This paper presents a novel optimization model called hierarchical swarm optimization (HSO), which simulates the natural hierarchical complex system from where more complex intelligence can emerge for complex problems solving. This proposed model is intended to suggest ways that the performance of HSO-based algorithms on complex optimization problems can be significantly improved. This performance improvement is obtained by constructing the HSO hierarchies, which means that an agent in a higher level swarm can be composed of swarms of other agents from lower level and different swarms of different levels evolve on different spatiotemporal scale.

A novel optimization algorithm (named PS2O), based on the HSO model, is instantiated and tested to illustrate the ideas of HSO model clearly. Experiments were conducted on a set of 17 benchmark optimization problems including both continuous and discrete cases. The results demonstrate remarkable performance of the PS2O algorithm on all chosen benchmark functions when compared to several successful swarm intelligence and evolutionary algorithms.

1. Introduction

Swarm intelligence (SI), which is inspired by the “swarm behaviors” of social animals [1], is an innovative artificial intelligence technique for solving hard optimization problems. In SI system, there are many simple individuals who can interact locally with one another and with their environments. Although such systems are decentralized, local interactions between individuals lead to the emergence of global behaviors or global properties. For instance, flock of birds and school of fish emerge spatial self-organized patterns through social foraging [2]. Similar phenomena can also be observed in colonies of single-cell bacteria, social insects like ants and bees, as well as multicellular vertebrates, which all display collective intelligence [3].

As a problem-solving technique, many algorithmic methods of SI were designed to deal with practical problems. In 1991, Dorigo proposed ant colony optimization (ACO) [4, 5] based on foraging behaviors of ant colonies. ACO has been successfully used to solve discrete
optimization problems, like traveling salesman problems (TSP) [6]. After that, another SI algorithm, namely, particle swarm optimization (PSO), was proposed by Kennedy and Eberhart [7], which gleaned ideas from the social behavior of bird flocking and fish schooling [8–10]. PSO is primarily concerned with continuous optimization problems. In 2001, Fassino proposed a technique known as bacterial foraging optimization (BFO) that inspired by the pattern exhibited by bacterial foraging behaviors [11]. Other swarm optimization methods have been developed like artificial immune systems (AIS) [12], which are based on the metaphor of the immune system as a collective intelligence process [13]. Recently, Karaboga has described a bee swarm algorithm called artificial bee colony (ABC) algorithm [14], and Basturk and Karaboga compared the performance of ABC algorithm with the performance of genetic algorithm (GA) in [15]. These SI paradigms have already come to be widely used in many areas [8, 16–22].

In current artificial SI systems, however, researchers only take into account the collective behaviors of one level of individuals, and ignored the hierarchical nature [23] of the real world systems and animal society. In fact, for most social insects and animals, their organizational structures are not flat. They can form complex hierarchical (or multilevel) system structures by self-organization and division of labor features [24]. In other words, in a hierarchical system, a swarm of lower level individuals can be the infrastructure of a single individual at the higher level [25, 26]. Here the term “swarm” is used in a general sense to refer to any collection of interacting agents. In most of natural hierarchically complex systems, swarms of lower level agents interact with each other to constitute more complex high-level swarms’ constituent agents, repeatedly, until very complex structures with greatly enhanced macroscopical intelligence emerge. Such phenomenon is so common in the natural world, this guides us to design a multilevel algorithmic model to mimic hierarchical emergence of nature society.

First, this paper extends the traditional SI framework from flat (one level) to hierarchical (multiple level) by proposing a novel optimization model called hierarchical swarm optimization (HSO). In HSO model, collective behaviors of multiple levels are taken into account to solve complex problems. Then some initial insights into this method are provided by designing a two-level HSO algorithm (named PS\textsuperscript{2}O) based on canonical PSO model. Four versions of PS\textsuperscript{2}O are realized according to different structures of cooperation and interaction types in each level. In order to evaluate the performance of PS\textsuperscript{2}O, extensive studies based on a set of 17 benchmark functions (including both continuous and discrete cases) have been carried out. For comparison purposes, we also implemented the genetic algorithm (GA), covariance matrix adaptation evolution strategy (CMA-ES), artificial bee colony algorithm (ABC), and four state-of-the-art PSO variants on these functions. The experimental results are encouraging; the PS\textsuperscript{2}O algorithm achieved remarkable search performance in terms of accuracy, robustness, and convergence speed on all benchmark functions.

The rest of the paper is organized as follows. Section 2 describes the proposed hierarchical swarm optimization model. In Section 3, a novel HSO-based optimization algorithm, namely, PS\textsuperscript{2}O, is given. Section 4 tests the algorithm on the benchmarks and illustrates the results. Finally, Section 5 outlines the conclusions.

2. Hierarchical Swarm Optimization

2.1. From Flat Swarm to Hierarchical Swarm

In [3], Bonabeau et al. define swarm intelligence as “the emergent collective intelligence of groups of simple agents”. In such a perspective, the artificial SI systems, which are designed
for complex problem solving, maintain a swarm made up of many isomorphic and relatively simple individuals that often share the same states and behaviors set. In such a swarm, all individuals have absolutely equal status in the whole life cycle. The interaction relations between these individuals are symmetrical and operate on the same spatiotemporal scale. One individual can be substituted by another one, while the function of the swarm remains steady. That is, the architecture and functionality of classical SI are flat (Figure 1(a)).

However, swarm intelligence only explains partial mechanisms of collective behavior of biology. The natural cases could be more complex: except for individual tasks, these units lie at a hierarchical level between an individual ant and the colony as a whole, and thus constitute what might be called “intermediate-level parts” [27]. Now consider two basic types of systems: hierarchical and nonhierarchical (flat). Flat systems can be regarded as a group of undifferentiated particles, such as traditional SI systems. Hierarchical swarm systems must have a structure requiring a minimum of two hierarchical levels (Figure 1(b)). In Figure 1(b), a particle is the minimum unit of the system, while an agent constitutes the intermediate hierarchical level, which is composed of a number of particles. In this perspective, it is evident that the complexity of the individual “agents” of SI systems is dramatically simplified to particles, which are the minimum unit of the systems. Hence, the hierarchical nature of swarm [23] is ignored in the traditional artificial SI systems, such as PSO and ACO.

Hierarchy is common in the real world. For examples, immune system antibodies continuously self-organize and evolve while being a part of the many “organism agents” of a bird, and that a bird is in turn an agent in the formation of a flock of birds, and the flock of birds is in turn an agent that is part of a particular ecosystem niche [28]. Genes, the elementary biochemical coding units are complicated macromolecular strings, as are the metabolic units, the proteins. Neurons, the basic elements of cognitive networks, themselves are cells. In any of these examples, it is evident that the interactions of the agents lead to a coherent structure at a higher level [29]. That is, the emergent characteristics of a particular lower level system frequently form an individual agent at a higher level of the hierarchical system. This aspect has been emphasized by many researchers on artificial intelligence and complex systems [23, 25, 29–32].

Hence, this paper strives to extend the traditional SI framework from flat to hierarchical, and propose the hierarchical swarm optimization model. By incorporating
these new degrees of complexity, HSO-based optimization algorithm can accommodate a considerable potential for solving more complex problems.

2.2. HSO Model Description

HSO model accommodates a hierarchical multiagent system, in which an agent can itself be a swarm of other agents.

1. HSO is composed of a number of levels. Each level is a multiagent system composed of several swarms of agents.

2. Each swarm of level $n - 1$ agents is aggregated into a level-$n$ agent.

3. Level-$n$ behavior emerges from the organization of level 1 to $n$.

HSO naturally admits of a description in terms of higher level and lower level, where the lower level is nested within the higher level. Any agent at any level is both a component of a given swarm in its own level and a subsystem decomposable into a swarm of other agents at its adjacent lower level of HSO (shown as in Figure 2). Note that the agents in the lowest level are the particles that are the minimum unit, which are indecomposable of this hierarchical system. HSO is a heterogeneous system that each swarm in each level is evolved in its own population and adapts to the environment through the application of any SI method at hand. The interaction topology of HSO can also be heterogeneous hierarchical structures. Namely, the evolution rules and the interaction topology of distinct swarms can be different, and these different SI paradigms hierarchically construct the HSO model and lead to the hierarchical emergence of intelligence. In mathematical terms, the HSO model can be defined as in Table 1.

Figure 3 lists a general description of HSO containing four main functional blocks. In the first block of Figure 3, we show that under the external environment pressure (defined by the object function), each agent in the HSO model evolves and adapts as a consequence of internal and external hierarchical interactions. Both in higher level and lower level, the swarms can be manipulated by different SI algorithms (shown as in blocks 2 and 3 of Figure 3). In principle, any SI algorithms (such as PSO, ACO, BFO, and ABC) can be used.
Table 1: The structure of HSO model.

HSO = (A, N, P, T, O, S)
A: agents that exist at various hierarchical levels.
N: the number of levels.
P: the populations of each swarm in each level.
T: the hierarchical interaction topology of HSO.
O: the objective optimization goals.
S: Swarm or Evolutionary optimization strategies used for each swarm to search the objective O.

Figure 3: The main functional blocks of the HSO model.

by any swarm at any level, and we have first hand experience constructing HSO paradigms using PSO and BFO [33–35]. Interactions that occur within one level (each entity of the interaction is operating on the same spatiotemporal scale) are called symmetrical relations.

On the other hand, asymmetrical relationships occurring between different levels are called “constraints” [32]. The forth block formed the constraint that higher level affects elements from lower level. When an agent of higher level transmits the information to its constituent swarms of other agents of lower level, the effect can be the according evolutional actions of this agent’s swarm of constituent agents.
3. Case Study: The PS^2O Algorithm

In this section, we implement a simple two-level HSO algorithm, which employs PSO method in each swarm of each level, and hence named it PS^2O. Here the agents (particles) in the lower level (level-1) are analogous to individuals in a biological population (species) and the agents in the higher level (level-2) are analogous to species. As the hierarchical interactions that occur in the real ecosystems, from the macro view, dissimilar species establish symbiotic relationships to improve their survivability in level-1 of PS^2O (i.e., interspecies cooperation); from the micro view, species’ members (the particles) cooperatively interact with each other in level-2 of PS^2O (i.e., intraspecies cooperation).

3.1. Levels Detail of PS^2O

Here the basic goals are to find the minimum of \( f(\vec{x}) \), \( \vec{x} \in \mathbb{R}^D \). We create an ecosystem in level-1 that contains a species set \( \Omega = \{S_1, S_2, \ldots, S_M\} \), and each species \( k \) possesses a member set, \( S_k = \{X_{1k}, X_{2k}, \ldots, X_{N_k}\} \), in level-2. The \( i \)th member of the \( k \)th species is characterized by the vector \( X_{ik} = (X_{1ik}^1, X_{1ik}^2, \ldots, X_{1ik}^D) \). In each generation \( t \), the evolution process of each level is detailed as follow.

3.1.1. Level 1

Level-1 agents are clustered into \( M \) swarms, each of which possesses \( N \) agents. Each swarm constitutes an agent of level-2. Each swarm of level-1 evolves within its own separate population via separate PSO algorithm. That is, there are \( M \) parallel PSO paradigms evolving separately in level-1. This process addresses the cooperation between individuals of the same species: within the species \( k \), one or more members in the neighborhood of \( X_{ik} \) contribute their experiments to \( X_{ik} \), and \( X_{ik} \) also share its knowledge with its neighbors. Then \( X_{ik} \) accelerates towards its personal best position and the best position found by its species members in neighborhood:

\[
\alpha_{ik}^{t+1} = c_1 r_1 (p_{best_{ik}}^t - X_{ik}^t) + c_2 r_2 (s_{best_{k}}^t - X_{ik}^t),
\]

where \( \alpha_{ik} \) is the social acceleration vector of \( X_{ik} \), \( p_{best_{ik}} \) is the personal best position found so far by \( X_{ik} \), \( s_{best_{k}} \) is the best position found so far by its neighbors within species \( k \), \( c_1 \) are individual learning rate, \( c_2 \) are social learning rate, and \( r_1, r_2 \in \mathbb{R}^d \) are two random vectors uniformly distributed in \([0, 1]\).

3.1.2. Level 2

All level-2 agents aggregate into a single swarm. This swarm of distinct symbiotic species coevolves via the social only version of the PSO [36] as the cognitive processes have already taken care of by the level-1 swarms. From the coevolution perspective, the species \( k \) accelerates towards the best position that the symbiotic partners of species \( k \) have found:

\[
p_{k}^{t+1} = c_3 r_3 (c_{best}^t - S_{k}^t),
\]

where \( c_{best} \) is the best position found by all species.
where $\beta_k$ is the symbiotic acceleration vector of $S_k$, $\textit{cbest}$ is the best position found so far by the symbiotic partners of the $k$th species, $c_3$ is the “symbiotic learning rate”, and $r_3 \in \mathbb{R}^d$ is a uniform random sequence in the range $[0, 1]$.

### 3.1.3. Constraints

When species $k$ in level-2 accelerates towards the best position, $\textit{cbest}$, found by its more successful symbiotic partners, the according evolutional action of this agent’s swarm of constituent agents from level-1 is that all the members of species $k$ accelerate to $\textit{cbest}$ too

$$\beta_{ik}^{t+1} = c_3 r_3 (\textit{cbest}^t - X_{ik}^t),$$

(3.3)

where $\beta_{ik}$ is the symbiotic acceleration vector of $X_{ik}$.

Then the velocity $V_{ik}$ and position $X_{ik}$ of each member of species $k$ are updated according to

$$V_{ik}^{t+1} = \chi (V_{ik}^t + \alpha_{ik}^{t+1} + \beta_{ik}^{t+1}),$$

$$X_{ik}^{t+1} = X_{ik}^t + V_{ik}^{t+1},$$

(3.4)

where $\chi$ is known as the constriction coefficient [37].

### 3.2. Hierarchical Interaction Topologies

Systems of interacting agents—like many natural and social systems—are typically depicted by scientists as the graphs or networks, in which individuals can be connected to one another according to a great number of schemes [38]. In PSO, since the original particle swarm model is a simulation of the social environment, a neighborhood that structured as the interaction topological graph is defined for an individual particle as the subset of particles it is able to communicate with. Four classical interaction topologies have been shown as in Figure 4.

Most particle swarm implementations use one of two simple interaction topologies. The first, namely, the fully-connected topology (see Figure 4(a)), conceptually connects all members of the population to one another. The effect of this topology is that each particle is influenced by the very best performance of any member of the entire population. This means faster convergence, which implies a higher risk to converge to a local minimum. Experiments show that the fully-connected topology is faster than the other neighborhoods, but it meets the optimal fewer times than any other one. The second, called ring topology (see Figure 4(b)), creates a neighborhood for each individual comprising itself and its two nearest neighbors in the population. The ring neighborhood is more robust if the maximum number of iterations was increased but much slower. However, experiments show that the ring neighborhood cannot meet the required precision for many complex problems. That is, it promotes the exploration, but unfortunately fails to provide the exploitation.

In our model, the interaction of agents occurred in a two-level hierarchical topology. By employing two simple topologies—the ring and the fully-connected topologies—for swarms in different levels, four hierarchically nested interaction topologies have been obtained.
Figure 4: Four interaction topologies for PSO: (a) fully-connected, (b) ring, (c) star, (d) grid.

Shown as in Figure 5, each hierarchical topology is comprised of 4 warms in level-2 and each swarm possesses 4 agents from level-1. The first two topologies have a homogeneous hierarchical structure (employ the ring or fully-connected topology in both levels) and the other two have the heterogeneous hierarchical structures (employ the ring and fully-connected topologies in different levels, resp.). Four variant versions of the PS^2O algorithms are studied, respectively, in this paper according to these four interaction topologies.

(i) PS^2O-S: in level-1, agents interact with each other in each swarm. In level-2, each agent is influenced by the performance of all the other agents. That is, swarms of both levels are configured into the fully-connected topology (Figure 5(a)).

(ii) PS^2O-R: in level-1, agents interact with 2 immediate agents in its neighborhood. In level-2, each agent is influenced by the performance of its two symbiotic partners only. That is, both levels are configured into the ring topology (Figure 5(b)).

(iii) PS^2O-S_R: In level-1, agents interact with each other in each swarm. In level-2, each agent is influenced by the performance of its two symbiotic partners only. That is, the level-2 is configured into the fully-connected topology while the each swarm of level-1 is configured into the ring topology (Figure 5(c)).

(iv) PS^2O-R_S: In level-1, each agent interacts with 2 immediate agents in its neighborhood. In level-2, each agent is influenced by the performance of all the other agents. That is, each swarm of the level-1 is configured into the ring topology while the level-2 is configured into the fully-connected topology (Figure 5(d)).
3.3. Matrix Representation

A multidimensional array representation of the PS^2O algorithm is proposed in this section. The PS^2O randomly initializes a number of $M$ species with each possesses a number of $N$ members to represent the biological community in the natural ecosystems. Then the positions $X$, velocities $V$, and personal best locations $P$ of the biological community are all specified as the three-dimensional (3D) matrixes (showed as in Figures 6(a)–6(c)), where the first matrix dimension—Species number—is the number of species in level-2, the second matrix dimension—Swarm size—is the number of agents of each swarm in level-1, and the third matrix dimension—Dimension—is the number of dimensions of the object problem.

In PS^2O model, in order to update the velocity and position matrixes, every agent in level-1 must accelerate to three factors: the previous best position of the agent itself (this factor is called “personal best”), the previous best position of other members in its neighborhood (we named this factor “species best”), and the previous best position found by other species (agents from level-2) that have the cooperative symbiotic relation to the species that this agent belongs to (we named this factor “community best”). The species best is represented by a 2D matrix $S$, which showed as in Figure 6(d) left, and the community best is referred to as a 1D matrix $C$, which showed as in Figure 6(d) right.
Figure 6: Matrix representation of the PS²O algorithm.

$X, V, P, S$, and $C$ matrixes together record all of the update information required by the PS²O algorithm. These 3D matrixes are elegantly updated in successive iteration to numerically model the hierarchical emergence. The velocity and position matrixes must be
updated element by element in each generation as:

\[
V_{ijk}^{t+1} = V_{ijk}^{t} + \varphi_1 r_1 \left( P_{ijk}^{t} - X_{ijk}^{t} \right) + \varphi_2 r_2 \left( S_{ijk}^{t} - X_{ijk}^{t} \right) + \varphi_3 r_3 \left( C_j^t - X_{ijk}^{t} \right),
\]

\[
X_{ijk}^{t+1} = X_{ijk}^{t} + V_{ijk}^{t+1}
\]

to obtain the intended behaviors. Note that these equations are exactly described in the previous section: the term \(\varphi_1 r_1 \left( P_{ijk}^{t} - X_{ijk}^{t} \right)\) associates with each individual’s own cognition, the term \(\varphi_2 r_2 \left( S_{ijk}^{t} - X_{ijk}^{t} \right)\) associates with cooperative coevolution within each swarm in level-1, and the term \(\varphi_3 r_3 \left( C_j^t - X_{ijk}^{t} \right)\) associates with the symbiotic coevolution between dissimilar species in level-2.

The main difference between PS\(^2\)O and PSO is the matrix implementation and the modified velocity updating equation, that is, the complexity of this new HSO algorithm is similar to the original PSO. The flowchart of the PS\(^2\)O algorithm is presented in Figure 7, and according variables used in PS\(^2\)O are summarized in Table 2.

## 4. Experimental Result and Discussion

In experimental studies, according to the no free lunch (NFL) theorem [39], a set of 17 benchmark functions (with continuous and discrete characters), which are listed in the appendix, was employed to fully evaluate the performance of the PS\(^2\)O algorithm without a biased conclusion towards some chosen problems.

### 4.1. Experimental Setting

Experiments were conducted with four variations of PS\(^2\)O (PS\(^2\)Os) according to the four hierarchical interaction topologies. To fully evaluate the performance of the proposed PS\(^2\)O, seven successful EA and SI algorithms were used for comparisons:

(i) canonical PSO with constriction factor (PSO) [37],
(ii) fully informed particle swarm (FIPS) [40],
(iii) unified particle swarm (UPSO) [41],
(iv) fitness-Distance-Ratio-based PSO (FDR-PSO) [42],
(v) standard genetic algorithm (GA) [43],
(vi) covariance matrix adaptation evolution strategy (CMA-ES) [44],
(vii) artificial bee colony algorithm (ABC) [15].

Among these optimization tools, GA is the classical search technique that enables the fittest candidate among discrete strings to survive and reproduce based on random information search and exchange imitating the natural biological selection; the underlying idea of
Table 2: Parameters of the PS^2O.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>The number of agents (species) in level-2</td>
</tr>
<tr>
<td>N</td>
<td>Population size of each species in level-1</td>
</tr>
<tr>
<td>k</td>
<td>Species’ ID counter from 1 to M</td>
</tr>
<tr>
<td>i</td>
<td>Individual’s ID counter from 1 to N</td>
</tr>
<tr>
<td>j</td>
<td>Dimension counter from 1 to D</td>
</tr>
<tr>
<td>t</td>
<td>Generation counter from 1 to max-gen</td>
</tr>
<tr>
<td>X_{ijk}</td>
<td>The i\textsuperscript{th} individual’s (of the k\textsuperscript{th} species) j\textsuperscript{th} dimension’s position value</td>
</tr>
<tr>
<td>V_{ijk}</td>
<td>The i\textsuperscript{th} individual’s (of the k\textsuperscript{th} species) j\textsuperscript{th} dimension’s velocity value</td>
</tr>
<tr>
<td>P_{ijk}</td>
<td>The j\textsuperscript{th} dimension value of the i\textsuperscript{th} individual’s (of the k\textsuperscript{th} species) personal best position</td>
</tr>
<tr>
<td>S_{jk}</td>
<td>The j\textsuperscript{th} dimension position value of the best position found by the k\textsuperscript{th} level-2 species</td>
</tr>
<tr>
<td>C_{jk}</td>
<td>The j\textsuperscript{th} dimension value of the community best position</td>
</tr>
<tr>
<td>χ</td>
<td>The constriction coefficient</td>
</tr>
<tr>
<td>c_{1}</td>
<td>The learning rates for individual cognition</td>
</tr>
<tr>
<td>c_{2}</td>
<td>The learning rates for intraspecies cooperation</td>
</tr>
<tr>
<td>c_{3}</td>
<td>The learning rates for interspecies coevolution</td>
</tr>
</tbody>
</table>

CMA-ES is to gather information about successful search steps, and to use that information to modify the covariance matrix of the mutation distribution in a goal-directed, derandomized fashion; ABC is a recently developed SI paradigm simulating foraging behavior of bees; UPSO combined the global version and local version PSO together to construct a unified particle swarm optimizer; FIPS used all the neighbors’ knowledge of the particle to update the velocity; when updating each velocity dimension, the FDR-PSO selects one other particle \textit{nbest}, which has a higher fitness value and is nearer to the particle being updated.

In all experiments in this section, the values of the common parameters used in each algorithm such as population size and total generation number were chosen to be the same. Population size was 150 and the maximum evaluation number was 10000 for continuous functions and 1000 for discrete functions.

According to Clerc’s method [37], when constriction factor is implemented as in the canonical PSO algorithm, χ is calculated from the values of the acceleration coefficients (i.e., the learning rate) \( c_{1} \) and \( c_{2} \); importantly, it is the sum of these two coefficients that determines what χ to use. This fact implies that the particle’s velocity can be adjusted by any number of terms, as long as the acceleration coefficients sum to an appropriate value. Thus, the constriction factor χ in velocity formula of PS^2O can be calculated by

\[
\chi = \frac{2}{2 - \phi - \sqrt{\phi^2 - 4\phi}},
\]

where \( \phi = c_{1} + c_{2} + c_{3} \), \( \phi > 4 \). Then the algorithm will behave properly, at least as far as its convergence and explosion characteristics, whether all of \( \phi \) is allocated to one term, or it is divided into thirds, fourths, and so forth. Hence, for each PS^2O, except when different interaction topologies are used, the parameters were set to the values \( c_{1} = c_{2} = c_{3} = 1.3667 \) (i.e., \( \phi = c_{1} + c_{2} + c_{3} \approx 4.1 > 4 \)) and then \( \chi = 0.729 \), which is calculated by (4.1).
Initialize $M$ species each possess $N$ individuals. set $t = 0$

$k = k + 1$

$i = i + 1$

$j = j + 1$

$j < D$

$pbest_k = X_k$

Fit ($X_k$) $>$ Fit (pbest$_k$)

Fit ($X_k$) $>$ Fit (sbest$_k$)

sbest$_k$ = $X_k$

Yes

$k = k + 1$

$c_{best} = S_k$

$i < N$

Fit ($S_k$) $>$ Fit (cbest)

No

$k < M$

$t < max_gen$

Yes

$t = t + 1$

No

End

Figure 7: Flowchart of the PS$^2$O algorithm.

All the control parameters for the other algorithms were set to be default of their original literatures. In continuous optimization experiment, for CMA-ES, initialization conditions are the same as in [44], and the number of offspring candidate solutions generated per time step is $\lambda = 4\mu$; for ABC, the limit parameter is set to be $SN \times D$, where $D$ is the dimension of the problem and $SN$ is the number of employed bees; for canonical PSO and UPSO, the learning rates $c_1$ and $c_2$ were both 2.05 and the constriction factor $\chi = 0.729$; for FIPS, the constriction factor $\chi$ equals to 0.729 and the U-ring topology that achieved highest success rate is used; for FDR-PSO, the inertia weight $\omega$ started at 0.9 and ended at 0.5 and a setting of $c_1 = c_2 = 2.0$ was adopted.
Since there are no literatures using CMA-ES, ABC, UPSO, FIPS, and FDR-PSO for discrete optimization so far, discrete optimization experiment just compares PS2Os with the binary version of canonical PSO and standard GA in discrete cases. For GA, single-point crossover operation with the rate of 0.8 was employed and mutation rate was set to be 0.01. For discrete PSO, the parameters were set to the values \(c_1 = c_2 = 2\) and \(\chi = 1\). For PS2O variants, the parameters were set to the values \(c_1 = c_2 = c_3 = 2\) and \(\chi = 1\). The sigmoid function \(S\) was used as the transfer function to discrete the position \(X\) of PSO and PS2O variants \[45\]. Then the velocity update equation remains unchanged, while the position update equation is defined by the following equation (4.2) for discrete problems:

\[
\text{if} \left( \text{rand}() < S\left(V_{ijk}^{t+1}\right) \right), \quad X_{ijk}^{t+1} = 1, \quad \text{else} \quad X_{ijk}^{t+1} = 0. \tag{4.2}
\]

The number of agents (species) in level-2 (i.e., swarm number \(M\) of the level-2 swarm) needs be tuned. Six benchmark functions—Sphere 10D, Rosenbrock 10D, Rastrigrin 10D, Goldberg 120D, Bipolar 60D, and Discrete multimodal problem 100D—are used to investigate the impact of this parameter. Experiments were executed with PS2O-R on Sphere, PS2O-SR on Rosenbrock, PS2O-S on Rastrigrin, PS2O-R on Goldberg, PS2O-Rs on Bipolar, and PS2O-S on Discrete multimodal problem by changing the number of swarms and fixing each swarm size \(N\) at 10. The average test results obtained from 30 runs are plot in Figure 8. For continuous problems, the performance measurement is the average best-so-far fitness value; while for discrete cases, the performance measurement is the mean iteration to the function minimum 0. From Figure 8, we can observe that the performance of PS2Os is sensitive to the number of agents in level-2. When \(M\) increased, we obtained faster convergence velocity.
and better results on all test functions. However, it can be observed that the performance improvement is not evident when $M > 15$ for most test functions. Thus, in our experiments, the parameter $M$ of PS$^2$Os is set at 15 for all test functions (i.e., each swarm of level-2 possesses $N = 100/10 = 10$ agents of level 1).

The experiment runs 30 times, respectively, for each algorithm on each benchmark function. The numbers of generations for the 10 continuous benchmark functions were set to be 10000 and for the 7 discrete functions were 1000, respectively.

### 4.2. Continuous Unimodal Functions

Unimodal problems have been adopted to assess the convergence rates of optimization algorithms. We test the four PS$^2$O variants on a set of unimodal functions ($f_1$–$f_5$) in comparison with CMA-ES, ABC, PSO, FIPS, UPSO, and FDR-PSO algorithms. Table 3 lists the experimental results (i.e., the mean and standard deviations of the function values found in 30 runs) for each algorithm on $f_1$–$f_5$. Figure 9 shows the search progress of the average values found by the eight algorithms over 30 runs for $f_1$–$f_5$.

From Table 3 and Figure 9, the four PS$^2$O variants converged much faster to significantly better results than all other algorithms. The PS$^2$O-RS, which has the heterogeneous hierarchical structures, is the fastest one for finding good results within relatively few generations. All PS$^2$O variants were able to consistently find the minimum to functions $f_1$, $f_4$, and $f_5$ within 10000 generations.

From the comparisons between PS$^2$O and other algorithms, we can see that, statistically, PS$^2$O has significantly better performance on continuous unimodal functions $f_1$–$f_5$. From the rank values presented in Table 2, the search performance of the algorithms tested here is ordered as PS$^2$O-S > PS$^2$O-R > CMA-ES > PS$^2$O-SR > PS$^2$O-RS > UPSO > FDR-PSO > ABC > FIPS > PSO.

### 4.3. Continuous Multimodal Functions

The first four multimodal functions $f_6$–$f_9$ are regarded as the most difficult functions to optimize since the number of local minima increases exponentially as the function dimension increases. According to the results reported in [22], the methods CL-PSO, PSO, CMA-ES, G3-PCX, DE, and the algorithms used for comparison all failed to find the minimum of the six composition function designed by Liang. Since these mentioned methods have demonstrated their excellent performance on standard benchmark functions, the six composition functions are very complex. In this paper, we only test PS$^2$O on the first composition function $f_{10}$ and the test on the other five composition functions will be studied in the future works. The mean and standard deviations of the function values found in 30 runs for each algorithm on each function are listed in Table 4. Figure 10 shows the search progress of the average values found by the ten algorithms over 30 runs for functions $f_6$–$f_{10}$.

From Table 4 and Figure 10, it is clear to see that for most of the tested continuous benchmark functions, all PS$^2$O algorithms except PS$^2$O-RS markedly outperformed the other algorithms. For example, PS$^2$O-R and PS$^2$O-SR found the global minimum every time of run on function $f_8$–$f_{10}$, and PS$^2$O-R can also consistently found the minimum of $f_{10}$ within relatively fewer generations, while the other algorithms generated poorer results on them. On functions $f_6$ and $f_7$, the four PS$^2$O algorithms yielded similar results to the other four algorithms. From the rank values presented in Table 4, the search performance of the algorithms tested here is ordered as PS$^2$O-R > PS$^2$O-SR > ABC > PS$^2$O-S > FIPS CMA-ES > FDR-PSO > PSO > UPSO > PSO > PS$^2$O-RS.
Figure 9: The median convergence results of 30D unimodal continuous functions. (a) Sphere function. (b) Rosenbrock’s function. (c) Quadric function. (d) Sum of different powers. (e) Sin function.
Table 3: Performance of all algorithms on benchmark functions $f_1$–$f_5$. In bold are the best results.

<table>
<thead>
<tr>
<th>Func.</th>
<th>PS\textsuperscript{2}O-S</th>
<th>PS\textsuperscript{2}O-R</th>
<th>PS\textsuperscript{2}O-S\textsubscript{R}</th>
<th>PS\textsuperscript{2}O-R\textsubscript{S}</th>
<th>PSO</th>
<th>FIPS</th>
<th>UPSO</th>
<th>FDR-PSO</th>
<th>CMA-ES</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0</td>
<td>0</td>
<td>6.24e – 305</td>
<td>0</td>
<td>2.42e – 114</td>
<td>1.73e – 029</td>
<td>3.71e – 183</td>
<td>2.47e – 169</td>
<td>0</td>
<td>5.52e – 016</td>
</tr>
<tr>
<td>$f_1$ Std</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3.39e – 114</td>
<td>2.29e – 029</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8.47e – 017</td>
<td></td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>1</td>
<td>7</td>
<td>8</td>
<td>6</td>
<td>5</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>Mean</td>
<td>1.32e – 001</td>
<td>1.04e – 014</td>
<td>2.55e – 016</td>
<td>3.71e – 026</td>
<td>6.61e + 000</td>
<td>2.25e + 001</td>
<td>2.09e + 000</td>
<td>2.79e – 001</td>
<td>7.91e – 029</td>
<td>8.80e – 002</td>
</tr>
<tr>
<td>$f_2$ Std</td>
<td>7.27e – 001</td>
<td>9.71e – 015</td>
<td>1.70e – 016</td>
<td>1.09e – 025</td>
<td>4.02e – 001</td>
<td>1.27e + 000</td>
<td>7.69e – 001</td>
<td>1.02e + 000</td>
<td>1.77e – 029</td>
<td>7.98e – 002</td>
</tr>
<tr>
<td>Rank</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>9</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Mean</td>
<td>1.04e – 090</td>
<td>2.19e – 047</td>
<td>1.42e – 051</td>
<td>2.50e – 092</td>
<td>3.51e – 018</td>
<td>1.38e + 000</td>
<td>6.12e – 013</td>
<td>4.25e – 014</td>
<td>3.95e – 235</td>
<td>1.16e + 001</td>
</tr>
<tr>
<td>$f_3$ Std</td>
<td>5.01e – 090</td>
<td>6.82e – 047</td>
<td>4.28e – 051</td>
<td>1.02e – 091</td>
<td>7.88e – 018</td>
<td>1.00e + 000</td>
<td>6.06e – 013</td>
<td>2.32e – 013</td>
<td>0</td>
<td>4.57e + 000</td>
</tr>
<tr>
<td>Rank</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>10</td>
<td>8</td>
<td>7</td>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>Mean</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8.23e – 320</td>
<td>4.54e – 185</td>
<td>5.03e – 065</td>
<td>0</td>
<td>8.64e – 280</td>
<td>9.48e – 006</td>
<td>7.75e – 006</td>
</tr>
<tr>
<td>$f_4$ Std</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.38e – 064</td>
<td>0</td>
<td>0</td>
<td>1.27e – 005</td>
<td>8.50e – 006</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>10</td>
<td>9</td>
</tr>
<tr>
<td>Mean</td>
<td>0</td>
<td>0</td>
<td>1.38e – 002</td>
<td>1.14e – 001</td>
<td>4.84e – 002</td>
<td>1.38e – 002</td>
<td>0</td>
<td>3.50e – 003</td>
<td>9.28e – 032</td>
<td>5.68e – 016</td>
</tr>
<tr>
<td>$f_5$ Std</td>
<td>0</td>
<td>0</td>
<td>3.58e – 002</td>
<td>2.80e – 001</td>
<td>8.04e – 002</td>
<td>4.50e – 002</td>
<td>0</td>
<td>1.89e – 002</td>
<td>6.57e – 033</td>
<td>8.19e – 017</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Aver. Rank</td>
<td>24</td>
<td>2.4</td>
<td>3.8</td>
<td>4</td>
<td>9.5</td>
<td>8.8</td>
<td>4.8</td>
<td>6.2</td>
<td>3.4</td>
<td>7.6</td>
</tr>
<tr>
<td>Final Rank</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>5</td>
<td>10</td>
<td>9</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>
Table 4: Performance of all algorithms on benchmark functions $f_6$–$f_{30}$. In bold are the best results.

<table>
<thead>
<tr>
<th>Func.</th>
<th>PS(^2)O-S</th>
<th>PS(^2)O-R</th>
<th>PS(^2)O-S(_R)</th>
<th>PS(^2)O-R(_S)</th>
<th>PSO</th>
<th>FIPS</th>
<th>UPSO</th>
<th>FDR-PSO</th>
<th>CMA-ES</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>7.11e-015</td>
<td>4.74e-015</td>
<td>4.97e-015</td>
<td>8.32e-002</td>
<td>6.16e-015</td>
<td>2.04e-014</td>
<td><strong>3.55e-015</strong></td>
<td>2.74e-014</td>
<td>1.96e+001</td>
<td>4.79e-014</td>
</tr>
<tr>
<td>Std</td>
<td>0</td>
<td>1.70e-015</td>
<td>1.77e-015</td>
<td>3.17e-002</td>
<td>1.80e-015</td>
<td>1.08e-014</td>
<td>3.16e-015</td>
<td>1.03e-014</td>
<td>4.30e-001</td>
<td>6.09e-015</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>3</td>
<td>4</td>
<td>9</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>7</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Func.</th>
<th>PS(^2)O-S</th>
<th>PS(^2)O-R</th>
<th>PS(^2)O-S(_R)</th>
<th>PS(^2)O-R(_S)</th>
<th>PSO</th>
<th>FIPS</th>
<th>UPSO</th>
<th>FDR-PSO</th>
<th>CMA-ES</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>8.37e-003</td>
<td>3.25e+001</td>
<td>3.61e+001</td>
<td>4.61e+001</td>
<td>2.78e+011</td>
<td>4.13e+001</td>
<td>4.96e+001</td>
<td>2.60e+001</td>
<td>7.56e+000</td>
<td><strong>1.64e-013</strong></td>
</tr>
<tr>
<td>Std</td>
<td>1.09e-002</td>
<td>8.41e+000</td>
<td>8.96e+000</td>
<td>1.11e+001</td>
<td>5.46e+000</td>
<td>2.34e+001</td>
<td>9.80e+000</td>
<td>8.32e+000</td>
<td>2.15e+000</td>
<td><strong>6.25e-014</strong></td>
</tr>
<tr>
<td>Rank</td>
<td>7</td>
<td>5</td>
<td>6</td>
<td>8</td>
<td>4</td>
<td>7</td>
<td>10</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Func.</th>
<th>PS(^2)O-S</th>
<th>PS(^2)O-R</th>
<th>PS(^2)O-S(_R)</th>
<th>PS(^2)O-R(_S)</th>
<th>PSO</th>
<th>FIPS</th>
<th>UPSO</th>
<th>FDR-PSO</th>
<th>CMA-ES</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>1.17e+000</td>
<td>0</td>
<td>0</td>
<td>9.26e-003</td>
<td>1.83e-002</td>
<td>1.56e-003</td>
<td>3.47e-003</td>
<td>1.79e-002</td>
<td>0</td>
<td>1.03e-015</td>
</tr>
<tr>
<td>Std</td>
<td>7.68e-001</td>
<td>0</td>
<td>0</td>
<td>1.07e-002</td>
<td>2.66e-002</td>
<td>3.78e-003</td>
<td>4.78e-003</td>
<td>1.82e-002</td>
<td>0</td>
<td>1.17e-015</td>
</tr>
<tr>
<td>Rank</td>
<td>6</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>10</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Func.</th>
<th>PS(^2)O-S</th>
<th>PS(^2)O-R</th>
<th>PS(^2)O-S(_R)</th>
<th>PS(^2)O-R(_S)</th>
<th>PSO</th>
<th>FIPS</th>
<th>UPSO</th>
<th>FDR-PSO</th>
<th>CMA-ES</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4.00e+001</td>
<td>5.00e+001</td>
<td>3.37e+001</td>
<td>8.00e+001</td>
<td>1.00e+002</td>
<td>1.30e+002</td>
<td>3.83e-002</td>
</tr>
<tr>
<td>Std</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8.94e+000</td>
<td>1.82e+002</td>
<td>2.58e+001</td>
<td>4.42e+001</td>
<td>4.42e+001</td>
<td>1.63e+002</td>
<td>1.21e-001</td>
</tr>
<tr>
<td>Rank</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Func.</th>
<th>PS(^2)O-S</th>
<th>PS(^2)O-R</th>
<th>PS(^2)O-S(_R)</th>
<th>PS(^2)O-R(_S)</th>
<th>PSO</th>
<th>FIPS</th>
<th>UPSO</th>
<th>FDR-PSO</th>
<th>CMA-ES</th>
<th>ABC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>4.8</td>
<td>2.2</td>
<td>2.6</td>
<td>7.6</td>
<td>6.8</td>
<td>5.4</td>
<td>7.2</td>
<td>6.6</td>
<td>6.4</td>
<td>3.6</td>
</tr>
<tr>
<td>Final Rank</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>8</td>
<td>5</td>
<td>9</td>
<td>7</td>
<td>6</td>
<td>3</td>
</tr>
</tbody>
</table>
Figure 10: The median convergence results of 30D multimodal continuous functions. (a) Ackley’s function. (b) Rastrigin’s function. (c) Griewank’s function. (d) Weierstrass function. (e) Composition function 1.
Figure 11: Continued.
4.4. Discrete Functions

In binary optimization, it is very easy to design some algorithms that are extremely good on some benchmarks (and extremely bad on some others) [46]. In order to fully evaluate the performance of PS$^2$O on discrete problems, we have employed a carefully chosen set of discrete benchmark functions $f_{11}^{-f_{17}}$. The results obtained by the GA, PSO, and four PS$^2$O algorithms on each discrete benchmark function are listed in Table 5, including the mean number of iterations required to reach the minimum, mean, and standard deviations of the function values found in 30 runs. Figure 11 shows the search progress of the average values found by the five algorithms over 30 runs for functions $f_{11}^{-f_{17}}$.

From the results, we can observe that PS$^2$O obtain an obviously remarkable performance. It can be seen from Figure 11 that all PS$^2$O variants converged greatly faster and to significantly better results than the other two algorithms for all discrete cases. From the rank values presented in Table 5, the search performance of the algorithms tested here is ordered as PS$^2$O-R > PS$^2$O-S > PS$^2$O-R$_S$ > PS$^2$O-S$_R$ > GA > PSO. It is worth mentioning that the PS$^2$O-R and PS$^2$O-R$_S$ were able to consistently find the minimum to all discrete benchmark functions.

5. Conclusion

This paper first describes the hierarchical swarm intelligence phenomenon: the emergence of high-level intelligence aggregates properties from low-level. This mechanism is so common in nature, and provides initial evidence of the potential problem solving capabilities.
Table 5: Performance of all algorithms on benchmark functions $f_{11}$–$f_{17}$. In bold are the best results.

<table>
<thead>
<tr>
<th>Func.</th>
<th>PS²O-S mean</th>
<th>PS²O-R mean</th>
<th>PS²O-SR mean</th>
<th>PS²O-RS mean</th>
<th>PSO mean</th>
<th>GA mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{11}$</td>
<td>0.67e + 002</td>
<td>2.60e + 001</td>
<td>2.60e + 001</td>
<td>3.19e + 000</td>
<td>3.19e + 000</td>
<td>2.64e + 001</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>0.60e + 000</td>
<td>1.33e + 000</td>
<td>1.33e + 000</td>
<td>2.94e + 001</td>
<td>2.94e + 001</td>
<td>2.38e + 001</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>0.60e + 000</td>
<td>1.30e + 000</td>
<td>1.30e + 000</td>
<td>2.00e + 003</td>
<td>2.00e + 003</td>
<td>2.00e + 003</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>0.60e + 000</td>
<td>1.41e + 000</td>
<td>1.41e + 000</td>
<td>4.00e + 000</td>
<td>4.00e + 000</td>
<td>3.80e + 000</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td>0.60e + 000</td>
<td>1.30e + 000</td>
<td>1.30e + 000</td>
<td>3.67e + 000</td>
<td>3.67e + 000</td>
<td>3.00e + 000</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td>0.60e + 000</td>
<td>1.83e + 000</td>
<td>1.83e + 000</td>
<td>2.03e + 000</td>
<td>2.03e + 000</td>
<td>2.00e + 000</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td>0.60e + 000</td>
<td>1.83e + 000</td>
<td>1.83e + 000</td>
<td>39</td>
<td>39</td>
<td>45</td>
</tr>
</tbody>
</table>

Furthermore, this paper presents the hierarchical swarm optimization (HSO) model that simulates the hierarchical swarm intelligence for function optimization. HSO is an artificial hierarchical complex system, in which agents are composed of swarms of other agents in nested hierarchies. That is, HSO is configured into several levels and each level is composed of a number of independent swarms. Note that any traditional SI methods or evolutionary algorithms can be used to manipulate any swarm of any level in HSO. HSO can be considered as not only an extension of the traditional SI model to design novel optimization algorithms, but also an open framework to hybrid traditional SI or EA algorithms to tackle hard optimization problems.

HSO model has a considerable potential in optimization domain. This paper provides some initial insights into this potential by designing a two-level HSO algorithm, namely...
PS\textsuperscript{2}O, which employs PSO method in each swarm of each level. This algorithm is conceptually simple, has low complexity, and is easy to implement. A set of 17 benchmark functions (including continuous and discrete cases) have been used to test four PS\textsuperscript{2}O variants in comparison with GA, CMA-ES, ABC, canonical PSO, FIPS, UPSO, and FDR-PSO. The simulation results show that, for all the test functions, the PS\textsuperscript{2}Os reach remarkably better performance in terms of accuracy, convergence rate, and robustness than the other classical powerful algorithms.

It should be mentioned that the PS\textsuperscript{2}Os were the only ones that are able to consistently find the minimum of Sphere, Sum of different powers, Griewank, Weierstrass, Composition function 1 and all discrete test problems. Future research will address designing more robust and efficient HSO algorithms by integrating other SI algorithms to HSO and applying them to solve complex optimization problems. We should note that the two-level HSO algorithm is conceptually simple and easy to implement. However, in our future work, HSO algorithms with more hierarchical levels could be studied and tested on some complex benchmark functions and real-world problems.

### Appendix

#### List of Test Functions

These benchmark functions can be grouped as unimodal continuous functions $f_{1-5}$, multimodal continuous functions $f_{6-10}$, unimodal discrete functions $f_{11-16}$, and multimodal discrete function $f_{17}$. Functions $f_{1-5}$ were test widely in evolutionary computation domain to show solution quality and convergence rate. Function $f_{10}$ is a novel composition benchmark function developed by Liang et al. [47]. The discrete functions $f_{11-17}$ were used in Clerc’s literature [46, 48] and can be found at http://clerc.maurice.free.fr/pso/.

1. **Sphere Function**

   $$f_1(x) = \sum_{i=1}^{n} x_i^2.$$  \hfill (A.1)

2. **Rosenbrock's Function**

   $$f_2(x) = \sum_{i=1}^{n} 100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2.$$  \hfill (A.2)

3. **Quadric Function**

   $$f_3(x) = \sum_{i=1}^{n} \left( \sum_{j=1}^{n} x_j \right)^2.$$  \hfill (A.3)

4. **Sum of Different Powers**

   $$f_4(x) = \sum_{i=1}^{n} |x_i|^{i+1}.$$  \hfill (A.4)
(5) Sin Function

\[ f_5(x) = \frac{\pi}{n} \left\{ 10 \sin^2 \pi x_1 + \sum_{i=1}^{n-1} (x_i - 1)^2 (1 + 10 \sin^2 \pi x_{i+1}) + (x_n - 1)^2 \right\}. \quad (A.5) \]

(6) Ackley’s Function

\[ f_6(x) = -20 \exp \left( -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos(2 \pi x_i) \right) + 20 + e. \quad (A.6) \]

(7) Rastrigrin’s Function

\[ f_7(x) = \sum_{i=1}^{n} \left( x_i^2 - 10 \cos(2 \pi x_i) \right) + 10. \quad (A.7) \]

(8) Griewank’s Function

\[ f_8(x) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1. \quad (A.8) \]

(9) Weierstrass Function

\[ f_9(x) = \sum_{i=1}^{n} \left( \sum_{k=0}^{k_{\text{max}}} \left[ a^k \cos \left( 2 \pi b^k (x_i + 0.5) \right) \right] - \left[ a^k \cos \left( 2 \pi b^k \cdot 0.5 \right) \right] \right). \quad (A.9) \]

where \( a = 0.5, b = 3 \), and \( k_{\text{max}} = 20 \).

(10) Composition Function 1

The composition function 1 is constructed using 10 unimodal sphere functions. This results in an asymmetrical multimodal function with 1 global optimum and 9 local optima. The variables of the formulation can be referred to \([47]\)

\[ f_{10}(x) = \sum_{i=1}^{n} \left\{ w_i \ast \left[ f_i \left( \frac{(x - o_i + \text{oold}_i)}{\lambda_i} \ast M_i \right) + \text{bias}_i \right] \right\} + f_{\text{bias}}. \quad (A.10) \]
Goldberg’s Order-3

The fitness \( f \) of a bit-string is the sum of the result of separately applying the following function to consecutive groups of three components each:

\[
f_{11}(x) = \begin{cases} 
0.9 & \text{if } |y| = 0, \\
0.6 & \text{if } |y| = 1, \\
0.3 & \text{if } |y| = 2, \\
1.0 & \text{if } |y| = 3.
\end{cases}
\] (A.11)

If the string size (the dimension of the problem) is \( D \), the maximum value is \( D/3 \) for the string 111...111. In practice, we will then use as fitness the value \( D/3 - f \) so that the problem is now to find the minimum 0.

Bipolar Order-6

The fitness \( f \) is the sum of the result of applying the following function to consecutive groups of six components each:

\[
f_{12}(x) = \begin{cases} 
1.0 & \text{if } |y| = 0 \text{ or } 6, \\
0.0 & \text{if } |y| = 1 \text{ or } 5, \\
0.4 & \text{if } |y| = 2 \text{ or } 4, \\
0.8 & \text{if } |y| = 3.
\end{cases}
\] (A.12)

The maximum value is \( D/6 \). In practice, we will use as fitness the value \( D/6 - f \) so that the problem is now to find the minimum 0.

Mulenbein’s Order-5

The fitness \( f \) is the sum of the result of applying the following function to consecutive groups of five components each:

\[
f_{13}(x) = \begin{cases} 
4.0 & \text{if } y = 00000, \\
3.0 & \text{if } y = 00001, \\
2.0 & \text{if } y = 00011, \\
1.0 & \text{if } y = 00111, \\
3.5 & \text{if } y = 11111, \\
0.0 & \text{otherwise}.
\end{cases}
\] (A.13)
The maximum value is $3.5D/5$. In practice, the value $3.5D/5 - f$ is used as the fitness so that the problem is now to find the minimum 0.

(14) Clerc's Zebra3

The fitness $f$ is the sum of the result of applying the following function to consecutive groups of three components each, if the rank of the group is even (first rank = 0):

$$
f_{14}(x) = \begin{cases}
0.9 & \text{if } |y| = 0, \\
0.6 & \text{if } |y| = 1, \\
0.3 & \text{if } |y| = 2, \\
1.0 & \text{if } |y| = 3.
\end{cases}
$$

(A.14)

If the rank of the group is odd,

$$
f_{14}(x) = \begin{cases}
0.9 & \text{if } |y| = 3, \\
0.6 & \text{if } |y| = 2, \\
0.3 & \text{if } |y| = 1, \\
1.0 & \text{if } |y| = 0.
\end{cases}
$$

(A.15)

The maximum value is $D/3$. In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.

(15) Clerc's Order-3 Problem 1

The fitness $f$ is the sum of the result of applying the following function to consecutive groups of three components each:

$$
f_{15}(x) = \begin{cases}
0.9 & \text{if } y = 000, \\
0.6 & \text{if } y = 001, \\
0.3 & \text{if } y = 010, \\
1.0 & \text{if } y = 011, \\
0.2 & \text{if } y = 100, \\
0.4 & \text{if } y = 101, \\
0.6 & \text{if } y = 110, \\
0.8 & \text{if } y = 111.
\end{cases}
$$

(A.16)

The maximum value is $D/3$. In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.
Table 6: Parameters of the test functions.

<table>
<thead>
<tr>
<th>$f$</th>
<th>Dimensions</th>
<th>Initial range</th>
<th>$x^*$</th>
<th>$f(x^*)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>30</td>
<td>$[-100, 100]^D$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_2$</td>
<td>30</td>
<td>$[-30, 30]^D$</td>
<td>$[1, 1, \ldots, 1]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_3$</td>
<td>30</td>
<td>$[-65.536, 65.536]^D$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_4$</td>
<td>30</td>
<td>$[-1, 1]^D$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_5$</td>
<td>30</td>
<td>$[-10, 10]^D$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_6$</td>
<td>30</td>
<td>$[-32.768, 32.768]^D$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_7$</td>
<td>30</td>
<td>$[-5.12, 5.12]^D$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_8$</td>
<td>30</td>
<td>$[-600, 600]^D$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_9$</td>
<td>30</td>
<td>$[-0.5, 0.5]^D$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_{10}$</td>
<td>10</td>
<td>$[-5, 5]^D$</td>
<td>Predefined rand number distributed in the search range</td>
<td>0</td>
</tr>
<tr>
<td>$f_{11}$</td>
<td>120</td>
<td>$[0, 1]$</td>
<td>$[1, 1, \ldots, 1]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_{12}$</td>
<td>120</td>
<td>$[0, 0, \ldots, 0]$ or $[1, 1, \ldots, 1]$ or $[\ldots, 6 \cdot 0, \ldots, 6 \cdot 1, \ldots]$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_{13}$</td>
<td>120</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_{14}$</td>
<td>30</td>
<td>$[0, 0, \ldots, 0]$ or $[1, 1, \ldots, 1]$ or $[\ldots, 6 \cdot 0, \ldots, 6 \cdot 1, \ldots]$</td>
<td>$[0, 0, \ldots, 0]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_{15}$</td>
<td>60</td>
<td>$[0, 1]$</td>
<td>$[0, 1, 0, 1, 1, \ldots]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_{16}$</td>
<td>60</td>
<td>$[0, 1]$</td>
<td>$[0, 0, 1, 0, 1, 1, \ldots]$</td>
<td>0</td>
</tr>
<tr>
<td>$f_{17}$</td>
<td>200</td>
<td>$[0, 1]$</td>
<td>Predefined rand number distributed in the search range</td>
<td>0</td>
</tr>
</tbody>
</table>

(16) Clerc’s Order-3 Problem 2

The fitness $f$ is the sum of the result of applying the following function to consecutive groups of three components each:

$$f_{16}(x) = \begin{cases} 
0.2 & \text{if } y = 000, \\
1.0 & \text{if } y = 001, \\
0.3 & \text{if } y = 010, \\
0.8 & \text{if } y = 011, \\
0.6 & \text{if } y = 100, \\
0.4 & \text{if } y = 101, \\
0.6 & \text{if } y = 110, \\
0.9 & \text{if } y = 111. 
\end{cases} \quad (A.17)$$

The maximum value is $D/3$. In practice, we will then use as fitness the value $D/3 - f$ so that the problem is now to find the minimum 0.
// Definition of the fitness landscape.
For (i = 1 to number of peaks p)
    For (j = 1 to number of Dimension D)
        landscape (i, j) = rand();
    End for
End for

// the fitness f is computed as follows:

f = 0;
For (i = 1 to number of peaks p)
    q = 0;
    For (j = 1 to number of Dimension D)
        If (x(j) = landscape (i, j))
            q = q + 1;
        End if
    End for
    If (f < q)
        f = q;
    End if
End for
f = 1 - f / D.

Algorithm 1

(17) Discrete Multimodal Problem

This problem is defined by Algorithm 1.

The dimensions, initialization ranges, global optima x*, and the corresponding fitness value f(x*) of each function are listed in Table 6.

References


Submit your manuscripts at http://www.hindawi.com