078E+03	1.1868E+03	1.1935E+03	1.2471E+03	1.2154E+03	1.3009E+03	1.2703E+03	1.3302E+03	1.3429E+03
	Worst	1.1223E+03	9.0009E+02	1.2707E+03	1.2687E+03	1.2452E+03	1.2599E+03	1.2920E+03	1.2678E+03	1.3539E+03	1.3421E+03	1.4235E+03	1.3978E+03
	Mean	1.0874E+03	9.0001E+02	1.2431E+03	1.2046E+03	1.1714E+03	1.1861E+03	1.2448E+03	1.2164E+03	1.2868E+03	1.2604E+03	1.2510E+03	1.3194E+03
	St.dev	2.3412E+01	1.9883E−02	2.0570E+01	3.8475E+01	3.9421E+01	5.3112E+01	2.3559E+01	4.0184E+01	6.3697E+01	6.4277E+01	1.8450E+02	8.1383E+01
1E+5	Best	9.0011E+02	9.0000E+02	5.0001E+02	5.0060E+02	9.5939E+02	9.6088E+02	5.4052E+02	5.3421E+02	2.0003E+02	2.0481E+02	2.1781E+02	2.2634E+02
	Median	9.3678E+02	9.0000E+02	5.0009E+02	5.0241E+02	9.8426E+02	1.0190E+03	5.7088E+02	5.3890E+02	2.0007E+02	2.9691E+02	2.2100E+02	2.4550E+02
	Worst	9.6984E+02	9.0000E+02	5.0033E+02	5.2177E+02	1.0203E+03	1.0875E+03	6.1026E+02	5.5182E+02	2.0031E+02	4.7108E+02	2.3788E+02	2.6256E+02
	Mean	9.3301E+02	9.0000E+02	5.0013E+02	5.0607E+02	9.8882E+02	1.0150E+03	5.7090E+02	5.4085E+02	2.0012E+02	3.3330E+02	2.2676E+02	2.4351E+02
	St.dev	2.5640E+01	0.0000E+00	1.2823E−01	8.9221E+00	2.8433E+01	5.4322E+01	2.7805E+01	6.7564E+00	1.1782E−01	1.2863E+02	9.4636E+00	1.5785E+01
	Best	8.0000E+02	8.0049E+02	5.0000E+02	5.0000E+02	9.0763E+02	9.0639E+02	5.3416E+02	5.3416E+02	2.0000E+02	2.0000E+02	2.1215E+02	2.1476E+02
	Median	9.1917E+02	9.0000E+02	5.0000E+02	5.0000E+02	9.4672E+02	9.3832E+02	5.3416E+02	5.3416E+02	2.0000E+02	2.0000E+02	2.1386E+02	2.1963E+02
3E+5	Worst	9.3006E+02	9.0000E+02	5.0000E+02	5.0000E+02	1.0311E+03	1.0087E+03	5.3417E+02	5.3417E+02	2.0000E+02	2.0000E+02	2.1753E+02	2.5615E+02
	Mean	8.8591E+02	8.9602E+02	5.0000E+02	5.0000E+02	9.5006E+02	9.4781E+02	5.3416E+02	5.3416E+02	2.0000E+02	2.0000E+02	2.1426E+02	2.2316E+02
	St.dev	5.5295E+01	1.9903E+01	5.5265E−12	5.5781E−11	2.7525E+01	2.8736E+01	4.4466E−04	3.8109E−04	4.3422E−11	2.7334E−05	1.4876E+00	9.4918E+00

and COIO, the best solution achieved being refined using Quasi-Newton method with 0. 05*Max_FEs fitness evaluations. Where, Max_FEs is the maximum number of function evaluations which is equal to 100,000 for 10D functions and 300,000 for 30D functions.

Let us first start with the case of 10D functions. The first five functions are unimodal functions, F1 is Shifted Sphere, F2 is shifted Schwefel 1.2, and F3 is shifted rotated high conditioned elliptic function. F3 is harder than F2 and F2 is harder than F1. We observe that DMS-L-PSO, OIO and COIO achieve better result for F1 than F2; better result for F2 than F3. On F1, all algorithms perform the same. While On F2, OIO always find the global optimum, DMS-L-PSO and COIO converge truly to the global optimum with a very low standard deviation. On F3, DMS-L-PSO's mean performance is significantly better than OIO and COIO. Though both of OIO and COIO perform the same and can find the global optimum with a relatively good precision. Function 4 is shifted Schwefel 1.2 with noise in fitness. On this function OIO is able to find the global minimum. Moreover the average performance of OIO is far better than that of DMS-L-PSO and COIO. The very small standard deviation value reported by OIO indicate that it converges really close to the global optimum in each run. Here COIO performs a bit better than DMS-L-PSO. F5 is Schwefel 2.6 function with global optimum on bounds. On this function DMS-L-PSO and OIO perform almost the same. However the performance of COIO is not competitive. Functions F6 to F25 are multimodal functions. F6 is the shifted Rosenbrock function, a function between unimodal and multimodal and an algorithm with good local search ability can achieve good results. The performance of COIO is very satisfactory in all runs and better than DMS-L-PSO and OIO. Though these latter algorithms also perform well. F7 is the shifted rotated Griewank function without bounds. Only the initialization range is given. Here the performance of DMS-L-PSO is the best and OIO performs rather comparable. COIO is in turn less competitive. F8 is shifted rotated Ackley's function with global optimum on bounds, and has a very narrow global basin and the search is like seeking a needle in a haystack. Here, all algorithms perform constantly non-optimal. As can be seen from Table 10, difference between the final output of the search and its output after FEs=1e+3 is quite small and algorithms get stuck with no freedom to do a smooth search. F9 is the shifted Rastrigin function with a huge number of local optima. However all algorithms are able to perform always optimal. F10 is the shifted rotated Rastrigin function on which, the search is quickly biased toward global optimum but it always is stuck in a local optimum. Here, the average performance of DMS-L-PSO is better than others. F11 is the shifted rotated Weierstrass function. Though the results are not very good, OIO performs better than rivals. Here COIO takes the second place. F12 is Schwefel's problem. All algorithms are able to find the global optimum of F12 with a very good precision repeatedly. However, the average performance of DMS-L-PSO is better than OIO and COIO. On both F13 and F14 functions, DMS-L-PSO is inferior to OIO based algorithms. F15 to F25 are all composition functions built up with basic functions. They are most time consuming functions for evaluation and give a big challenge to any search algorithm. On F15, both of OIO and COIO perform optimal in all runs. However, the worst performance of DMS-L-PSO is far from optimum. Albeit the success rate of this algorithm is 88%. None of DMS-L-PSO, OIO and COIO are not able to find any optimal solution for functions F16 to F25. However, the overall average performance of OIO is better than rivals. On 7 out of 10 functions (F17, F18, F20, F22-F25) OIO takes the first rank (values in parenthesis indicate the ranks). This record for DMS-L-PSO is 3 (F16, F19 and F21) and for COIO is only 1 (F24). From results of Table 17 it can be found that the performance of OIO is acceptable, especially in comparison with an advanced algorithm like DMS-L-PSO.

Table 17
Results obtained by different algorithms on CEC2005 benchmark functions.

Function		$n=10$			$n=30$		
		DMS-L-PSO	**OIO**	**COIO**	**DMS-L-PSO**	**OIO**	**COIO**
F1	Mean	0.0000E+00 (1)	0.0000E+00 (1)	0.0000E+00 (1)	0.00E+00 (1)	5.9117E−14 (2)	0.0000E+00 (1)
	St.dev	0.0000E+00	0.0000E+00	0.0000E+00	0.00E+00	2.2045E−13	0.0000E+00
F2	Mean	1.2960e−E13 (3)	0.0000E+00 (1)	1.0603E−09 (2)	1.1757E−007 (2)	1.0118E−07 (1)	1.2611E−07 (3)
	St.dev	1.5612.013	0.0000E+00	3.0502E−09	6.5592E−008	1.2838E−07	1.9255E−07
F3	Mean	7.0064E−09 (1)	1.2365E+00 (2)	6.2145E+00 (3)	1.6343E−006 (1)	2.7980E+00 (3)	2.1602E+00 (2)
	St.dev	2.6589E−09	3.6749E+00	1.7393E+01	3.9247E−006	7.4333E+00	8.3788E+00
F4	Mean	1.8051.E−03 (3)	2.8462E−10 (1)	1.0798E−04 (2)	2.5487E+003 (2)	1.0383E+03 (1)	7.8323E+03 (3)
	St.dev	1.8932E−03	1.1932E−09	3.9186E−04	3.0638E+002	8.0004E+02	3.3759E+03
F5	Mean	1.1383E−06 (1)	1.4801E−05 (2)	1.3554E+01 (3)	2.1858E+003 (1)	2.8958E+03 (2)	4.5967E+03 (3)
	St.dev	2.1828E−06	2.6005E−05	2.9080E+01	8.2641E+002	6.1070E+02	6.7805E+02
F6	Mean	6.8925E−08 (2)	1.6738E−06 (3)	4.3841E−09 (1)	4.7840E−001 (1)	9.5679E−01 (3)	6.3786E−01 (2)
	St.dev	3.1904E−07	6.2281E−06	5.4563E−10	1.3222E+000	1.7377E+00	1.4917E+00
F7	Mean	4.5109E−02 (1)	8.2112E−02 (2)	1.9422E−01 (3)	6.9990E−003 (1)	1.7407E−02 (2)	2.5446E−02 (3)
	St.dev	3.2611E−02	4.5783E−02	1.2914E−01	4.5371E−003	1.2999E−02	2.2452E−02
F8	Mean	2.0000E+001 (1)	2.0000E+01 (1)	2.0000E+01 (1)	2.0000E+001 (1)	2.0000E+01 (1)	2.0000E+01 (1)
	St.dev	5.5382E−09	2.5658E−06	7.7677E−06	2.3029E−004	7.7690E−07	5.0809E−06
F9	Mean	0.0000E+00 (1)	0.0000E+00 (1)	0.0000E+00 (1)	1.7591E+001 (3)	3.9798E−02 (2)	2.4852E−12 (1)
	St.dev	0.0000E+00	0.0000E+00	0.0000E+00	3.0222E+000	1.9899E−01	2.0919E−13
F10	Mean	3.6217E+00 (1)	5.6912E+00 (2)	6.6065E+00 (3)	3.7410E+001 (1)	1.3384E+02 (3)	1.1534E+02 (2)
	St.dev	8.5590E−01	1.7593E+00	2.7078E+00	5.2883E+000	2.2765E+01	1.6757E+01
F11	Mean	4.6229E+00 (3)	2.0814E+00 (1)	3.6384E+00 (2)	2.7278E+001 (3)	1.9122E+01 (1)	2.2108E+01 (2)
	St.dev	5.8400E−01	8.4384E−01	1.0212E+00	1.5739E+000	3.3239E+00	2.3074E+00
F12	Mean	2.4007E+00 (1)	7.7213E+00 (2)	1.1266E+01 (3)	2.5359E+002 (1)	1.4058E+03 (2)	1.6264E+03 (3)
	St.dev	4.3602E+00	7.7419E+00	7.0834E+00	2.8883E+002	2.3863E+03	1.8141E+03
F13	Mean	3.6865E−01 (3)	2.4798E−01 (2)	1.0884E−01 (1)	2.3595E+000 (3)	1.1044E+00 (2)	4.4599E−01 (1)
	St.dev	5.6411E−02	1.0938E−01	8.4099E−02	5.2023E−001	2.7700E−01	2.1734E−01
F14	Mean	2.3601E+00 (3)	2.0486E+00 (1)	2.3379E+00 (2)	1.1961E+001 (3)	1.0889E+01 (1)	1.0977E+01 (2)
	St.dev	3.3750E−01	4.9643E−01	6.6551E−01	4.1146E−001	4.1727E−01	5.2329E−01
F15	Mean	4.8539E+00 (3)	1.6030E−13 (1)	3.2685E−13 (2)	3.4400E+002 (3)	5.1459E+01 (1)	5.3960E+01 (2)
	St.dev	1.3415E+01	8.0149E−13	1.09658E−12	5.0662E+001	1.6115E+01	2.6119E+01
F16	Mean	9.4756E+01 (1)	1.0156E+02 (2)	1.1014E+02 (3)	1.1950E+002 (1)	1.4320E+02 (2)	1.5412E+02 (3)
	St.dev	1.0086E+01	9.6673E+00	1.1952E+01	1.2068E+002	3.9571E+01	3.2605E+01
F17	Mean	1.1009E+02 (2)	1.0857E+02 (1)	1.1312E+02 (3)	1.4519E+002 (3)	1.4251E+02 (2)	1.4073E+02 (1)
	St.dev	4.3453E+00	1.0619E+01	1.0785E+01	7.3247E+001	4.7125E+01	3.7937E+01
F18	Mean	7.6067E+02 (2)	7.4086E+02 (1)	7.9728E+02 (3)	9.1053E+002 (3)	8.7381E+02 (1)	9.0000E+02 (2)
	St.dev	1.8458E+02	1.9672E+02	1.8685E+02	1.5761E+000	6.1555E+01	0.0000E+00
F19	Mean	7.14303E+02 (1)	7.3444E+02 (2)	8.4749E+02 (3)	9.1060E+002 (3)	8.8783E+02 (1)	8.9600E+02 (2)
	St.dev	2.0105E+02	1.9122E+02	1.1447E+02	1.3383E+000	5.4991E+01	2.0000E+01
F20	Mean	8.2106E+02 (3)	7.2114E+02 (1)	7.5377E+02 (2)	9.0189E+002 (3)	8.8591E+02 (1)	8.9602E+02 (2)
	St.dev	4.5874E+01	2.3224E+02	2.2026E+02	3.0719E+001	5.5295E+01	1.9903E+01
F21	Mean	5.3600E+02 (1)	5.5242E+02 (2)	6.5845E+02 (3)	NA	5.0000E+02 (1)	5.0000E+02 (1)
	St.dev	2.1772E+02	1.5266E+02	2.1932E+02	NA	5.5265E−12	5.5781E−11
F22	Mean	6.9242E+02 (3)	5.6078E+02 (1)	6.3766E+02 (2)	NA	9.5006E+02 (2)	9.4781E+02 (1)
	St.dev	1.5647E+02	2.3698E+02	2.1616E+02	NA	2.7525E+01	2.8736E+01
F23	Mean	7.3034E+02 (3)	6.1824E+02 (1)	7.2224E+02 (2)	NA	5.3416E+02 (1)	5.3416E+02 (1)
	St.dev	1.6620E+02	1.2183E+02	2.2574E+02	NA	4.4466E−04	3.8109E−04
F24	Mean	2.2400E+02 (2)	2.0000E+02 (1)	2.0000E+02 (1)	NA	2.0000E+02 (1)	2.0000E+02 (1)
	St.dev	8.3066E+01	0.0000E+00	0.0000E+00	NA	4.3422E−11	2.7334E−05
F25	Mean	3.6571E+02 (2)	3.6306E+02 (1)	3.8109E+02 (3)	NA	2.1426E+02 (1)	2.2316E+02 (2)
	St.dev	1.5096E+02	4.9220E+01	1.3223E+01	NA	1.4876E+00	9.4918E+00
Points		**48**	**36**	**55**	**at least 44**	**40**	**48**

On 15 out of 25 10D functions (F1, F2, F4, F8, F9, F11, F14, F15, F17, F18, F20, F22-F25) OIO is ranked first. This record is 11 out of 25 (F1, F3, F5, F7-F10, F12, F16, F19 and F21) for DMS-L-PSO and 6 out of 25 (F1, F6, F8, F9, F13 and F24) for COIO. There was only one function (F6) for which the performance of OIO becomes worst among rivals.

Back to Table 17 for 30D functions, the results of DMS-L-PSO have been reported only for the first twenty functions by Liang and Suganthan [26]. However, we have reported the performance of OIO and COIO on the remaining five functions, too. From the results of Table 17 we can see that on 11 out of the first 20 functions, the same pattern of rankings is observable between two cases of 10D and 30D. These functions are F1, F5, F7, F8, F11-F15, F16 and F20. On F2, all algorithms exhibit a same acceptable pattern of mean performance. However, performances are not as good as the case of 10D. On F3, DMS-L-PSO's mean performance is significantly better than OIO and COIO. Here, COIO performs a bit better than OIO. On F4, all algorithms exhibit a significantly worse performance compared to the case of 10D. Again OIO is showing a better mean performance. From Table 13 it can be seen that the best performance of OIO (i.e., 9.1116E+01) is significantly better than COIO (i.e., 2.8246E+03) and also DMS-L-PSO (i.e., 2.0255e+003). On F6, the mean performance of the algorithms are acceptable but not as good as the case of 10D. Though all algorithms always found the global optimum of the 10D shifted Rastrigin function (F9) in all runs, in case of 30D this is only COIO that shows a remarkable performance and always finds the global optimum. The performance of OIO is also far better than DMS-L-PSO. On F10 the performance of all algorithm is being worse than the case of 10D. However, DMS-L-PSO performs considerably better than others. On F17, the performance degradation is still observable. However, the mean performances are comparable. On F18, COIO always ends with a same solution in all of 25 runs. Such a solution is

Table 18
Significance test for DMS-L-PSO, OIO and COIO algorithms on CEC2005 benchmark functions.

Function	n=10			n=30		
	DMS-L-PSO vs OIO	DMS-L-PSO vs COIO	OIO vs COIO	DMS-L-PSO vs OIO	DMS-L-PSO vs COIO	OIO vs COIO
F1	–	–	–	–	–	–
F2	OIO	DMS-L-PSO	OIO	–	–	–
F3	–	DMS-L-PSO	–	DMS-L-PSO	–	–
F4	OIO	COIO	–	OIO	DMS-L-PSO	OIO
F5	DMS-L-PSO	DMS-L-PSO	OIO	DMS-L-PSO	DMS-L-PSO	OIO
F6	–	–	–	–	–	–
F7	DMS-L-PSO	DMS-L-PSO	OIO	DMS-L-PSO	DMS-L-PSO	
F8	–	–	–	–	–	–
F9	–	–	–	OIO	COIO	–
F10	DMS-L-PSO	DMS-L-PSO	–	DMS-L-PSO	DMS-L-PSO	COIO
F11	OIO	COIO	OIO	OIO	COIO	OIO
F12	DMS-L-PSO	DMS-L-PSO	OIO	DMS-L-PSO	DMS-L-PSO	–
F13	OIO	COIO	COIO	OIO	COIO	COIO
F14	OIO	–	OIO	OIO	COIO	–
F15	OIO	COIO	–	OIO	COIO	–
F16	DMS-L-PSO	DMS-L-PSO	OIO	–	–	–
F17	–	–	–	–	–	–
F18	–	–	–	OIO	COIO	OIO
F19	–	DMS-L-PSO	OIO	OIO	COIO	–
F20	OIO	–	–	–	–	–
F21	–	DMS-L-PSO	OIO	–	–	–
F22	OIO	–	–	–	–	–
F23	OIO	–	OIO	–	–	–
F24	–	–	–	–	–	–
F25	–	–	OIO	–	–	OIO
Total	**5 vs 9**	**9 vs 4**	**11 vs 1**	**5 vs 8**	**5 vs 7**	**4 vs 2**

better than the best solution obtained by DMS-L-PSO. The mean performance of OIO is better than COIO. The performance of algorithms On F19 is almost similar to their performance on F18.

From results of Table 17 it can be found that the performance of OIO is still acceptable. On 9 out of 20 functions (F2, F4, F8, F11, F14, F15, F18, F19 and F20) OIO is ranked first. This record is 9 out of 20 (F1, F3, F5-F8, F10, F12 and F16,) for DMS-L-PSO and 5 out of 20 (F1, F8, F9, F13 and F7) for COIO.

The last row in Table 17 shows the total points gathered by each algorithm (through summation over all rank values). We see that over 10D functions, OIO is ranked first, DMS-L-PSO is ranked second and COIO is ranked third. However, no straight ranking is possible on 30D functions.

A similar statistical analysis followed in Experiment 1 is conducted to find out whether the difference between the mean performances of algorithms is significant or not (see Table 18). When comparing OIO versus DMS-L-PSO on 10D functions, the statistical tests show that on 9 functions the mean performance of OIO is significantly better than DMS-L-PSO and on 5 functions the situation is reverse. Comparing with COIO, OIO performs better in 11 functions. This is only F13 on which COIO's mean performance is better than OIO. DMS-L-PSO is performing better on 9 functions but worse on 4 functions than COIO. Significance summaries for 30D functions have been demonstrated in the last row of Table 18. While on 10D functions the final ranking of algorithms would be OIO > DMS-L-PSO > COIO, on 30D functions the true ranking seems to be OIO > COIO = DMS-L-PSO. (where " > " denotes "generally better" relationship and "=" denotes "almost equal" relationship).

While in Experiments 1, 2 3 and 5 (see the next section), COIO performed a more efficient search than OIO on many functions, results on CEC2005 demonstrated another fact that COIO may become inferior to OIO on several functions (especially 10D functions). Our justifications for such outcomes are in twofold. First, such a conclusion is not absolute and in all experiments there were a number of functions that OIO was inferior to COIO, vice versa. Due to the existence of the No-Free Lunch (NFL) Theorem, only via problem-specific knowledge can an algorithm outperform randomness. Therefore we cannot expect to design a perfect OIO which performs the best for all the optimization problems. Second, most functions in Experiments 1, 2, 3 are symmetric functions and the notion of rotation via symmetric vector reflection used in COIO helps to bias the search more efficient toward the global minimum inside a symmetric valley. Therefore on symmetric or locally symmetric smooth functions, COIO performs well. However, CEC2005 set of functions are itself rotated and not even locally symmetric. Therefore, it seems that the vector rotation mechanism of COIO does not helps more.

5.5. Experiment 5: A real world engineering design application: Bi-objective optimization of a centrifuge pump

Centrifugal pumps are widely used in process industries for different applications, such as lifting fluid from one level to another [27]. A centrifugal pump is a turbo pump that uses a rotating impeller to increase the pressure of a fluid. Centrifugal pumps work by converting kinetic energy into potential energy which is measured as the static fluid pressure at the outlet of the pump. The definitions of the most important hydraulic parameters in a centrifugal pump are given below [27]:

- η: is the efficiency of a centrifugal pump and is defined by $\eta = p_{out}/p_{in}$
- *NPSHr*: is the amount of energy in the liquid which is required to overcome the friction losses from the suction nozzle to the eye of the impeller without causing vaporization. *NPSHr* is a design feature and varies by size and the operating conditions. Increasing *NPSHr* may results in reduction or halting of the fluid flow.

Nourbakhsh et al., [27] modeled η and *NPSHr* using GMDH-type neural network to map inputs to outputs. They have used such networks for input–output data to find the polynomial models of η and *NPSHr* with respect to their effective input parameters. They have obtained the corresponding polynomial representation for η as follows:

$$Y_1 = 0.476 - 0.33\beta_{1Hub} + 2.027\beta_{1Shroud} + 0.014\beta_{1Hub}^2 - 0.013\beta_{1Shroud}^2 + 0.0001\beta_{1Hub}\beta_{1Shroud}$$

$$Y_2 = 20.3595 + 1.1797\gamma_{mid} + 0.5391\beta_2 + 0.00787\gamma_{mid}^2 - 0.00397\beta_2^2 + 0.00115\gamma_{mid}\beta_2$$

$$Y_3 = -17.93 + 2.01\beta_{1Shroud} + 0.5805\beta_2 - 0.013\beta_{1Shroud}^2 - 0.00397\beta_2^2 + 0.0002\beta_{1Shroud}\beta_2$$

$$Y_4 = 37.03 + 1.228\gamma_{mid} - 0.352\beta_{1Hub} - 0.0078\gamma_{mid}^2 + 0.0142\beta_{1Hub}^2 + 0.00062\gamma_{mid}\beta_{1Hub}$$

$$Y_5 = 60.5535 - 0.66962Y_1 - 0.91212Y_2 + 0.004299Y_1^2 + 0.005978Y_2^2 + 0.012869Y_1Y_2$$

$$Y_6 = 57.0403 - 0.52137Y_3 - 0.97334Y_4 + 0.003361Y_3^2 + 0.0063741Y_4^2 + 0.0128Y_4Y_3$$

$$\eta = 0.68350 - 3.42012Y_5 + 4.39817Y_6 - 2.330201Y_5^2 - 2.37611Y_6^2 + 4.706481Y_5Y_6 \quad (17)$$

The polynomial representation of the model for *NPSHr* is in the form of

$$Y_1' = -1.62 - 0.014\beta_{1Hub} + 0.16\beta_{1Shroud} + 0.005\beta_{1Hub}^2 - 0.0010\beta_{1Shroud}^2$$

$$+2.2\times10^{-6}\beta_{1Hub}\beta_{1Shroud}$$

$$Y_2' = 3.426-0.0152\beta_{1Hub}+0.0177\beta_2+0.0005\beta_{1Hub}^2 +3.109\times10^{-12}\beta_2^2+1.22\times10^{-5}\beta_2\beta_{1Hub}$$

$$Y_3' = 6.175-0.1292\gamma_{mid}-0.01609\beta_{1Hub}+0.001\gamma_{mid}^2+0.0005\beta_{1Hub}^2 +2.83\times10^{-5}\gamma_{mid}\beta_{1Hub}$$

$$Y_4' = -2.47+0.159\beta_{1Shroud}+0.017\beta_2-0.0010\beta_{1Shroud}^2 +1.27\times10^{-11}\beta_2^2+1.35\times10^{-5}\beta_{1Shroud}\beta_2$$

$$Y_5' = 7.5491-3.5231Y_1'+0.42880Y_2'+0.46290Y_1'^2+0.019501Y_2'^2 +0.060303Y_1'Y_2'$$

$$Y_6' = 5.9118-1.963Y_3'-0.6936Y_4'+0.21291308Y_3'^2 +0.0801906Y_4'^2+0.2316076Y_3'Y_4'$$

$$NPSHr = -0.3809-0.0652Y_5'+1.217Y_6'-0.0704Y_5'^2 +0.030280Y_6'^2-0.1160804Y_5'Y_6' \quad (18)$$

where $\beta_{1Hub}, \beta_{1Shroud}, \gamma_{mid}$ and β_2 are design variables. Optimization of centrifugal pumps is a multi-objective optimization problem rather than the single objective optimization. The two conflicting objectives of η and *NPSHr* have to be simultaneously optimized with respect to the design variables. The multi-objective optimization of a centrifuge pump can be formulated as follows [27]:

$$\begin{cases} \text{Maximize} \quad \eta \\ \text{Minmize} \quad NPSHr \\ st: 30^\circ \le \gamma_{mid} \le 70^\circ \\ \quad 0^\circ \le \beta_{1Hub} \le 30^\circ \\ \quad 60^\circ \le \beta_{1Shroud} \le 89^\circ \\ \quad 40^\circ \le \beta_2 \le 60^\circ \end{cases} \quad (19)$$

To solve the above problem with OIO, ROIO and COIO algorithms which have been designed for optimization over the single objective environments we convert the above bi-objective problem into a weighted sum single objective problem as follows:

$$\begin{cases} \text{Minmize} \quad -w_i\eta+(1-w_i)NPSHr \\ st: 30^\circ \le \gamma_{mid} \le 70^\circ \\ \quad 0^\circ \le \beta_{1Hub} \le 30^\circ \\ \quad 60^\circ \le \beta_{1Shroud} \le 89^\circ \\ \quad 40^\circ \le \beta_2 \le 60^\circ \end{cases} \quad (20)$$

where w_i is a given weight in the range of $0 \le w_i \le 1$. To obtain the Pareto solutions of the problem (19) via solving problem (20) by an algorithm, we set $w_i=(i-1)/10,000, \quad \forall i=1,...,10,001$. This means that each algorithm is run 10,001 times on problem (20) with different weight w_i and the non-dominated Pareto front is obtained accordingly.

All of OIO, ROI and COIO algorithms are executed with 15000 number of function evaluations allowed for each of 10,001 runs. All parameter settings are similar to that were used in Experiment 1 except for the population size which is set equal to 50 for all of OIO, ROIO and COIO. We use from PSO, which have been used by Nourbakhsh et al., [27], as the comparator algorithm with parameters: $c_1=c_2=1.49445$, *swarm size*=100, *maximum iterations*=150, $V_{min}=-4$ and $V_{max}=4$. The inertia weight is linearly decreased from 1 to 0.

Fig. 12 depicts the obtained non-dominated design points by each algorithm as a Pareto front of the two objective functions of problem (19). As can be seen from the results, all of OIO, ROIO and

Table 19
Results of different algorithms on bi-objective optimization of centrifuge pumps.

Algorithm	No. of Parerto solutions in 10001 runs (Diversity goal)	No. of Parerto solutions in the final Pareto-set (P_C).	% Of contribution in final Pareto-set (Quality goal)
PSO	629	103	0.57
OIO	7251	4817	26.81
ROIO	6068	3493	19.44
COIO	9559	9553	53.17

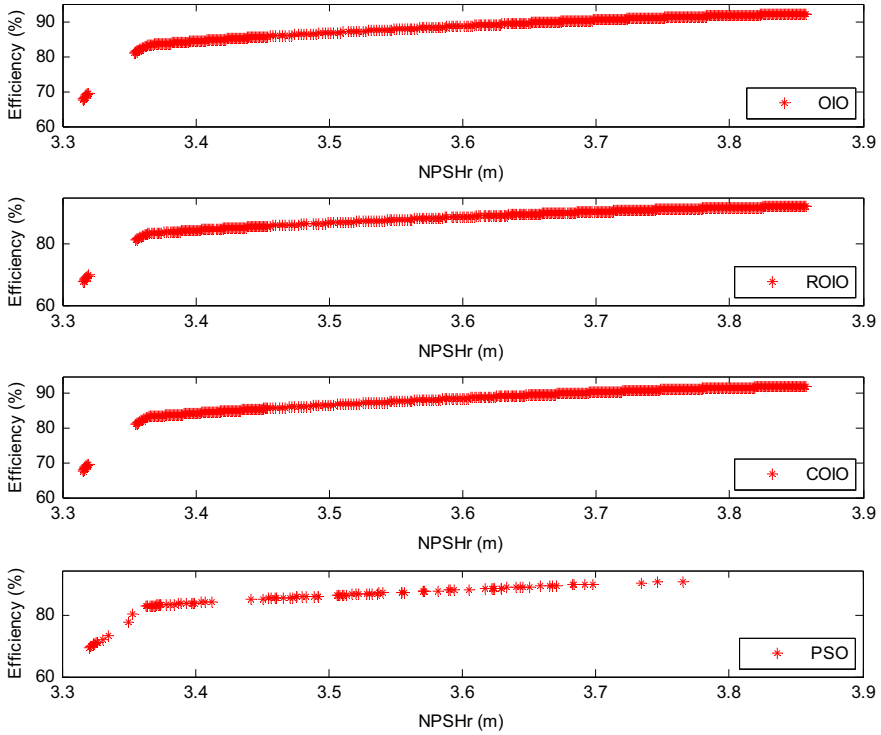

Fig. 12. Pareto optimal fronts obtained by different algorithms for bi-objective optimization of centrifuge pumps.

Table 20
The effect of spherical aberration correction on the performance of OIO.

Function	OIO without correction of spherical aberration			OIO with correction of spherical aberration		
	Mean ± St.dev of the best function values	Mean ± St.dev of the function evaluations	Success rate (out of 50 runs)	Mean ± St.dev of the best function values	Mean ± St.dev of the function evaluations	Success rate (out of 50 runs)
P22-Shekel 7	−2.713E−11 ± 1.477E−10	100000 ± 0	0	−10.402 ± 2.1E−4	24972 ± 4588	50

COIO will obtain a smooth Pareto front while PSO finds a shattered front.

From Results of Table 19 we can found that the Pareto fronts obtained by OIO, ROIO and COIO have been superimposed with the Pareto front of PSO. In terms of the diversity goal, which is considered as maximizing the extent of the obtained Pareto-set, this table expresses that only 629 solutions among 10,001 solutions obtained by PSO are non-dominated Pareto solutions. While this record for OIO, ROIO and COIO is 7251, 6068 and 9559, respectively which all of them are far greater than that is given by PSO.

Let P_{OIO} and P_{ROIO}, P_{COIO} and P_{PSO} be the set of Pareto-solutions obtained by algorithms. Let P_C be the set of non-dominated solutions obtained after combining P_{OIO} and P_{ROIO}, P_{COIO} and P_{PSO}. A quality measure of performance for OIO can be defined as $|P_{OIO} \cap P_C| / |P_C|$. The same measure can be obtained for other algorithms. Such a quality measure calculates the contribution of an algorithm in the construction of the final Pareto-set (P_C). Back to Table 19, we can see that the quality measure for PSO is only 0.57%. While it is 26.81% for OIO, 19.44% for ROIO and 53.17% for COIO. We can therefore infer that COIO performs as the best since it's contribution in the construction of the final Pareto front is more than 50%, while the contribution of PSO is about only 0.5%. The performance of COIO is also better than OIO and ROIO. As an evidence on the distinguished performance of COIO, it is enough to note that, 9553 solutions out of 9559 non-dominated Pareto solutions obtained by COIO contributes in the construction of P_C.

5.6. The effect of correcting artificial spherical aberration on the performance of OIO

In this last section we are interested to know whether we should take care of the correction of artificial spherical aberration in OIO or not. We have currently corrected the artificial spherical aberration in OIO through reducing the lateral aberration via increasing the artificial mirror radius of curvature (see Section 3.3). All computational experiments in the previous sections have been also carried out under the situation that the correction of spherical aberration had been allowed in OIO. To investigate whether the use of spherical aberration correction module has any impact on the performance of OIO or not, let us temporarily do not allow such a correction in OIO (that is we do not execute "Correction of spherical aberration" module in the flowchart of OIO). To measure the performance of OIO without spherical aberration correction module, we select Shekel 7 (P22) function from Table 2 for minimization over a four dimensional search space with decision variables ranged in$[0, 10^{+12}]$. We have compared the performance of OIO, with and without spherical aberration correction module and reported the results in Table 20. As can be seen from the result of this table, the correction of spherical aberration can have a significant impact on improving the performance of OIO and also on validating the optical rationale of the algorithm. When the spherical aberration is corrected in OIO, the algorithm is able to find the global minimum in each run with relatively few searches. But if we do not allow aberration correction in OIO, it performs very weak and is not able to find the true minimum in any run. Indeed, the final solutions obtained by the algorithm all lie far away from the true minimum. As an instant conclusion, the correction of spherical aberration is necessary for the algorithm since it improves the performance and helps the artificial mirror equations used in OIO (Eqs. (6) and (8)) be valid.

6. Conclusion

The optics inspired optimization (OIO) was introduced as a new nature inspired metaheuristic for numerical optimization. OIO is a population based search methodology that treats the surface of the numerical function to be optimized as a reflecting surface in which each peak is assumed to reflect as a convex mirror and each valley to reflect as a concave one. In each iteration, a number of artificial light points (individuals) are assumed to lie in front of the function surface and their artificially glittered ray is reflected back by the function surface, given that the surface is convex or concave, and the position of the artificial image is formed as a new solution in the search domain based on the mirror equations adopted from physics of optics. Extensive analyses were carried out to find out the merit of OIO and its two variants, ROIO and COIO, empirically. We considered four experiments including a large number of functions with different scales ranged from 2 to 500. Besides, a real world application in engineering design of a centrifuge pump was also considered. Our comparisons indicate that the proposed algorithms are generally able to perform the same or better than the best of rivals and exhibit a good convergence toward global minima. However, any statement on the convergence behavior should be based on the theoretical foundations or a formal proof of convergence.

OIO is in its infancy, and it is therefore the author's hope that this paper inspires future works on developing theory and practice of OIO. The performance of OIO can be further tested on the real world engineering optimization problems addressed in the literature. Besides, self-adaptive strategies or multiple populations could be employed in OIO as they are used in other evolutionary algorithms.

References

[1] Kennedy J, Eberhard RC. Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, Piscataway, NJ, USA, 1995, 1942–1948.
[2] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001;76:60–8.
[3] Passino KM. Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst Mag 2002;22:52–67.
[4] Karaboga D. An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department; 2005.
[5] Formato RA. Central force optimization: a new metaheuristic with applications in applied electromagnetics. Prog Electromagn Res PIER 2007;77:425–91.
[6] Yang XS. Firefly algorithm. In: Nature-inspired metaheuristic algorithms (chapter 8), Luniver Press; 2008.
[7] Husseinzadeh Kashan A. League championship algorithm: a new algorithm for numerical function optimization. In: Proceedings of SoCPaR 2009 IEEE international conference of soft computing and pattern recognition, 2009, 43-48.

[8] He S, Wu QH, Saunders JR. Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evol Comput 2009;13:973–90.
[9] Rao RV, Savsani VJ, Vakharia DP. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput Aided Des 2011;43:303–15.
[10] Gandomi AH, Alavi AH. Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci Numer Simul 2012;17:4831–45.
[11] Husseinzadeh Kashan A. An efficient algorithm for constrained global optimization and application to mechanical engineering design: league championship algorithm (LCA). Comput Aided Des 2011:1769–92.
[12] Husseinzadeh Kashan A. League championship algorithm (LCA): an algorithm for global optimization inspired by sport championships. Appl Soft Comput 2013.
[13] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Proceedings of IEEE congress on evolutionary computation, 2007, 4661-4666.
[14] Zitzewitz PW, Elliott TG, Haase DG, Harper KA, Herzog MR, Nelson JB, et al. Physics: principles and problems. Glencoe/McGraw-Hill; 2005.
[15] Jenkins FA, White HE. Fundamentals of optics. 4th ed. McGraw-Hill Book Company; 1976.
[16] Griffith WT, Brosing J. Physics of everyday phenomena. 7th ed. McGraw-Hill Higher Education; 2011.
[17] Konkar R. Analysing spherical aberration in concave mirrors. Reson J Sci Edu 2012;17:779–90.
[18] Bretscher O. Linear algebra with applications. 3rd ed. Pearson; 2005.
[19] Karaboga D, Akay B. A comparative study of Artificial Bee Colony algorithm. Appl Math Comput 2009;214:108–32.
[20] Rao V, Patel V. Comparative performance of an elitist teaching-learning-based optimization algorithm for solving unconstrained optimization problems. 2013, http://dx.doi.org/10.5267/j.ijiec.2012.09.001.
[21] Zhang M, Luo W, Wang X. Differential evolution with dynamic stochastic selection for constrained optimization. Inf Sci 2008;178:3043–74.
[22] Crepinsek M, Liu S-H, Mernik L. A note on teaching–learning-based optimization algorithm. Inf Sci 2012;212:79–93.
[23] Tamura K, Yasuda K. Primary study of spiral dynamics inspired optimization. IEEJ Trans Electr Electron Eng 2011;6:98–100.
[24] Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization. Comput Struct 2012;112-113:283–94.
[25] Suganthan PN, Hansen N, Liang JJ, Deb K, Chen YP, Auger A, et al. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report, Nanyang Technological University, Singapore, 2005, 2005, ⟨http://www.ntu.edu.sg/home/EPNSugan⟩.
[26] Liang JJ, Suganthan PN. Dynamic multi-swarm particle swarm optimizer with local search. In: Proceedings of IEEE congress on evolutionary computation, 2005, 522-528.
[27] Nourbakhsh A, Safikhani H, Derakhshan S. The comparison of multi-objective particle swarm optimization and NSGA II algorithm: applications in centrifugal pumps. Eng Optim 2011;43:1095–113.
[28] Muller SD, Marchetto J, Airaghi S, Koumoutsakos P. Optimization based on bacterial chemotaxis, IEEE transactions on evolutionary computation, 6, 2002, 16-29.
[29] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm. Inf Sci 2009;179:2232–48.