mathematics

Article

A Dimension Group-Based Comprehensive Elite Learning Swarm Optimizer for Large-Scale Optimization

Qiang Yang [1], Kai-Xuan Zhang [1], Xu-Dong Gao [1,*], Dong-Dong Xu [1], Zhen-Yu Lu [1], Sang-Woon Jeon [2] and Jun Zhang [2,3]

1 School of Artificial Intelligence, Nanjing University of Information Science and Technology, Nanjing 210044, China; qiang_yang@nuist.edu.cn (Q.Y.); kaixuan.zhang@nuist.edu.cn (K.-X.Z.); 001601@nuist.edu.cn (D.-D.X.); 001114@nuist.edu.cn (Z.-Y.L.)
2 Department of Electrical and Electronic Engineering, Hanyang University, Ansan 15588, Korea; sangwoonjeon@hanyang.ac.kr (S.-W.J.); junzhang@ieee.org (J.Z.)
3 Department of Computer Science and Information Engineering, Chaoyang University of Technology, Taichung 413310, Taiwan
* Correspondence: 003103@nuist.edu.cn

Abstract: High-dimensional optimization problems are more and more common in the era of big data and the Internet of things (IoT), which seriously challenge the optimization performance of existing optimizers. To solve these kinds of problems effectively, this paper devises a dimension group-based comprehensive elite learning swarm optimizer (DGCELSO) by integrating valuable evolutionary information in different elite particles in the swarm to guide the updating of inferior ones. Specifically, the swarm is first separated into two exclusive sets, namely the elite set (***ES***) containing the top best individuals, and the non-elite set (***NES***), consisting of the remaining individuals. Then, the dimensions of each particle in ***NES*** are randomly divided into several groups with equal sizes. Subsequently, each dimension group of each non-elite particle is guided by two different elites randomly selected from ***ES***. In this way, each non-elite particle in ***NES*** is comprehensively guided by multiple elite particles in ***ES***. Therefore, not only could high diversity be maintained, but fast convergence is also likely guaranteed. To alleviate the sensitivity of DGCELSO to the associated parameters, we further devise dynamic adjustment strategies to change the parameter settings during the evolution. With the above mechanisms, DGCELSO is expected to explore and exploit the solution space properly to find the optimum solutions for optimization problems. Extensive experiments conducted on two commonly used large-scale benchmark problem sets demonstrate that DGCELSO achieves highly competitive or even much better performance than several state-of-the-art large-scale optimizers.

Keywords: large-scale optimization; particle swarm optimization; dimension group-based comprehensive elite learning; high-dimensional problems; elite learning

MSC: 37N40; 46N10; 47N10

check for updates

Citation: Yang, Q.; Zhang, K.-X.; Gao, X.-D.; Xu, D.-D.; Lu, Z.-Y.; Jeon, S.-W.; Zhang, J. A Dimension Group-Based Comprehensive Elite Learning Swarm Optimizer for Large-Scale Optimization. *Mathematics* **2022**, *10*, 1072. https://doi.org/10.3390/math10071072

Academic Editors: Ioannis G. Tsoulos, Hsien-Chung Wu and Frank Werner

Received: 4 February 2022
Accepted: 24 March 2022
Published: 26 March 2022

1. Introduction

Large-scale optimization problems, also called high-dimensional problems, are ubiquitous in daily life and industrial engineering in the era of big data and the Internet of Things (IoT), such as water distribution optimization problems [1], cyber-physical systems design problems [2], control of pollutant spreading on social networks [3], and offshore wind farm collector system planning problems [4]. As the dimensionality of optimization problems increases, most existing optimization methods encounter the degradation of optimization effectiveness, due to the "curse of dimensionality" [5,6].

Specifically, the increase of dimensionality results in the following challenges for existing optimization algorithms: (1) With the growth of dimensionality, the properties of optimization problems become much more complicated. In particular, in the high-dimensional environment, optimization problems usually are non-convex, non-differentiable, or even non-continuous [7–9]. This makes traditional gradient-based optimization algorithms become infeasible. (2) The solution space grows exponentially as the dimensionality increases [10–13]. This greatly challenges the optimization efficiency of most existing algorithms. (3) The landscape of optimization problems becomes more complex in a high-dimensional space. On the one hand, some unimodal problems may become multimodal with the increase of dimensionality; on the other hand, in some multimodal problems, not only does the number of local optimal regions increase rapidly, but also the local regions become much wider and flatter [11,12,14]. This likely leads to premature convergence and stagnation of existing optimization techniques.

As a kind of metaheuristic algorithm, particle swarm optimization (PSO) maintains a population of particles, each of which represents a feasible solution to optimization problems, to search the solution space for the global optimum solutions [15–17]. By means of its great merits, such as strong global search ability, independence in the mathematic properties of optimization problems, and inherent parallelism [17], PSO has witnessed rapid development and excellent success in solving complex optimization problems [18–22] since it was proposed in 1995 [15]. As a result, PSO has been widely employed to solve real-world optimization problems in daily life and industrial engineering [1,23].

However, most existing PSOs are initially designed for low-dimensional optimization problems. Confronted with large-scale optimization problems, their effectiveness usually deteriorates due to the previously mentioned challenges [24–26]. To improve the optimization effectiveness of PSO in tackling high-dimensional problems, researchers have been devoted to designing novel and effective evolution mechanisms for PSO. Broadly speaking, existing large-scale PSOs can be divided into two categories [27], namely cooperative coevolutionary large-scale PSOs [6,28,29] and holistic large-scale PSOs [24,26,30–32].

Cooperative coevolutionary PSOs (CCPSOs) [6,28,29,33] adopt the divide-and-conquer technique to decompose one large-scale optimization problem into several exclusive smaller sub-problems and then optimize these sub-problems individually by traditional PSOs designed for low-dimensional problems to find the optimal solution to the large-scale optimization problem. Since the decomposed subproblems are separately optimized, the key component of CCPSOs is the decomposition strategy [6,28]. Ideally, a good decomposition strategy should place interacted variables into the same sub-problem, so that they can be optimized together. However, without prior knowledge, it is considerably difficult to decompose a large-scale problem accurately. As a result, current research on CCPSOs lies in developing novel decomposition strategies to divide the large-scale optimization problem as accurately as possible. Hence, many effective decomposition strategies [6,34–38] have been put forward.

However, CCPSOs heavily rely on the quality of the decomposition strategies. According to the no free lunch theorem, there is no decomposition strategy suitable for all large-scale problems. Therefore, some researchers attempt to design large-scale PSOs from another perspective, namely the holistic large-scale PSOs [5,26,30,39].

In contrast to CCPSOs, holistic large-scale PSOs [5,26,30,39,40] still optimize all variables simultaneously such as traditional PSOs. Since the learning strategy in updating the velocity of particles plays the most important role in PSO [15,16,18], the key to improving the effectiveness of PSO in coping with large-scale optimization is to devise effective learning strategies for particles, which should not only help particles explore the solution space efficiently to locate promising areas fast, but also aid particles to exploit the promising areas effectively to obtain high-quality solutions. Along this line, researchers have developed many remarkable learning strategies for PSO to solve high-dimensional problems, such as the competitive learning scheme [26], the social learning strategy [30], the two-phase learning method [1], and the level-based learning approach [25]. Recently, some researchers even have attempted to develop novel coding schemes for PSO to improve its optimization performance in solving large-scale optimization problems [41].

Although the above-mentioned large-scale PSOs have presented excellent optimization performance in solving some large-scale optimization problems, they still encounter limitations, such as premature convergence and stagnation into local areas, in solving complicated high-dimensional problems, especially those with overlapping correlated variables or fully non-separable variables. Therefore, the optimization performance of PSOs in tackling large-scale optimization still deserves improvement, which still remains an open and hot topic to study in the evolutionary computation community.

In nature, individuals with better fitness usually preserve more valuable evolutionary information than those with worse fitness, to guide the evolution of one species [42]. Moreover, in general, different individuals usually preserve different useful genes. Inspired by these observations, in this paper, we propose a dimension group-based comprehensive elite learning swarm optimizer (DGCELSO) by integrating useful genes embedded in different elite individuals to guide the update of particles to search the large-scale solution space effectively and efficiently. Specifically, the main components of the proposed DGCELSO are summarized as follows:

(1) A dimension group-based comprehensive elite learning scheme is proposed to guide the update of inferior particles by learning from multiple superior ones. Instead of learning from only at most two exemplars in existing holistic large-scale PSOs [24–26,30], the devised learning strategy first randomly divides the dimensions of each inferior particle into several equally sized groups and then employs different superior particles to guide the update of different dimension groups. Moreover, unlike existing elite strategies that only use one elite to direct the evolution of an individual [43,44], it employs a random dimension group-based recombination techniques to try to integrate valuable evolutionary information in multiple elites to guide the update of each non-elite particle. In this way, the learning diversity of particles could be largely promoted, which is beneficial for particles to avoid falling into local traps. Moreover, it is also possible that useful evolutionary information embedded in different superior particles could be integrated to direct the learning of inferior particles, which may be profitable for particles to approach promising areas quickly.

(2) Dynamic adjustment strategies for the control parameters involved in the proposed learning strategy are further designed to cooperate with the learning strategy to help PSO search the large-scale solution space properly. With these dynamic strategies, the developed DGCELSO could appropriately compromise the intensification and diversification of the search process at the swarm level and the particle level.

To verify the effectiveness of the proposed DGCELSO, extensive experiments are conducted to compare DGCELSO with several state-of-the-art large-scale optimizers on the widely used CEC'2010 [7] and CEC'2013 [8] large-scale benchmark optimization problem sets. Meanwhile, deep investigations on DGCELSO are also conducted to discover what contributes to its good performance.

The rest of this paper is organized as follows. Section 2 introduces the classical PSO and large-scale PSO variants. Then, the proposed DGCELSO is elucidated in detail in Section 3. Section 4 conducts extensive experiments to verify the effectiveness of the proposed DGCELSO. Finally, Section 5 concludes this paper.

2. Related Work

In this paper, a D-dimensional single-objective minimization optimization problem is considered, which is defined as follows:

$$\min f(\boldsymbol{x}),\ \boldsymbol{x} \in R^D \tag{1}$$

where $\boldsymbol{x}$ consisting of D variables is a feasible solution to the optimization problem, and D is the dimension size. In this paper, we directly use the function value as the fitness value of one particle.

2.1. Canonical PSO

In the canonical PSO [15,16], each particle is represented by two vectors, namely the position vector $\boldsymbol{x}$ and the velocity vector $\boldsymbol{v}$. During the evolution, in the canonical PSO [15,16], each particle is guided by its historically personal best position and the historically best position of the whole swarm. Specifically, each particle is updated as follows:

$$\boldsymbol{v}_i^d \leftarrow w\boldsymbol{v}_i^d + c_1 r_1 (\boldsymbol{pbest}_i^d - \boldsymbol{x}_i^d) + c_2 r_2 (\boldsymbol{gbest}^d - \boldsymbol{x}_i^d) \tag{2}$$

$$\boldsymbol{x}_i^d \leftarrow \boldsymbol{x}_i^d + \boldsymbol{v}_i^d \tag{3}$$

where $\boldsymbol{v}_i^d$ is the dth dimension of the velocity of the ith particle, $\boldsymbol{x}_i^d$ is the dth dimension of the position of the ith particle, $\boldsymbol{pbest}_i^d$ is the dth dimension of the historically personal best position found by the ith particle, and $\boldsymbol{gbest}^d$ is the dth dimension of the historically global best position found by the whole swarm. As for the parameters, c_1 and c_2 are two acceleration coefficients, while r_1 and r_2 are two real random numbers uniformly generated within [0, 1]. w represents the inertia weight.

As shown in Equation (2), in the canonical PSO, each particle is cognitively directed by its ***pbest*** (the second part in the right hand of Equation (2) and socially guided by ***gbest*** of the whole swarm (the third part in the right hand of Equation (2). Due to the greedy attraction of ***gbest***, the swarm in the canonical PSOs usually becomes trapped in local areas when tackling multimodal problems [18,45]. Therefore, to improve the effectiveness of PSO in searching multimodal space with many local areas, researchers developed many novel learning strategies to guide the learning of particles, such as the comprehensive learning strategy [46], the genetic learning strategy [47], the scatter learning strategy [18], and the orthogonal learning strategy [48], etc.

Though a lot of novel learning strategies have helped PSO achieve very promising performance in solving multimodal problems, most of them are particularly designed for low-dimensional optimization problems. Encountered with large-scale optimization problems, most existing PSOs lose their effectiveness due to the "curse of dimensionality" and the aforementioned challenges in high-dimensional problems.

2.2. Large-Scale PSO

To solve the previously mentioned challenges of large-scale optimization, researchers devoted extensive attention to designing novel PSOs. As a result, numerous large-scale PSO variants have sprung up [1,26]. In a broad sense, existing large-scale PSOs can be classified into the following two categories.

2.2.1. Cooperative Coevolutionary Large-Scale PSO (CCPSO)

Cooperative coevolutionary PSOs (CCPSOs) [6,29,49] mainly use the divide-and-conquer technique to separate all variables of one high-dimensional problem into several exclusive groups, and then optimize each group of variables independently to obtain the optimal solution to the high-dimensional problem. Bergh and Engelbrecht put forward the earliest CCPSO [49]. In this algorithm, all variables in a large-scale optimization problem are randomly divided into K groups with each containing D/K variables (where D is the dimension size). Then the canonical PSO described in Section 2.1 is employed to optimize each group of variables. Nevertheless, the performance of this algorithm heavily relies on the setting of the number of groups (namely K). To alleviate this issue, in [29], an improved CCPSO, named CCPSO2, was proposed by first predefining a set of group numbers and then randomly selecting a group number in each iteration to separate variables into groups. In the above two algorithms, the correlations between variables are not taken into account explicitly. Hence, their optimization effectiveness degrades dramatically in solving problems with many interacted variables [11,12].

To alleviate the above issue, researchers have attempted to design effective variable grouping strategies to separate variables into groups by detecting the correlations between variables [6,35–37]. In the literature, the most representative grouping strategy is the differential grouping (DG) method [6], which uses the differential function values to detect the correlation between any two variables by exerting the same disturbance on the two variables. Based on the detected correlations between variables, DG could separate variables into groups satisfactorily. However, this method has two drawbacks. (1) It cannot detect the indirect interaction between variables [36], and (2) it consumes a lot of fitness evaluations ($O(D^2)$, D is the number of variables) in the variable decomposition stage [35,37].

To fill the first gap, Sun et al. devised an extended DG (XDG) [36], and Mei et al. brought up a global DG (GDG) [50] to detect both the direct and indirect interactions between variables. To alleviate the second predicament, a fast DG, named DG2 [35], and a recursive DG (RDG) [37] were put forward to reduce the consumption of fitness evaluations in the variable grouping stage. To further improve the detection efficiency of RDG, an efficient recursive differential grouping (ERDG) [51] was devised to reduce the used fitness evaluations in the decomposition stage, and to alleviate the sensitivity of RDG to parameters, an improved version, named RDG2, was developed [52] by adaptively adjusting the setting of parameters. In [53], Ma et al. proposed a merged differential grouping method based on subset-subset interaction and binary search by first identifying separable variables and non-separable variables, and putting all separable variables into the same subset, while dividing the non-separable variables into multiple subsets via a binary-tree-based iterative merging method. To further promote the variable grouping accuracy, Liu et al. proposed a deep grouping method by considering both the variable interaction and the essentialness of the variable to decompose one high-dimensional problem [54]. Instead of decomposing a large-scale optimization problem into fixed variable groups, Zhang et al. developed a dynamic grouping strategy to dynamically separate variables into groups during the evolution [55]. Specifically, the proposed algorithm first evaluates the contribution of variables based on the historical information and then constructs dynamic variable groups for the next generation based on the evaluated contribution and the detected interaction information.

By means of their promising performance in solving large-scale optimization problems, cooperative coevolutionary algorithms have been widely applied to solve various industrial engineering problems. For instance, Neshat et al. [56] proposed a novel multi-swarm cooperative co-evolution algorithm with the multi verse optimizer algorithm, the equilibrium optimization method, and the moth flame optimization approach, to optimize the layout of offshore wave energy converters. To tackle distributed flowshop group scheduling problems, Pan et al. [57] proposed a cooperative co-evolutionary algorithm with a collaboration model and a re-initialization scheme to tackle them. In [58], a hybrid cooperative co-evolution algorithm with a symmetric local search plus Nelder–Mead was devised to optimize the positions and the power-take-off settings of wave energy converters. In [59], Liang et al. developed a cooperative coevolutionary multi-objective evolutionary algorithm to tackle the transit network design and frequency setting problem.

Although the above-mentioned cooperative coevolutionary algorithms including CCPSOs achieved good performance in dealing with certain kinds of high-dimensional problems and have been applied to solve real-world problems, they are still confronted with limitations in tackling complicated high-dimensional problems. On the one hand, according to the theorem of No Free Lunch, there is no universal grouping method that could accurately separate variables into groups for all types of large-scale optimization problems; on the other hand, faced with high-dimensional problems with overlapping variable correlations, most existing variable grouping strategies would separate all these variables into the same group, leading to a very large variable group. Under this situation, traditional PSOs designed for low-dimensional problems used in CCPSO still cannot effectively optimize such a large group of variables. As a result, some researchers have attempted to design large-scale PSOs from another perspective to be elucidated next.

2.2.2. Holistic Large-Scale PSO

Unlike CCPSOs, holistic large-scale PSOs [18,26] still consider all variables as a whole and optimize them simultaneously like in traditional low-dimensional PSOs [16]. To solve the previously mentioned challenges of large-scale optimization, the key to holistic large-scale PSOs is to devise effective and efficient learning strategies for particles to largely promote the swarm diversity so that particles could explore the exponentially increased solution space efficiently and exploit the promising areas extensively to obtain high-quality solutions.

In [60], a dynamic multi-swarm PSO along with the Quasi-Newton local search method (DMS-L-PSO) was proposed to optimize large-scale optimization problems by dynamically separating particles into smaller sub-swarms in each generation. Taking inspiration from the competitive learning scheme in human society, Cheng and Jin proposed a competitive swarm optimizer (CSO) [26]. Specifically, this optimizer first separates particles into exclusive pairs and then lets each pair of particles compete with each other. After the competition, the winner is not updated and thus directly enters the next generation, while the loser is updated by learning from the winner. Likewise, inspired by the social learning strategy in animals, a social learning PSO (SLPSO) [61] was devised to let each particle probabilistically learn from those which are better than itself. By extending the pairwise competition mechanism in CSO to a tri-competitive strategy, Mohapatra et al. [62] developed a modified CSO (MCSO) to accelerate the convergence speed of the swarm to tackle high-dimensional problems. Taking inspiration from the comprehensive learning strategy designed for low-dimensional problems [46] and the competitive learning approach in CSO [26], Yang et al. designed a segment-based predominant learning swarm optimizer (SPLSO) [30] to cope with large-scale optimization. Specifically, this optimizer first uses the pairwise competition mechanism in CSO to divide particles into two groups, namely the relatively good particles and the relatively poor particles. Then, it further randomly separates the dimensions of each relatively poor particle into a certain number of exclusive segments, and subsequently randomly selects a relatively good particle to direct the update of each segment of the inferior particle.

Unlike the above large-scale PSOs [26,30,62], which let the updated particle learn from only one superior, Yang et al. devised a level-based learning swarm optimizer (LLSO) [25] by taking inspiration from the teaching theory in pedagogy. Specifically, this optimizer first separates particles into different levels and then lets each particle in lower levels learn from two random superior exemplars selected from higher levels. Inspired by the cooperative learning behavior in human society, Lan et al. put forward a two-phase learning swarm optimizer (TPLSO) [24]. This optimizer separates the learning of each particle into the mass learning phase and the elite learning phase. In the former learning phase, the tri-competitive mechanism is employed to update particles, while in the elite learning phase, the elite particles are picked out to learn from each other to further exploit promising areas to refine the found solutions. Similarly, Wang et al. proposed a multiple strategy learning particle swarm optimization (MSL-PSO) [40], in which different learning strategies are used to update particles in different evolution stages. In the first stage, each particle learns from those with better fitness and the mean position of the swarm to probe promising positions. Then, all the best probed positions are sorted based on their fitness and the top best ones are used to update particles in the second stage. In [41], Jian et al. developed a novel region encoding scheme to extend the solution representation from a single point to a region, and a novel adaptive region search strategy to keep the search diversity. These two schemes are then embedded into SLPSO to tackle large-scale optimization problems.

To find a good compromise between exploration and exploitation, Li et al. devised a learning structure to decouple exploration and exploitation for PSO in [63] to solve large-scale optimization. In particular, an exploration learning strategy was devised to direct particles to sparse areas based on a local sparseness degree measurement, and then an adaptive exploitation learning strategy was developed to let particles exploit the found promising areas. Deng et al. [39] devised a ranking-based biased learning swarm optimizer

(RBLSO) based on the principle that the fitness difference between learners and exemplars should be maximized. In particular, in this algorithm, a ranking paired learning (RPL) scheme was designed to let the worse particles learn peer-to-peer from the better ones, and at the same time, a biased center learning (BCL) strategy was devised to let each particle learn from the weighted mean position of the whole swarm. Lan et al. [64] proposed a hierarchical sorting swarm optimizer (HSSO) to tackle large-scale optimization. Specifically, this optimizer first divides particles into a good swarm and a bad swarm with equal sizes based on their fitness. Then, particles in the bad group are updated by learning from those in the good one. Subsequently, the good swarm is taken as a new swarm to execute the above swarm division and particle updating operations until there is only one particle in the good swarm. Kong et al. [65] devised an adaptive multi-swarm particle swarm optimizer to cope with high-dimensional problems. Specifically, it first adaptively divides particles into several sub-swarms and then employs the competition mechanism to select exemplars for particle updating. Huang et al. [66] put forward a convergence speed controller to cooperate with PSO to deal with large-scale optimization. Specifically, this controller is triggered periodically to produce an early warning to PSO before it falls into premature convergence.

Though most existing large-scale PSOs have presented their success in solving certain kinds of high-dimensional problems, their effectiveness still degrades in solving complicated high-dimensional problems [11,12,27,67], especially on those with many wide and flat local areas. Therefore, promoting the effectiveness and efficiency of PSO in solving large-scale optimization still deserves extensive attention and thus this research direction is still an active and hot topic in the evolutionary computation community.

3. Dimension Group-Based Comprehensive Elite Learning Swarm Optimizer

In nature, during the evolution of one species, those elite individuals with better adaptability to the environment usually preserve more valuable evolutionary information, such as genes, to direct the evolution of the species [42]. Moreover, different individuals may preserve different useful genes. Likewise, during the evolution of the swarm in PSO, different particles may contain useful variable values that may be close to the true global optimal solutions. Therefore, a natural idea is to integrate those useful values embedded in different particles to guide the evolution of the swarm. To this end, this paper proposes a dimension group-based comprehensive elite learning swarm optimizer (DGCELSO) to tackle large-scale optimization. The detailed components of this optimizer are elucidated as follows.

3.1. Dimension Group-Based Comprehensive Elite Learning

Given that NP particles are maintained in the swarm, the proposed DGCEL strategy first partitions the swarm into two exclusive sets, namely the elite set, denoted by ***ES***, and the non-elite set, denoted by ***NES***. Specifically, ***ES*** contains the best es particles in the swarm, while ***NES*** consists of the rest $nes = (NP - es)$ particles. Since the size of ***ES***, namely es, is related to NP, we set $es = \lceil tp * NP \rceil$ (where tp is the ratio of the elite particles in ***ES*** out of the whole swarm), for the convenience of parameter fine-tuning.

Since elite particles usually preserve more valuable evolutionary information than the non-elite ones, in this paper, we first develop an elite learning strategy (EL). Specifically, we let the elite particles in ***ES*** directly enter the next generation, while only updating the non-elite particles in ***NES***. Moreover, the elite particles in ***ES*** are employed to guide the learning of non-elite particles in ***NES***.

With respect to the elite particles, during the evolution, though they may be far from the global optimal area, they usually contain valuable genes that are very close to the true global optimal solution. To integrate the useful evolutionary information embedded in different elites, we propose a dimension group-based comprehensive learning strategy (DGCL). Specifically, during the update of each non-elite particle, the whole dimensions of this particle are first randomly shuffled and then are partitioned into NDG dimension groups (where NDG denotes the number of dimension groups), with each group containing D/NDG dimensions. In this way, the dimensions of each non-elite particle are randomly divided into NDG groups, namely $\boldsymbol{DG} = [DG^1,\ DG^2,\ \ldots,\ DG^{NDG}]$.

Here, it should be mentioned that for each non-elite particle, the dimensions are randomly shuffled, and thus it is likely that the division of dimension groups is different for different non-elite particles. In addition, if $D\%NDG$ is not zero, then the remaining dimensions are equally allocated to the first ($D\%NDG$) groups, i.e., each of the first ($D\%NDG$) groups contains ($D/NDG + 1$) dimensions.

Subsequently, unlike most existing large-scale PSOs [25,26,30] which use the same exemplars to update all dimensions of one inferior particle, the proposed DGCL uses one exemplar to update each dimension group of each non-elite particle, and thus one non-elite particle could learn from different exemplars.

Incorporating the proposed EL into the DGCL, the DGCEL is developed by using the elite particles in ***ES*** to direct the update of each dimension group of a non-elite particle. Specifically, each non-elite particle is updated as follows:

$$V_{NES_j}^{DG_i} \leftarrow r_1 V_{NES_j}^{DG_i} + r_2(X_{ES_{r1}}^{DG_i} - X_{NES_j}^{DG_i}) + \phi r_3(X_{ES_{r2}}^{DG_i} - X_{NES_j}^{DG_i}) \tag{4}$$

$$X_{NES_j}^{DG_i} \leftarrow V_{NES_j}^{DG_i} + X_{NES_j}^{DG_i} \tag{5}$$

where NES_j represents the *j*th non-elite particle in ***NES***; DG_i denotes the *i*th dimension group of the *j*th non-elite particle; $X_{NES_j}^{NG_i}$ and $V_{NES_j}^{DG_i}$ are the *i*th dimension group of the position and velocity of the *j*th particle in ***NES***, respectively; ES_{r1} and ES_{r2} are two different elite particles randomly selected from ***ES***; r_1, r_2, and r_3 are three random real parameters uniformly sampled within [0, 1]; $\phi \in [0,\ 1]$ is a control parameter in charge of the influence of the second elite particle.

As for the update of each non-elite particle in ***NES***, as shown in Equation (4), the following details should be paid careful attention:

(1) As previously mentioned, for each non-elite particle, the dimensions are randomly shuffled. As a result, the partition of dimension groups is different for different non-elite particles.

(2) For each dimension group $\boldsymbol{DG_i}$, two different elite particles $\boldsymbol{X_{ES_{r1}}}$ and $\boldsymbol{X_{ES_{r2}}}$ are first randomly selected from ***ES***. Then, the better one between these two elites (suppose it is $\boldsymbol{X_{ES_{r1}}}$) acts as the first exemplar in Equation (4), while the worse one (suppose it is $\boldsymbol{X_{ES_{r2}}}$) acts as the second exemplar to guide the update of the dimension group of the non-elite particle.

(3) The two elite particles guiding the update of each dimension group are both randomly selected. Therefore, they are likely to be different for different dimension groups.

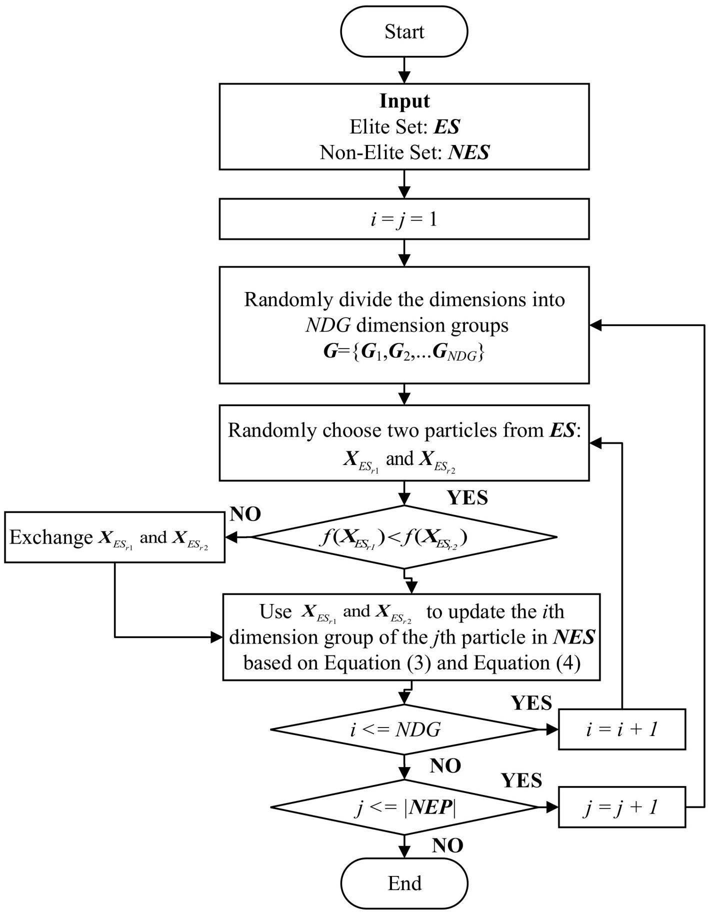
As a whole, a complete flowchart of the proposed DGCEL is shown in Figure 1. Taking deep analysis on Equation (4) and Figure 1, we find that the proposed DGCEL strategy brings the following advantages to PSO:

(1) Instead of using historical evolutionary information, such as the historically global best position (***gbest***), the personal best positions (***pbest***), and the neighborhood best position (***nbest***), in traditional PSOs [18,47], the devised DGCEL employs the elite particles in the current swarm to direct the learning of the non-elite particles. In contrast to the historical information, which may remain unchanged for many generations, particles in the swarm are usually updated generation by generation. Therefore, in the proposed DGCEL, the selected two guiding exemplars are not only likely different for different particles but also probably different for the same particle in different generations. This is very beneficial for the promotion of swarm diversity.

(2) Instead of updating each particle with the same exemplars for all dimensions in most existing large-scale PSOs [5,24–26,30], the proposed DGCEL updates non-elite particles at the dimension group level. Therefore, for different dimension groups, the

two guiding exemplars are likely different. In this way, not only could one non-elite particle learn from multiple different elite ones, but also the useful genes hidden in different elites could be incorporated to direct the evolution of the swarm. As a result, not only the learning diversity of particles could be improved, but also the learning efficiency of particles could be promoted.

(3) In DGCEL, each dimension group of a non-elite particle is guided by two randomly selected elite particles in ***ES***. With the guidance of multiple elites, each non-elite particle is expected to approach promising areas quickly. In addition, since the elite particles in ***ES*** are not updated and directly enter the next generation, the useful evolutionary information in the current swarm is protected from being destroyed by uncertain updates. Therefore, the elites in ***ES*** become better and better as the evolution iterates, and at last, it is expected that these elites converge to the optimal areas.



Start
Input
Elite Set: ES
Non-Elite Set: NES
i = j = 1
Randomly divide the dimensions into NDG dimension groups G={G1,G2,...GNDG}
Randomly choose two particles from ES: X_ES_r1 and X_ES_r2
YES
NO
f(X_ES_r1)<f(X_ES_r2)
Exchange X_ES_r1 and X_ES_r2
Use X_ES_r1 and X_ES_r2 to update the ith dimension group of the jth particle in NES based on Equation (3) and Equation (4)
YES
i <= NDG
i = i + 1
NO
YES
j <= |NEP|
j = j + 1
NO
End

Figure 1. Flowchart of the proposed DGCEL strategy.

Remark

To the best of our knowledge, there are four existing PSOs that are very similar to the proposed DGCELSO. They are CLPSO [46], OLPSO [48], GLPSO [47], and SPLSO [30]. The first three were originally designed for low-dimensional problems, while the last one was initially devised for large-scale optimization. Compared with these existing PSOs, the developed DGCELSO distinguishes from them in the following ways:

(1) In contrast to the three low-dimensional PSOs [46–48], the proposed DGCELSO uses the elite particles in the swarm to comprehensively guide the learning of the non-elite particles at the dimension group level. First, the three low-dimensional PSOs all use the personal best positions (***pbests***) of particles to construct only one guiding exemplar for each updated particle, whereas DGCELSO leverages the elite

particles in the current swarm to construct two different guiding exemplars for each non-elite particle. Second, the three low-dimensional PSOs construct the guiding exemplar dimension by dimension. Nevertheless, DGCELSO constructs the two guiding exemplars group by group. With these two differences, DGCELSO is expected to construct more promising guiding exemplars for the updated particles, and thus the learning effectiveness and efficiency of particles could be largely promoted to explore the large-scale solution space.

(2) In contrast to the large-scale PSO, namely SPLSO [30], DGCELSO uses two different elite particles to direct the update of each dimension group of each non-elite particle. First, the partition of the swarm in DGCELSO is very different from the one in SPLSO. In DGCELSO, the swarm is divided into two exclusive sets according to the fitness of particles, with the best *es* particles entering ***ES*** and the rest entering ***NES***. However, in SPLSO, particles in the swarm are paired together and each paired two particles compete with each other, with the winner entering the relatively good set and the loser entering the relatively poor set. Second, for each non-elite particle, DGCELSO adopts two random elites in ***ES*** to guide the update of each dimension group, whereas in SPLSO, each dimension group of a loser is updated by only one random relatively good particle with the other exemplar being the mean position of the relatively good set, which is shared by all updated particles. Therefore, it is expected that the learning effectiveness and efficiency of particles in DGCELSO are higher than in SPLSO. Hence, DGCELSO is expected to explore and exploit the large-scale solution space more appropriately than SPLSO.

3.2. Adaptive Strategies for Control Parameters

Taking deep investigation on the proposed DGCELSO, we find that except for the swarm size *NP*, it has three control parameters, namely the ratio of elite particles out of the whole swarm *tp*, the number of dimension groups *NDG*, and the control parameter ϕ in Equation (4). The swarm size *NP* is a common parameter for all evolutionary algorithms, which is usually problem-dependent and thus remains fine-tuned. As for ϕ, it subtly controls the influence of the second guiding exemplar in the velocity update. We also leave it to be fine-tuned in the experiment as *NP*. For the other two control parameters, we devise the following dynamic adjustment schemes to alleviate the sensitivity of DGCELSO to them.

3.2.1. Dynamic Adjustment for *tp*

With respect to the ratio of elite particles out of the whole swarm *tp*, it determines the size of the elite set ***ES***. When *tp* is large, on the one hand, a large number of particles are preserved and enter the next generation directly; on the other hand, the learning of non-elite particles is diversified due to a large number of candidate exemplars, namely the elite particles. In this situation, the swarm biases to explore the solution space. In contrast, when *tp* is small, only a small number of elites are preserved. In this case, the learning of non-elite particles is concentrated to exploit the promising areas where the elites locate. Therefore, the swarm biases to exploit the solution space. However, it should be mentioned that such a bias is not at the serious sacrifice of swarm diversity because the guiding exemplars are both randomly selected for each dimension group of each non-elite particle.

Based on the above consideration, it seems rational not to keep *tp* fixed during the evolution. To this end, we devise a dynamic adjustment strategy for *tp* as follows:

$$tp = 0.4 - 0.2 \times \frac{fes}{Fes_{max}} \tag{6}$$

where *fes* represents the number of fitness evaluations used so far, and Fes_{max} is the maximum number of fitness evaluations.

From Equation (6), it is found that *tp* is linearly decreased from 0.4 to 0.2. Therefore, at the early stage, *tp* is high, while at the late stage, *tp* is small. As a result, as the evolution proceeds, the swarm gradually tends to exploit the solution space. This just matches the expectation that the swarm should explore the solution fully in the early stages to find promising areas while exploiting the found promising areas in the late stage to obtain high-quality solutions. The effectiveness of this dynamic adjustment scheme will be verified in the experiments in Section 4.3.

3.2.2. Dynamic Adjustment for NDG

In terms of the number of dimension groups *NDG*, it directly affects the learning of non-elite particles. A large *NDG* leads to a large number of elite particles that might participate in the learning of non-elite particles. This might be useful when the useful genes are scattered in very diversified dimensions. In this situation, with a large *NDG*, the chance of integrating the useful genes together to direct the learning of non-elite particles could be promoted. By contrast, when the useful genes are scattered in centered dimensions, a small *NDG* is preferred. However, without prior knowledge of the positions of useful genes embedded in the elite particles, it is difficult to give a proper setting of *NDG*.

To alleviate the above concern, we devise the following dynamic adjustment of *NDG* for each non-elite particle based on the Cauchy distribution:

$$NDG_{NES_j} \sim Cauchy(60, 10) \tag{7}$$

$$NDG_{NES_j} = floor(NDG_{NES_j}/10) * 10 + \begin{cases} 0 & if \bmod(NDG_{NES_j}, 10) < 5 \\ 10 & otherwise \end{cases} \tag{8}$$

where NDG_{NES_j} denotes the setting of *NDG* for the *j*th particle in ***NES***, *Cauchy* (60, 10) is a Cauchy distribution with the position parameter 60 and scaling parameter 10. *floor*(x) is a function that returns the largest integer smaller than x. *mod*(x,y) is a function that returns the remainder when x/y.

In Equations (7) and (8), two details deserve careful attention. First, the Cauchy distribution is used here because it can generate values around the position parameter with a long fat tail. With this distribution, the generated *NDG*s for different non-elite particles are likely diversified. Second, with Equation (8), we keep the setting of *NDG* for each non-elite particle at multiple times of 10. This setting is adopted here for promoting the difference between two different values of *NDG* to improve the learning diversity of non-elite particles and for the convenience of computation.

From Equations (7) and (8), it is found that different non-elite particles likely preserve different *NDGs*. On the one hand, the learning diversity of non-elite particles could be further improved. On the other hand, the chance of integrating useful genes embedded in different elite particles is likely promoted with different settings of *NDG*. The effectiveness of this dynamic adjustment scheme for *NDG* will be verified in the experiments in Section 4.3.

3.3. Overall Procedure of DGCELSO

By integrating the above components, DGCELSO is developed with the overall procedure outlined in Algorithm 1 and the complete flowchart shown in Figure 2. Specifically, after the swarm is initialized and evaluated (Line 1), the algorithm goes to the main iteration loop (Lines 2~17). First, the swarm is partitioned into the elite set (***ES***) and the non-elite set (***NES***) as shown in Lines 3 and 4. Then, each particle in ***NES*** is updated as shown in Lines 5~16. During the update of one non-elite particle, the dimensions of this particle are first separated into several dimension groups (Lines 6 and 7). Then, for each dimension group of the non-elite particle, two different elite particles are randomly selected from ***ES*** (Line 9), and then the dimension group is updated by learning from these two elites (Line 13). The above process iterates until the termination condition is met. At the end of the algorithm, the best solution in the swarm is output (Line 18).

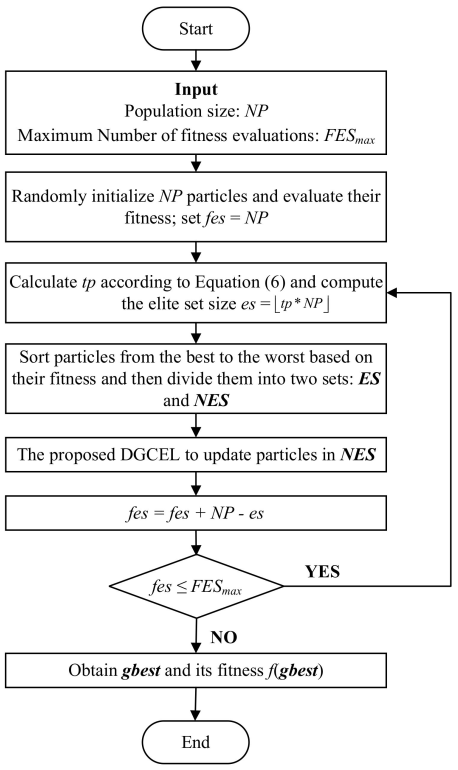

Start
Input
Population size: NP
Maximum Number of fitness evaluations: FES_max
Randomly initialize NP particles and evaluate their fitness; set fes = NP
Calculate tp according to Equation (6) and compute the elite set size es = ⌊tp * NP⌋
Sort particles from the best to the worst based on their fitness and then divide them into two sets: ES and NES
The proposed DGCEL to update particles in NES
fes = fes + NP - es
fes ≤ FES_max
YES
NO
Obtain gbest and its fitness f(gbest)
End

Figure 2. Flowchart of the proposed DGCELSO.

With respect to the computational complexity in time, from Algorithm 1, it is found that in each generation, it takes $O(NP\log_2 NP)$ to sort the swarm and $O(NP)$ to partition the swarm into two sets in Line 4; then, it takes $O(NP*D)$ to shuffle the dimensions and $O(NP*D)$ to partition the shuffled dimensions into groups for all non-elite particles (Line 7); at last, it takes $O(NP*D)$ to update all non-elite particles (Lines 8~14). To sum up, the time complexity of DGCELSO is $O(NP*D)$ based on the consideration that the swarm size is usually much smaller than the dimension size in large-scale optimization.

Algorithm 1: The Pseudocode of DGCELSO.

```
Input: Population size $NP$, Maximum number of fitness evaluations $FES_{max}$, Control parameter $\phi$;
1:  Initialize $NP$ particles randomly and calculate their fitness; $fes = NP$;
2:  While ($fes \leq FES_{max}$) do
3:    Calculate $tp$ according to Equation (6) and obtain the elite set size $es = \lfloor tp * NP \rfloor$;
4:    Sort particles based on their fitness and divide them into two sets, namely ES and NES;
5:    For each non-elite particle $NES_j$ in NES do
6:      Generate $NDG_{NES_j}$ based on Equation (7);
7:      Random shuffle the dimensions and then split the dimensions into $NDG_{NES_j}$ groups;
8:      For each dimension group $DG_i$ do
9:        Randomly select two different elite particles from ES: $X_{ESr1}$ and $X_{ESr2}$;
10:       If ($f(X_{ESr2}) < f(X_{ESr1})$) then
11:         Swap $ESr1$ and $ESr2$;
12:       End If
13:       Update the dimension group of $NES_j$ according to Equations (3) and (4);
14:     End For
15:     Calculate the fitness of the updated $NES_j$, and $fes$ ++;
16:   End For
17: End While
18: Obtain the best solution in the swarm gbest and its fitness $f(gbest)$
Output: $f(gbest)$ and gbest
```

Regarding the computational complexity in space occupation, in Algorithm 1, we can see that except for $O(NP*D)$ to store the positions of all particles and $O(NP*D)$ to store the velocities of all particles, it only takes extra $O(NP)$ to store the index of particles in the two sets, and $O(D)$ to store the dimension groups. Comprehensively, DGCELSO only takes $O(NP*D)$ space.

Based on the above time and space complexity analysis, it is found that the proposed DGCELSO remains as efficient as the classical PSO, which also takes $O(NP*D)$ time in each generation and $O(NP*D)$ space.

4. Experimental Section

To verify the effectiveness of the proposed DGCELSO, extensive experiments are conducted on two sets of large-scale optimization problems, namely the CEC'2010 [7] and the CEC'2013 [8] large-scale benchmark sets in this section. The CEC'2010 set contains 20 high-dimensional problems with 1000 dimensions, while the CEC'2013 set consists of 15 problems with 1000 dimensions as well. In particular, the CEC'2013 set is an extension of the CEC'2010 set by introducing more complicated features, such as overlapping interactions among variables and imbalance contribution of variables. Therefore, compared with the CEC'2010 problems, the CEC'2013 problems are more complicated and more difficult to optimize. For more detailed information on the two benchmark large-scale problem sets, readers are referred to [7,8].

In this section, we first investigate the settings of two key parameters (namely the swarm size NP and the control parameter ϕ) for DGCELSO in Section 4.1. Then, extensive experiments are conducted on the two benchmark sets to compare DGCELSO with several state-of-the-art large-scale optimizers in Section 4.2. At last, a deep investigation into the proposed DGCELSO is performed to observe what contributes to the good performance of DGCELSO.

In the experiments, unless otherwise stated, the maximum number of fitness evaluations is set as $3000 \times D$, where D is the dimension size. In this paper, the dimension size of all optimization problems is 1000, and thus the total number of fitness evaluations is 3×10^6. To make fair and comprehensive comparisons, the median, the mean, and the standard deviation (Std) values over 30 independent runs are used to evaluate the performance of all algorithms. Moreover, to tell the statistical significance, the Wilcoxon rank-sum test at the significance level of "$\alpha = 0.05$" was conducted to compare two different algorithms. Furthermore, to obtain the overall ranks of different algorithms on one whole benchmark set, the Friedman test at the significance level of "$\alpha = 0.05$" was conducted on each benchmark set.

Lastly, it is worth noting that we use the C programming language and Code Blocks software to implement the proposed DGCELO. Moreover, all experiments were run on a PC with 8 Intel Core i7-10700 2.90-GHz CPUs, 8-GB memory, and the 64-bit Ubuntu 12.04 LTS system.

4.1. Parameter Setting

Due to the proposed two dynamic adjustment strategies of the associated parameters in DGCELSO, there are only two parameters, namely the swarm size NP and the control parameter ϕ that need fine-tuning. Therefore, to investigate the optimal setting of the two parameters for DGCELSO in solving 1000-D large-scale optimization problems, we conduct experiments by varying NP from 100 to 600 and ϕ ranging from 0.1 to 0.9 for DGCELSO on the CEC'2010 benchmark set. Table 1 shows the mean fitness values obtained by DGCELSO with different settings of NP and ϕ on the CEC'2010 set. In this table, the best results are highlighted in bold, and the average rank of each configuration is also presented, which was obtained using the Friedman test at the significance level of "$\alpha = 0.05$".

Table 1. Comparison results among DGCELSO with different settings of *NP* and ϕ on the 1000-D CEC′2010 problems.

F					$NP = 100$									$NP = 200$				
	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	$\phi = 0.6$	$\phi = 0.7$	$\phi = 0.8$	$\phi = 0.9$	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	$\phi = 0.6$	$\phi = 0.7$	$\phi = 0.8$	$\phi = 0.9$
F_1	3.31×10^{2}	5.18×10^{7}	8.23×10^{7}	1.34×10^{7}	1.12×10^{3}	2.92×10^{-23}	9.51×10^{-20}	5.27×10^{5}	1.01×10^{8}	5.12×10^{-26}	6.22×10^{-29}	5.73×10^{-27}	0.00×10^{0}	1.10×10^{-26}	2.11×10^{-22}	9.04×10^{2}	5.08×10^{7}	1.22×10^{9}
F_2	2.93×10^{3}	3.58×10^{3}	3.64×10^{3}	3.22×10^{3}	2.50×10^{3}	1.62×10^{3}	1.12×10^{3}	9.12×10^{3}	1.13×10^{4}	1.16×10^{3}	1.61×10^{3}	1.69×10^{3}	1.40×10^{3}	8.84×10^{2}	2.95×10^{3}	1.07×10^{4}	1.14×10^{4}	1.19×10^{4}
F_3	5.74×10^{0}	1.13×10^{1}	1.14×10^{1}	8.23×10^{0}	3.24×10^{0}	2.18×10^{-1}	6.43×10^{-14}	3.89×10^{-1}	1.36×10^{1}	3.47×10^{-14}	2.90×10^{-2}	1.19×10^{-1}	3.42×10^{-14}	3.81×10^{-14}	4.88×10^{-14}	3.63×10^{-1}	1.27×10^{1}	1.71×10^{1}
F_4	4.77×10^{11}	5.57×10^{12}	5.93×10^{12}	2.88×10^{12}	1.66×10^{11}	1.14×10^{11}	1.53×10^{11}	2.90×10^{11}	7.08×10^{11}	1.74×10^{11}	1.96×10^{11}	4.21×10^{11}	1.28×10^{11}	1.25×10^{11}	1.67×10^{11}	2.48×10^{11}	6.04×10^{11}	2.14×10^{13}
F_5	2.96×10^{7}	3.16×10^{7}	3.01×10^{7}	3.54×10^{7}	1.30×10^{8}	2.75×10^{8}	2.86×10^{8}	2.96×10^{8}	3.05×10^{8}	2.81×10^{8}	2.36×10^{8}	2.24×10^{8}	2.55×10^{8}	2.77×10^{8}	2.84×10^{8}	2.91×10^{8}	3.04×10^{8}	3.09×10^{8}
F_6	1.99×10^{1}	2.02×10^{1}	2.02×10^{1}	2.01×10^{1}	1.99×10^{1}	2.01×10^{1}	2.15×10^{1}	2.15×10^{1}	2.03×10^{1}	1.94×10^{1}	1.97×10^{1}	1.97×10^{1}	1.98×10^{1}	1.96×10^{1}	4.00×10^{-9}	3.82×10^{-1}	1.34×10^{1}	1.78×10^{1}
F_7	2.94×10^{6}	9.82×10^{8}	1.28×10^{9}	3.37×10^{8}	7.90×10^{5}	7.40×10^{5}	1.17×10^{5}	8.40×10^{4}	7.80×10^{5}	3.34×10^{-6}	3.70×10^{4}	1.75×10^{6}	1.40×10^{3}	8.73×10^{-6}	3.14×10^{-1}	2.51×10^{4}	5.02×10^{5}	1.76×10^{7}
F_8	3.39×10^{7}	4.89×10^{7}	4.72×10^{7}	4.42×10^{7}	1.67×10^{5}	6.68×10^{4}	1.58×10^{7}	4.17×10^{7}	4.89×10^{7}	3.33×10^{5}	5.47×10^{6}	2.47×10^{7}	1.86×10^{3}	3.95×10^{3}	1.73×10^{7}	3.97×10^{7}	4.52×10^{7}	4.62×10^{7}
F_9	8.72×10^{7}	1.03×10^{9}	1.17×10^{9}	6.34×10^{8}	2.80×10^{7}	1.75×10^{7}	4.08×10^{7}	4.70×10^{8}	1.36×10^{10}	1.97×10^{7}	3.44×10^{7}	6.48×10^{7}	1.98×10^{7}	1.47×10^{7}	4.03×10^{7}	3.65×10^{9}	2.29×10^{10}	4.11×10^{10}
F_{10}	3.14×10^{3}	3.85×10^{3}	3.97×10^{3}	3.43×10^{3}	2.65×10^{3}	1.69×10^{3}	2.66×10^{3}	1.09×10^{4}	1.16×10^{4}	1.20×10^{3}	1.76×10^{3}	1.82×10^{3}	1.48×10^{3}	9.59×10^{2}	1.01×10^{4}	1.08×10^{4}	1.14×10^{4}	1.20×10^{4}
F_{11}	7.08×10^{1}	9.71×10^{1}	9.38×10^{1}	8.60×10^{1}	5.22×10^{1}	3.03×10^{1}	2.47×10^{1}	2.53×10^{1}	6.32×10^{1}	1.57×10^{1}	2.00×10^{1}	2.03×10^{1}	2.00×10^{1}	1.09×10^{1}	1.85×10^{-13}	1.38×10^{0}	5.04×10^{1}	1.41×10^{2}
F_{12}	9.54×10^{4}	1.03×10^{6}	1.13×10^{6}	6.89×10^{5}	4.82×10^{3}	8.18×10^{2}	6.14×10^{4}	5.07×10^{6}	6.64×10^{6}	2.35×10^{3}	1.71×10^{4}	7.25×10^{4}	1.72×10^{3}	1.92×10^{3}	2.36×10^{6}	5.14×10^{6}	6.61×10^{6}	7.97×10^{6}
F_{13}	5.89×10^{3}	3.83×10^{6}	4.40×10^{6}	9.77×10^{5}	5.29×10^{3}	3.06×10^{3}	2.35×10^{3}	6.58×10^{4}	1.36×10^{8}	6.55×10^{2}	8.02×10^{2}	1.02×10^{3}	5.58×10^{2}	4.97×10^{2}	5.48×10^{2}	2.96×10^{3}	3.95×10^{7}	9.38×10^{9}
F_{14}	2.68×10^{8}	2.11×10^{9}	2.30×10^{9}	1.45×10^{9}	8.31×10^{7}	4.56×10^{7}	1.39×10^{8}	3.47×10^{9}	3.23×10^{10}	5.82×10^{7}	1.12×10^{8}	2.20×10^{8}	6.07×10^{7}	4.61×10^{7}	2.11×10^{8}	2.02×10^{10}	5.18×10^{10}	7.58×10^{10}
F_{15}	3.33×10^{3}	4.07×10^{3}	4.13×10^{3}	3.53×10^{3}	2.81×10^{3}	1.11×10^{4}	1.09×10^{4}	1.12×10^{4}	1.17×10^{4}	1.07×10^{4}	3.71×10^{3}	3.33×10^{3}	1.07×10^{4}	1.05×10^{4}	1.05×10^{4}	1.08×10^{4}	1.14×10^{4}	1.21×10^{4}
F_{16}	1.89×10^{2}	2.56×10^{2}	2.56×10^{2}	2.22×10^{2}	1.47×10^{2}	8.35×10^{1}	5.10×10^{1}	6.61×10^{1}	2.58×10^{2}	6.10×10^{-1}	1.60×10^{1}	2.71×10^{1}	6.75×10^{0}	3.42×10^{-2}	2.93×10^{-13}	1.35×10^{1}	2.52×10^{2}	3.39×10^{2}
F_{17}	3.02×10^{5}	1.61×10^{6}	1.71×10^{6}	1.27×10^{6}	3.52×10^{4}	1.06×10^{4}	2.08×10^{6}	9.92×10^{6}	1.40×10^{7}	4.93×10^{4}	9.98×10^{4}	2.77×10^{5}	2.24×10^{4}	1.18×10^{5}	6.85×10^{6}	1.08×10^{7}	1.47×10^{7}	1.80×10^{7}
F_{18}	1.70×10^{4}	7.87×10^{8}	1.16×10^{9}	3.75×10^{7}	2.67×10^{3}	1.71×10^{3}	2.72×10^{3}	6.17×10^{6}	4.49×10^{10}	1.95×10^{3}	2.54×10^{3}	3.90×10^{3}	1.66×10^{3}	1.30×10^{3}	1.59×10^{3}	1.71×10^{7}	2.95×10^{10}	1.40×10^{11}
F_{19}	2.34×10^{6}	4.66×10^{6}	4.74×10^{6}	3.92×10^{6}	1.72×10^{6}	6.52×10^{6}	1.48×10^{7}	2.01×10^{7}	2.49×10^{7}	9.10×10^{6}	2.46×10^{6}	2.41×10^{6}	5.98×10^{6}	1.09×10^{7}	1.60×10^{7}	2.09×10^{7}	2.58×10^{7}	3.04×10^{7}
F_{20}	8.41×10^{3}	1.01×10^{9}	1.39×10^{9}	4.32×10^{7}	2.93×10^{3}	1.29×10^{3}	1.25×10^{3}	7.61×10^{6}	4.93×10^{10}	1.41×10^{3}	2.13×10^{3}	2.72×10^{3}	1.51×10^{3}	1.10×10^{3}	1.02×10^{3}	2.27×10^{7}	3.25×10^{10}	1.47×10^{11}
Rank	3.75	6.35	7.05	5.30	2.80	2.45	3.25	5.75	8.30	3.25	4.25	5.25	3.15	2.45	3.95	6.25	7.70	8.75

F					$NP = 300$									$NP = 400$				
	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	$\phi = 0.6$	$\phi = 0.7$	$\phi = 0.8$	$\phi = 0.9$	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	$\phi = 0.6$	$\phi = 0.7$	$\phi = 0.8$	$\phi = 0.9$
F_1	9.78×10^{-27}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	7.97×10^{-10}	5.83×10^{5}	1.86×10^{8}	2.23×10^{9}	6.50×10^{-24}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	8.36×10^{-24}	4.18×10^{-3}	3.39×10^{6}	3.30×10^{8}	3.01×10^{9}
F_2	6.93×10^{2}	1.06×10^{3}	1.15×10^{3}	8.88×10^{2}	5.90×10^{2}	1.04×10^{4}	1.09×10^{4}	1.15×10^{4}	1.21×10^{4}	5.75×10^{2}	8.12×10^{2}	8.78×10^{2}	6.57×10^{2}	9.82×10^{3}	1.05×10^{4}	1.10×10^{4}	1.16×10^{4}	1.22×10^{4}
F_3	3.36×10^{-14}	3.05×10^{-14}	3.15×10^{-14}	3.18×10^{-14}	3.88×10^{-14}	1.15×10^{-7}	5.14×10^{0}	1.47×10^{1}	1.77×10^{1}	3.51×10^{-14}	2.99×10^{-14}	2.98×10^{-14}	3.15×10^{-14}	3.98×10^{-14}	3.29×10^{-4}	7.64×10^{0}	1.54×10^{1}	1.79×10^{1}
F_4	2.22×10^{11}	2.13×10^{11}	2.00×10^{11}	1.60×10^{11}	1.57×10^{11}	2.06×10^{11}	3.80×10^{11}	1.31×10^{12}	6.16×10^{13}	2.88×10^{11}	2.60×10^{11}	2.27×10^{11}	1.96×10^{11}	1.82×10^{11}	2.48×10^{11}	5.19×10^{11}	3.12×10^{12}	1.09×10^{14}
F_5	2.83×10^{8}	2.81×10^{8}	2.76×10^{8}	2.80×10^{8}	2.82×10^{8}	2.86×10^{8}	2.93×10^{8}	3.02×10^{8}	3.18×10^{8}	2.82×10^{8}	2.82×10^{8}	2.81×10^{8}	2.78×10^{8}	2.83×10^{8}	2.89×10^{8}	2.94×10^{8}	3.06×10^{8}	3.16×10^{8}
F_6	4.00×10^{-9}	6.18×10^{0}	1.86×10^{1}	4.00×10^{-9}	4.00×10^{-9}	2.08×10^{-7}	5.36×10^{0}	1.54×10^{1}	1.84×10^{1}	4.00×10^{-9}	3.88×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.72×10^{-4}	8.08×10^{0}	1.61×10^{1}	1.86×10^{1}
F_7	3.43×10^{-3}	1.20×10^{-3}	3.63×10^{-2}	2.15×10^{-5}	2.54×10^{-3}	3.60×10^{2}	1.83×10^{5}	9.66×10^{5}	4.68×10^{8}	8.32×10^{-1}	3.50×10^{-2}	3.16×10^{-2}	7.06×10^{-3}	1.03×10^{0}	7.18×10^{3}	3.40×10^{5}	5.23×10^{6}	1.55×10^{9}
F_8	1.37×10^{7}	3.51×10^{5}	3.72×10^{4}	4.36×10^{3}	9.82×10^{5}	3.01×10^{7}	4.30×10^{7}	4.58×10^{7}	4.65×10^{7}	2.23×10^{7}	1.00×10^{7}	3.36×10^{6}	6.67×10^{5}	1.33×10^{7}	3.52×10^{7}	4.41×10^{7}	4.61×10^{7}	4.67×10^{7}
F_9	2.47×10^{7}	2.28×10^{7}	2.49×10^{7}	1.77×10^{7}	2.14×10^{7}	1.69×10^{8}	1.40×10^{10}	3.19×10^{10}	5.08×10^{10}	3.12×10^{7}	2.42×10^{7}	2.41×10^{7}	2.01×10^{7}	3.01×10^{7}	1.36×10^{9}	1.92×10^{10}	3.70×10^{10}	5.55×10^{10}
F_{10}	8.49×10^{2}	1.13×10^{3}	1.22×10^{3}	9.23×10^{2}	9.75×10^{3}	1.05×10^{4}	1.09×10^{4}	1.15×10^{4}	1.22×10^{4}	9.74×10^{3}	8.59×10^{2}	9.25×10^{2}	1.11×10^{3}	1.02×10^{4}	1.05×10^{4}	1.10×10^{4}	1.16×10^{4}	1.22×10^{4}
F_{11}	1.25×10^{-13}	2.23×10^{-1}	6.68×10^{0}	1.10×10^{-13}	1.17×10^{-13}	3.45×10^{-7}	8.82×10^{0}	7.59×10^{1}	1.59×10^{2}	1.30×10^{-13}	1.11×10^{-13}	1.05×10^{-13}	1.06×10^{-13}	1.31×10^{-13}	7.60×10^{-4}	1.51×10^{1}	9.53×10^{1}	1.67×10^{2}
F_{12}	2.50×10^{4}	4.39×10^{3}	5.55×10^{3}	2.55×10^{3}	8.74×10^{4}	4.13×10^{6}	5.75×10^{6}	7.14×10^{6}	8.45×10^{6}	1.71×10^{5}	9.83×10^{3}	7.41×10^{3}	1.18×10^{4}	2.03×10^{6}	4.62×10^{6}	6.16×10^{6}	7.49×10^{6}	8.77×10^{6}
F_{13}	5.69×10^{2}	5.35×10^{2}	5.91×10^{2}	5.15×10^{2}	4.69×10^{2}	4.85×10^{2}	1.08×10^{5}	5.63×10^{8}	1.67×10^{10}	5.31×10^{2}	5.36×10^{2}	5.46×10^{2}	4.93×10^{2}	4.50×10^{2}	4.80×10^{2}	3.72×10^{5}	1.46×10^{9}	2.23×10^{10}
F_{14}	7.79×10^{7}	6.96×10^{7}	7.62×10^{7}	5.17×10^{7}	7.69×10^{7}	5.73×10^{9}	3.85×10^{10}	6.50×10^{10}	8.96×10^{10}	1.14×10^{8}	7.11×10^{7}	7.20×10^{7}	6.01×10^{7}	1.43×10^{8}	1.68×10^{10}	4.63×10^{10}	7.22×10^{10}	9.64×10^{10}
F_{15}	1.04×10^{4}	1.05×10^{4}	1.05×10^{4}	1.04×10^{4}	1.03×10^{4}	1.06×10^{4}	1.10×10^{4}	1.16×10^{4}	1.22×10^{4}	1.04×10^{4}	1.03×10^{4}	1.04×10^{4}	1.03×10^{4}	1.03×10^{4}	1.06×10^{4}	1.11×10^{4}	1.17×10^{4}	1.23×10^{4}
F_{16}	2.15×10^{-13}	5.86×10^{-2}	9.78×10^{-2}	1.55×10^{-13}	2.01×10^{-13}	2.36×10^{-6}	1.02×10^{2}	2.92×10^{2}	3.52×10^{2}	2.39×10^{-13}	1.61×10^{-13}	1.53×10^{-13}	1.60×10^{-13}	2.44×10^{-13}	7.14×10^{-3}	1.52×10^{2}	3.07×10^{2}	3.57×10^{2}
F_{17}	1.53×10^{6}	5.71×10^{4}	5.65×10^{4}	6.57×10^{4}	4.44×10^{6}	8.75×10^{6}	1.28×10^{7}	1.65×10^{7}	1.98×10^{7}	4.81×10^{6}	1.72×10^{5}	1.03×10^{5}	7.07×10^{5}	6.20×10^{6}	1.02×10^{7}	1.37×10^{7}	1.72×10^{7}	2.06×10^{7}
F_{18}	1.56×10^{3}	1.65×10^{3}	1.56×10^{3}	1.31×10^{3}	1.13×10^{3}	1.26×10^{3}	1.57×10^{9}	5.42×10^{10}	1.76×10^{11}	1.31×10^{3}	1.31×10^{3}	1.33×10^{3}	1.25×10^{3}	1.12×10^{3}	3.88×10^{3}	4.92×10^{9}	6.87×10^{10}	1.94×10^{11}
F_{19}	1.33×10^{7}	8.92×10^{6}	8.14×10^{6}	1.02×10^{7}	1.43×10^{7}	1.88×10^{7}	2.25×10^{7}	2.75×10^{7}	3.21×10^{7}	1.45×10^{7}	1.09×10^{7}	1.02×10^{7}	1.21×10^{7}	1.53×10^{7}	1.94×10^{7}	2.47×10^{7}	2.87×10^{7}	3.36×10^{7}
F_{20}	1.12×10^{3}	1.32×10^{3}	1.40×10^{3}	1.08×10^{3}	9.79×10^{2}	1.00×10^{3}	1.88×10^{9}	5.74×10^{10}	1.82×10^{11}	9.89×10^{2}	1.15×10^{3}	1.10×10^{3}	9.85×10^{2}	9.82×10^{2}	1.72×10^{3}	5.69×10^{9}	7.40×10^{10}	2.03×10^{11}
Rank	3.80	3.48	4.03	2.03	2.88	5.00	6.90	7.95	8.95	3.90	2.90	2.50	2.05	3.95	5.70	7.00	8.00	9.00

Table 1. *Cont.*

F	$NP = 500$									$NP = 600$								
	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	$\phi = 0.6$	$\phi = 0.7$	$\phi = 0.8$	$\phi = 0.9$	$\phi = 0.1$	$\phi = 0.2$	$\phi = 0.3$	$\phi = 0.4$	$\phi = 0.5$	$\phi = 0.6$	$\phi = 0.7$	$\phi = 0.8$	$\phi = 0.9$
F_1	$8.02 \times 1^{0-22}$	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	6.91×10^{-21}	4.51×10^{0}	8.52×10^{6}	4.72×10^{8}	3.65×10^{9}	2.72×10^{-19}	2.86×10^{-26}	0.00×10^{0}	5.33×10^{-26}	9.43×10^{-16}	1.90×10^{2}	1.57×10^{7}	6.02×10^{8}	4.18×10^{9}
F_2	8.73×10^{3}	6.81×10^{2}	7.34×10^{2}	5.74×10^{2}	1.01×10^{4}	1.06×10^{4}	1.11×10^{4}	1.16×10^{4}	1.23×10^{4}	9.87×10^{3}	6.03×10^{2}	6.59×10^{2}	4.15×10^{3}	1.02×10^{4}	1.06×10^{4}	1.11×10^{4}	1.17×10^{4}	1.23×10^{4}
F_3	3.92×10^{-14}	2.96×10^{-14}	2.90×10^{-14}	3.12×10^{-14}	8.55×10^{-14}	1.24×10^{-2}	9.15×10^{0}	1.58×10^{1}	1.80×10^{1}	8.00×10^{-13}	2.97×10^{-14}	2.90×10^{-14}	3.16×10^{-14}	6.07×10^{-11}	1.06×10^{-1}	1.02×10^{1}	1.61×10^{1}	1.82×10^{1}
F_4	3.35×10^{11}	3.11×10^{11}	2.86×10^{11}	2.49×10^{11}	2.16×10^{11}	3.21×10^{11}	6.31×10^{11}	1.05×10^{13}	1.22×10^{14}	4.16×10^{11}	3.83×10^{11}	3.30×10^{11}	2.84×10^{11}	2.67×10^{11}	3.79×10^{11}	7.59×10^{11}	1.73×10^{13}	1.39×10^{14}
F_5	2.80×10^{8}	2.76×10^{8}	2.78×10^{8}	2.77×10^{8}	2.77×10^{8}	2.90×10^{8}	2.95×10^{8}	3.06×10^{8}	3.20×10^{8}	2.80×10^{8}	2.79×10^{8}	2.76×10^{8}	2.79×10^{8}	2.84×10^{8}	2.87×10^{8}	2.99×10^{8}	3.03×10^{8}	3.18×10^{8}
F_6	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	1.69×10^{-2}	9.72×10^{0}	1.66×10^{1}	1.88×10^{1}	4.09×10^{-9}	3.88×10^{-9}	3.88×10^{-9}	4.00×10^{-9}	6.07×10^{-9}	1.45×10^{-1}	1.08×10^{1}	1.68×10^{1}	1.89×10^{1}
F_7	2.61×10^{1}	1.21×10^{0}	7.81×10^{-1}	4.75×10^{-1}	3.43×10^{1}	3.08×10^{4}	4.80×10^{5}	3.01×10^{7}	2.65×10^{9}	2.63×10^{2}	1.79×10^{1}	1.21×10^{1}	8.88×10^{0}	3.52×10^{2}	7.34×10^{4}	6.46×10^{5}	1.15×10^{8}	3.95×10^{9}
F_8	2.74×10^{7}	1.73×10^{7}	1.16×10^{7}	9.50×10^{6}	2.05×10^{7}	3.79×10^{7}	4.47×10^{7}	4.63×10^{7}	4.68×10^{7}	3.08×10^{7}	2.23×10^{7}	1.74×10^{7}	1.59×10^{7}	2.52×10^{7}	3.96×10^{7}	4.50×10^{7}	4.64×10^{7}	4.69×10^{7}
F_9	3.94×10^{7}	2.71×10^{7}	2.62×10^{7}	2.33×10^{7}	4.11×10^{7}	4.17×10^{9}	2.23×10^{10}	4.01×10^{10}	5.87×10^{10}	4.84×10^{7}	3.01×10^{7}	2.85×10^{7}	2.62×10^{7}	5.78×10^{7}	6.49×10^{9}	2.49×10^{10}	4.26×10^{10}	6.14×10^{10}
F_{10}	1.00×10^{4}	1.35×10^{3}	7.94×10^{2}	9.73×10^{3}	1.02×10^{4}	1.06×10^{4}	1.11×10^{4}	1.17×10^{4}	1.23×10^{4}	1.02×10^{4}	9.45×10^{3}	6.43×10^{3}	9.99×10^{3}	1.03×10^{4}	1.06×10^{4}	1.11×10^{4}	1.17×10^{4}	1.24×10^{4}
F_{11}	1.67×10^{-13}	1.11×10^{-13}	1.04×10^{-13}	1.10×10^{-13}	5.51×10^{-13}	2.59×10^{-2}	1.94×10^{1}	1.08×10^{2}	1.72×10^{2}	6.69×10^{-12}	1.12×10^{-13}	1.05×10^{-13}	1.13×10^{-13}	4.00×10^{-10}	1.90×10^{-1}	2.66×10^{1}	1.17×10^{2}	1.75×10^{2}
F_{12}	1.54×10^{6}	2.34×10^{4}	1.47×10^{4}	4.12×10^{4}	3.02×10^{6}	4.87×10^{6}	6.37×10^{6}	7.75×10^{6}	9.01×10^{6}	2.65×10^{6}	4.99×10^{4}	2.82×10^{4}	1.18×10^{5}	3.37×10^{6}	5.07×10^{6}	6.50×10^{6}	7.83×10^{6}	9.11×10^{6}
F_{13}	5.27×10^{2}	4.92×10^{2}	4.67×10^{2}	4.65×10^{2}	4.69×10^{2}	4.80×10^{2}	1.08×10^{6}	2.45×10^{9}	2.62×10^{10}	4.82×10^{2}	5.44×10^{2}	5.20×10^{2}	4.56×10^{2}	4.42×10^{2}	5.63×10^{2}	3.15×10^{6}	3.37×10^{9}	2.90×10^{10}
F_{14}	1.66×10^{8}	8.14×10^{7}	7.67×10^{7}	7.21×10^{7}	3.28×10^{8}	2.36×10^{10}	5.14×10^{10}	7.78×10^{10}	1.30×10^{11}	2.61×10^{8}	9.02×10^{7}	8.54×10^{7}	8.71×10^{7}	8.67×10^{8}	2.77×10^{10}	5.44×10^{10}	7.81×10^{10}	1.03×10^{11}
F_{15}	1.03×10^{4}	1.03×10^{4}	1.03×10^{4}	1.03×10^{4}	1.03×10^{4}	1.07×10^{4}	1.11×10^{4}	1.17×10^{4}	1.24×10^{4}	1.03×10^{4}	1.03×10^{4}	1.03×10^{4}	1.03×10^{4}	1.03×10^{4}	1.07×10^{4}	1.12×10^{4}	1.17×10^{4}	1.24×10^{4}
F_{16}	2.83×10^{-13}	1.65×10^{-13}	1.55×10^{-13}	1.66×10^{-13}	1.34×10^{-12}	2.74×10^{-1}	1.83×10^{2}	3.16×10^{2}	3.61×10^{2}	1.54×10^{-11}	1.74×10^{-13}	1.58×10^{-13}	1.80×10^{-13}	1.18×10^{-9}	2.41×10^{0}	2.05×10^{2}	3.22×10^{2}	3.63×10^{2}
F_{17}	5.98×10^{6}	7.91×10^{5}	2.80×10^{5}	3.07×10^{6}	7.05×10^{6}	1.07×10^{7}	1.43×10^{7}	1.76×10^{7}	2.10×10^{7}	6.68×10^{6}	2.56×10^{6}	1.14×10^{6}	4.24×10^{6}	7.51×10^{6}	1.11×10^{7}	1.44×10^{7}	1.81×10^{7}	2.12×10^{7}
F_{18}	1.18×10^{3}	1.26×10^{3}	1.18×10^{3}	1.13×10^{3}	1.00×10^{3}	1.10×10^{5}	8.30×10^{9}	7.98×10^{10}	2.08×10^{11}	1.08×10^{3}	1.22×10^{3}	1.22×10^{3}	1.05×10^{3}	9.59×10^{2}	1.41×10^{6}	1.12×10^{10}	8.67×10^{10}	2.19×10^{11}
F_{19}	1.55×10^{7}	1.23×10^{7}	1.15×10^{7}	1.31×10^{7}	1.64×10^{7}	2.02×10^{7}	2.50×10^{7}	2.92×10^{7}	3.32×10^{7}	1.62×10^{7}	1.31×10^{7}	1.23×10^{7}	1.40×10^{7}	1.72×10^{7}	2.06×10^{7}	2.48×10^{7}	2.97×10^{7}	3.49×10^{7}
F_{20}	9.94×10^{2}	1.02×10^{3}	1.06×10^{3}	9.70×10^{2}	9.78×10^{2}	1.10×10	9.17×10^{9}	8.52×10^{10}	2.20×10^{11}	9.91×10^{2}	9.85×10^{2}	9.94×10^{2}	9.77×10^{2}	9.86×10^{2}	1.53×10^{6}	1.28×10^{10}	9.33×10^{10}	2.30×10^{11}
Rank	4.25	2.75	2.00	2.00	4.15	5.85	7.00	8.00	9.00	4.10	2.65	1.85	2.30	4.20	5.90	7.00	8.00	9.00

From this table, we obtain the following findings. (1) From the perspective of the Friedman test, when *NP* is fixed, the setting of parameter ϕ is neither too small nor too large, and the optimal setting is usually within [0.3, 0.6]. Specifically, when *NP* is 100 and 200, the optimal ϕ is 0.6 and 0.5 respectively. When *NP* is within [300, 500], the optimal ϕ is consistently 0.4. When *NP* is 600, the optimal ϕ is 0.3. (2) More specifically, we find that when *NP* is small, such as 100, the optimal ϕ is usually large. This is because a small *NP* could not afford enough diversity for DGCELPSO to explore the solution space. Therefore, to improve the diversity, ϕ should be large to enhance the influence of the second guiding exemplar in Equation (4), which is in charge of preventing the updated particle from being greedily attracted by the first guiding exemplar. On the contrary, when *NP* is large, such as 600, a small ϕ is preferred. This is because a large *NP* offers too high diversity for DGCELPSO to slow down its convergence. Consequently, to let particles fully exploit the found promising areas, ϕ should be small to decrease the influence of the second guiding exemplar in Equation (4). (3) Taking comprehensive comparisons among all settings of *NP* along with the associated optimal settings of ϕ, we find that DGCELSO with *NP* = 300 and ϕ = 0.4 achieves the best overall performance.

Based on the above observation, *NP* = 300 and ϕ = 0.4 are adopted for DGCELSO in the experiments related to 1000-D optimization problems.

4.2. Comparisons with State-of-the-Art Methods

To comprehensively verify the effectiveness of the devised DGCELSO, this section conducts extensive comparison experiments to compare DGCELSO with several state-of-the-art large-scale algorithms. Specifically, nine popular and latest large-scale methods are selected, namely TPLSO [24], SPLSO (The source code can be downloaded from https://gitee.com/mmmyq/SPLSO, accessed on 1 January 2022) [30], LLSO (The source code can be downloaded from https://gitee.com/mmmyq/LLSO, accessed on 1 January 2022) [25], CSO (The source code can be downloaded from http://www.soft-computing.de/CSO_Matlab_New.zip, accessed on 1 January 2022) [26], SLPSO (The source code can be downloaded from http://www.soft-computing.de/SL_PSO_Matlab.zip, accessed on 1 January 2022) [61], DECC-GDG (The source code can be downloaded from https://ww2.mathworks.cn/matlabcentral/mlc-downloads/downloads/submissions/45783/versions/1/download/zip/CC-GDG-CMAES.zip, accessed on 1 January 2022) [50], DECC-DG2 (The source code can be downloaded from https://bitbucket.org/mno/differential-grouping2/src/master/, accessed on 1 January 2022) [35], DECC-RDG (The source code can be downloaded from https://www.researchgate.net/profile/Yuan-Sun-18/publications, accessed on 1 January 2022) [37], and DECC-RDG2 (The source code can be downloaded from https://www.researchgate.net/profile/Yuan-Sun-18/publications, accessed on 1 January 2022) [52]. The former five large-scale optimizers are state-of-the-art holistic large-scale PSO variants, while the latter four algorithms are state-of-the-art cooperative coevolutionary evolutionary algorithms. Compared with these nine different state-of-the-art large-scale optimizers, the effectiveness of DGCELSO is expected to be demonstrated.

Tables 2 and 4 display the comparison results between DGCELSO and the nine compared algorithms on the 1000-D CEC'2010 and the 1000-D CEC'2013 large-scale benchmark sets, respectively. In these two tables, the symbols, "+", "−", and "=" above the p-values obtained from the Wilcoxon rank test denote that the proposed DGCELSO is significantly superior to, significantly inferior to, and equivalent to the associated compared algorithms on the related functions, respectively. "*w/t/l*" in the second to last rows of the two tables count the numbers of functions where DGCELSO performs significantly better, equivalently, and significantly worse than the associated compared methods. Actually, they are the numbers of "+", "=" and "−", respectively. In the last rows of the two tables, the averaged ranks of all algorithms obtained from the Friedman test are presented as well.

Table 2. Fitness comparison between DECELSO and the compared algorithms on the 1000-D CEC'2010 problems with 3×10^6 fitness evaluations.

F	Quality	DGCELSO	TPLSO	SPLSO	LLSO	CSO	SLPSO	DECC-GDG	DECC-DG2	DECC-RDG	DECC-RDG2
F_1	Median	0.00×10^{0}	1.98×10^{-18}	7.70×10^{-20}	2.97×10^{-22}	4.64×10^{-12}	7.65×10^{-18}	6.53×10^{0}	1.95×10^{-1}	2.60×10^{-3}	1.05×10^{-3}
	Mean	0.00×10^{0}	1.93×10^{-18}	7.73×10^{-20}	3.13×10^{-22}	4.75×10^{-12}	7.73×10^{-18}	6.54×10^{0}	7.34×10^{-1}	6.42×10^{0}	8.08×10^{-3}
	Std	0.00×10^{0}	3.04×10^{-19}	6.95×10^{-21}	6.93×10^{-23}	7.77×10^{-13}	8.84×10^{-19}	9.35×10^{-1}	1.61×10^{0}	3.41×10^{1}	3.28×10^{-2}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_2	Median	8.85×10^{2}	1.13×10^{3}	4.45×10^{2}	9.71×10^{2}	7.52×10^{3}	1.94×10^{3}	1.40×10^{3}	3.00×10^{3}	2.98×10^{3}	2.99×10^{3}
	Mean	8.88×10^{2}	1.11×10^{3}	4.45×10^{2}	9.78×10^{2}	7.48×10^{3}	1.93×10^{3}	1.40×10^{3}	3.00×10^{3}	2.98×10^{3}	3.00×10^{3}
	Std	4.13×10^{1}	8.28×10^{1}	1.63×10^{1}	5.17×10^{1}	2.60×10^{2}	8.05×10^{1}	2.67×10^{1}	1.34×10^{2}	1.16×10^{2}	1.35×10^{2}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10{-}8^{+}$	$4.32 \times 10{-}8^{+}$	$4.32 \times 10{-}8^{+}$
F_3	Median	3.24×10^{-14}	1.44×10^{0}	2.56×10^{-13}	2.89×10^{-14}	2.56×10^{-9}	1.88×10^{0}	1.12×10^{1}	1.08×10^{1}	1.12×10^{1}	1.11×10^{1}
	Mean	3.18×10^{-14}	1.45×10^{0}	2.52×10^{-13}	2.76×10^{-14}	2.57×10^{-9}	1.84×10^{0}	1.11×10^{1}	1.09×10^{1}	1.11×10^{1}	1.10×10^{1}
	Std	1.32×10^{-15}	1.34×10^{-1}	1.86×10^{-14}	2.16×10^{-15}	1.82×10^{-10}	2.62×10^{-1}	5.69×10^{-1}	6.40×10^{-1}	6.46×10^{-1}	6.88×10^{-1}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_4	Median	1.58×10^{11}	2.77×10^{11}	4.36×10^{11}	4.48×10^{11}	6.92×10^{11}	2.68×10^{11}	1.37×10^{14}	1.44×10^{12}	1.39×10^{12}	1.37×10^{12}
	Mean	1.60×10^{11}	2.89×10^{11}	4.30×10^{11}	4.54×10^{11}	6.87×10^{11}	2.83×10^{11}	1.38×10^{14}	1.69×10^{12}	1.49×10^{12}	1.44×10^{12}
	Std	3.72×10^{10}	9.22×10^{10}	8.17×10^{10}	1.29×10^{11}	1.76×10^{11}	8.77×10^{10}	2.68×10^{13}	6.16×10^{11}	6.33×10^{11}	5.35×10^{11}
	p-value	-	$1.00 \times 10^{0=}$	$3.49 \times 10^{-3+}$	$4.32 \times 10^{-8+}$	$1.02 \times 10^{-3+}$	$3.19 \times 10^{-7+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$3.19 \times 10^{-7+}$
F_5	Median	2.82×10^{8}	1.63×10^{7}	5.97×10^{6}	1.09×10^{7}	2.00×10^{6}	2.89×10^{7}	3.84×10^{8}	1.72×10^{8}	1.75×10^{8}	1.72×10^{8}
	Mean	2.80×10^{8}	1.59×10^{7}	6.30×10^{6}	1.16×10^{7}	2.46×10^{6}	3.04×10^{7}	3.82×10^{8}	1.75×10^{8}	1.71×10^{8}	1.73×10^{8}
	Std	9.11×106	4.51×10^{6}	1.73×10^{6}	2.93×10^{6}	1.33×10^{6}	8.42×10^{6}	1.54×10^{7}	1.84×10^{7}	1.84×10^{7}	1.50×10^{7}
	p-value	-	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$
F_6	Median	4.00×10^{-9}	2.08×10^{0}	1.00×10^{-8}	4.00×10^{-9}	8.18×10^{-7}	2.14×10^{1}	3.51×10^{5}	8.81×10^{0}	1.07×10^{1}	1.06×10^{1}
	Mean	4.00×10^{-9}	2.20×10^{0}	9.44×10^{-9}	4.00×10^{-9}	8.16×10^{-7}	1.95×10^{1}	3.58×10^{5}	8.90×10^{0}	1.05×10^{1}	1.05×10^{1}
	Std	3.73×10^{-15}	3.74×10^{-1}	1.18×10^{-9}	8.27×10^{-25}	2.57×10^{-8}	4.13×10^{0}	4.27×10^{4}	6.50×10^{-1}	7.02×10^{-1}	6.84×10^{-1}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_7	Median	1.89×10^{-5}	9.21×10^{2}	4.51×10^{2}	6.58×10^{0}	2.13×10^{4}	6.26×10^{4}	2.98×10^{10}	1.80×10^{3}	4.86×10^{1}	5.18×10^{1}
	Mean	2.15×10^{-5}	5.86×10^{3}	4.76×10^{2}	2.31×10^{1}	2.13×10^{4}	6.49×10^{4}	3.10×10^{10}	1.98×10^{3}	6.40×10^{1}	5.87×10^{1}
	Std	1.55×10^{-5}	1.03×10^{4}	1.29×10^{2}	7.45×10^{1}	4.53×10^{3}	3.81×10^{4}	4.19×10^{9}	9.49×10^{2}	4.67×10^{1}	3.71×10^{1}
	p-value	-	$2.07 \times 10^{-6+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$

Table 2. *Cont.*

F	Quality	DGCELSO	TPLSO	SPLSO	LLSO	CSO	SLPSO	DECC-GDG	DECC-DG2	DECC-RDG	DECC-RDG2
F_8	Median	4.28×10^{3}	4.78×10^{5}	3.11×10^{7}	2.33×10^{7}	3.86×10^{7}	7.51×10^{6}	6.78×10^{8}	6.05×10^{2}	6.57×10^{-1}	3.68×10^{-1}
	Mean	4.36×10^{3}	4.98×10^{5}	3.11×10^{7}	2.33×10^{7}	3.87×10^{7}	7.57×10^{6}	8.05×10^{8}	2.71×10^{5}	6.65×10^{5}	7.43×10^{-1}
	Std	4.17×10^{2}	1.43×10^{5}	9.43×10^{4}	2.96×10^{5}	8.47×10^{4}	2.44×10^{6}	4.70×10^{8}	9.94×10^{5}	1.49×10^{6}	1.24×10^{0}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_9	Median	1.76×10^{7}	4.25×10^{7}	4.57×10^{7}	4.64×10^{7}	6.65×10^{7}	3.31×10^{7}	7.45×10^{8}	2.15×10^{8}	1.76×10^{8}	1.77×10^{8}
	Mean	1.77×10^{7}	4.32×10^{7}	4.59×10^{7}	4.48×10^{7}	6.68×10^{7}	3.35×10^{7}	7.43×10^{8}	2.18×10^{8}	1.73×10^{8}	1.77×10^{8}
	Std	1.69×10^{6}	4.10×10^{6}	2.99×10^{6}	4.16×10^{6}	4.38×10^{6}	3.63×10^{6}	3.71×10^{7}	1.73×10^{7}	1.22×10^{7}	1.66×10^{7}
	p-value	-	$4.32 \times 10^{-8+}$	$1.00 \times 10^{0=}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_{10}	Median	9.18×10^{2}	9.67×10^{2}	7.99×10^{3}	8.87×10^{2}	9.58×10^{3}	2.59×10^{3}	4.16×10^{3}	6.73×10^{3}	6.32×10^{3}	6.27×10^{3}
	Mean	9.23×10^{2}	9.84×10^{2}	7.99×10^{3}	8.88×10^{2}	9.58×10^{3}	2.79×10^{3}	4.15×10^{3}	6.72×10^{3}	6.32×10^{3}	6.27×10^{3}
	Std	3.82×10^{1}	8.52×10^{1}	1.25×10^{2}	3.50×10^{1}	6.49×10^{1}	1.28×10^{3}	5.70×10^{1}	9.30×10^{1}	1.12×10^{2}	1.09×10^{2}
	p-value	-	$1.06 \times 10^{-2+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_{11}	Median	1.11×10^{-13}	3.48×10^{0}	3.02×10^{-12}	2.90×10^{0}	3.98×10^{-8}	2.37×10^{1}	5.58×10^{0}	5.39×10^{0}	4.76×10^{0}	4.86×10^{0}
	Mean	1.10×10^{-13}	3.50×10^{0}	3.05×10^{-12}	5.51×10^{0}	3.98×10^{-8}	2.42×10^{1}	5.53×10^{0}	5.59×10^{0}	4.75×10^{0}	4.86×10^{0}
	Std	2.36×10^{-15}	1.30×10^{0}	2.84×10^{-13}	5.43×10^{0}	3.19×10^{-9}	3.03×10^{0}	5.49×10^{-1}	6.12×10^{-1}	4.79×10^{-1}	3.88×10^{-1}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_{12}	Median	2.55×10^{3}	1.23×10^{4}	9.39×10^{4}	1.24×10^{4}	4.25×10^{5}	1.30×10^{4}	2.87×10^{5}	3.99×10^{4}	2.22×10^{4}	2.21×10^{4}
	Mean	2.55×10^{3}	1.23×10^{4}	9.53×10^{4}	1.23×10^{4}	4.37×10^{5}	1.54×10^{4}	2.87×10^{5}	3.94×10^{4}	2.21×10^{4}	2.19×10^{4}
	Std	2.13×10^{2}	1.30×10^{3}	6.64×10^{3}	1.32×10^{3}	6.49×10^{4}	7.06×10^{3}	1.10×10^{4}	2.17×10^{3}	1.28×10^{3}	1.45×10^{3}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$3.19 \times 10^{-7+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_{13}	Median	4.64×10^{2}	7.29×10^{2}	4.50×10^{2}	7.82×10^{2}	4.68×10^{2}	8.87×10^{2}	1.39×10^{3}	1.65×10^{3}	8.25×10^{2}	8.17×10^{2}
	Mean	5.15×10^{2}	7.54×10^{2}	5.48×10^{2}	7.91×10^{2}	5.53×10^{2}	9.81×10^{2}	1.42×10^{3}	1.77×10^{3}	8.24×10^{2}	8.40×10^{2}
	Std	1.49×10^{2}	1.07×10^{2}	1.66×10^{2}	2.37×10^{2}	1.75×10^{2}	3.86×10^{2}	3.40×10^{2}	5.06×10^{2}	1.35×10^{2}	1.98×10^{2}
	p-value	-	$5.90 \times 10^{-5+}$	$3.49 \times 10^{-3+}$	$4.32 \times 10^{-8+}$	$2.73 \times 10^{-1=}$	$2.85 \times 10^{-2+}$	$3.49 \times 10^{-3+}$	$5.90 \times 10^{-5+}$	$1.18 \times 10^{-5+}$	$2.07 \times 10^{-6+}$
F_{14}	Median	5.10×10^{7}	1.29×10^{8}	1.61×10^{8}	1.23×10^{8}	2.46×10^{8}	8.61×10^{7}	8.59×10^{8}	8.71×10^{8}	7.19×10^{8}	7.18×10^{8}
	Mean	5.17×10^{7}	1.32×10^{8}	1.60×10^{8}	1.22×10^{8}	2.46×10^{8}	8.55×10^{7}	8.64×10^{8}	8.60×10^{8}	7.23×10^{8}	7.25×10^{8}
	Std	2.76×10^{6}	9.33×10^{6}	8.42×10^{6}	6.41×10^{6}	1.29×10^{7}	7.57×10^{6}	3.30×10^{7}	4.17×10^{7}	3.65×10^{7}	3.44×10^{7}
	p-value	-	$4.32 \times 10^{-8+}$	$2.07 \times 10^{-6+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$

Table 2. *Cont.*

F	Quality	DGCELSO	TPLSO	SPLSO	LLSO	CSO	SLPSO	DECC-GDG	DECC-DG2	DECC-RDG	DECC-RDG2
F_{15}	Median	1.04×10^{4}	1.04×10^{4}	9.92×10^{3}	8.30×10^{2}	1.01×10^{4}	1.12×10^{4}	6.75×10^{3}	6.73×10^{3}	6.55×10^{3}	6.56×10^{3}
	Mean	1.04×10^{4}	8.88×10^{3}	9.91×10^{3}	8.97×10^{2}	1.01×10^{4}	1.12×10^{4}	6.76×10^{3}	6.73×10^{3}	6.55×10^{3}	6.55×10^{3}
	Std	6.65×10^{1}	3.41×10^{3}	6.31×10^{1}	3.47×10^{2}	6.48×10^{1}	1.19×10^{2}	8.82×10^{1}	7.27×10^{1}	8.86×10^{1}	8.39×10^{1}
	p-value	-	$1.44 \times 10^{-1=}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$1.06 \times 10^{-2-}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$
F_{16}	Median	1.55×10^{-13}	1.78×10^{1}	4.66×10^{-12}	4.40×10^{0}	5.64×10^{-8}	2.12×10^{1}	3.98×10^{-4}	3.89×10^{-4}	1.92×10^{-5}	1.88×10^{-5}
	Mean	1.55×10^{-13}	1.89×10^{1}	4.68×10^{-12}	4.33×10^{0}	5.68×10^{-8}	2.36×10^{1}	3.97×10^{-4}	3.90×10^{-4}	1.93×10^{-5}	1.89×10^{-5}
	Std	2.66×10^{-15}	7.46×10^{0}	4.41×10^{-13}	2.50×10^{0}	6.21×10^{-9}	1.11×10^{1}	1.44×10^{-5}	1.33×10^{-5}	8.87×10^{-7}	8.30×10^{-7}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_{17}	Median	6.70×10^{4}	9.65×10^{4}	6.90×10^{5}	9.17×10^{4}	2.19×10^{6}	8.64×10^{4}	2.64×10^{5}	2.64×10^{5}	1.99×10^{5}	1.97×10^{5}
	Mean	6.57×10^{4}	9.83×10^{4}	6.84×10^{5}	9.12×10^{4}	2.21×10^{6}	8.74×10^{4}	2.65×10^{5}	2.63×10^{5}	1.98×10^{5}	1.98×10^{5}
	Std	7.55×10^{3}	9.90×10^{3}	3.57×10^{4}	5.43×10^{3}	2.07×10^{5}	1.39×10^{4}	7.79×10^{3}	7.33×10^{3}	8.75×10^{3}	9.45×10^{3}
	p-value	-	$4.32 \times 10^{-8+}$	$7.15 \times 10^{-2+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_{18}	Median	1.25×10^{3}	2.29×10^{3}	1.25×10^{3}	2.49×10^{3}	1.38×10^{3}	2.95×10^{3}	1.15×10^{3}	1.14×10^{3}	1.08×10^{3}	1.11×10^{3}
	Mean	1.31×10^{3}	2.36×10^{3}	1.35×10^{3}	2.51×10^{3}	1.64×10^{3}	2.92×10^{3}	1.16×10^{3}	1.13×10^{3}	1.07×10^{3}	1.10×10^{3}
	Std	2.94×10^{2}	4.19×10^{2}	3.81×10^{2}	7.42×10^{2}	8.13×10^{2}	8.08×10^{2}	1.31×10^{2}	1.29×10^{2}	1.08×10^{2}	1.02×10^{2}
	p-value	-	$5.90 \times 10^{-5+}$	$3.49 \times 10^{-3+}$	$2.61 \times 10^{-4+}$	$2.73 \times 10^{-1=}$	$2.85 \times 10^{-2+}$	$3.19 \times 10^{-7-}$	$3.19 \times 10^{-7-}$	$3.19 \times 10^{-7-}$	$4.32 \times 10^{-8-}$
F_{19}	Median	1.02×10^{7}	3.94×10^{6}	8.19×10^{6}	1.85×10^{6}	9.78×10^{6}	5.20×10^{6}	2.11×10^{6}	2.09×10^{6}	1.96×10^{6}	1.93×10^{6}
	Mean	1.02×10^{7}	3.89×10^{6}	8.20×10^{6}	1.82×10^{6}	9.86×10^{6}	5.23×10^{6}	2.12×10^{6}	2.10×10^{6}	1.95×10^{6}	1.92×10^{6}
	Std	7.69×10^{5}	2.64×10^{5}	4.61×10^{5}	9.22×10^{4}	5.07×10^{5}	9.15×10^{5}	8.77×10^{4}	9.92×10^{4}	7.80×10^{4}	1.05×10^{5}
	p-value	-	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$
F_{20}	Median	1.06×10^{3}	2.04×10^{3}	9.79×10^{2}	1.88×10^{3}	9.87×10^{2}	1.73×10^{3}	5.43×10^{3}	5.33×10^{3}	4.32×10^{3}	4.25×10^{3}
	Mean	1.08×10^{3}	2.08×10^{3}	1.06×10^{3}	1.92×10^{3}	1.07×10^{3}	1.73×10^{3}	5.45×10^{3}	5.46×10^{3}	4.28×10^{3}	4.34×10^{3}
	Std	7.30×10^{1}	2.00×10^{2}	1.75×10^{2}	3.00×10^{2}	1.70×10^{2}	1.53×10^{2}	3.32×10^{2}	3.37×10^{2}	2.29×10^{2}	3.20×10^{2}
	p-value	-	$4.32 \times 10^{-8+}$	$5.90 \times 10^{-5-}$	$4.32 \times 10^{-8+}$	$5.90 \times 10^{-5-}$	$1.18 \times 10^{-5+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
	w/t/l		16/2/2	14/1/5	15/0/5	15/2/3	18/0/2	17/0/3	16/0/4	16/0/4	16/0/4
Rank		2.75	4.80	4.65	3.70	6.25	6.05	8.20	7.10	5.85	5.65

Table 3. Fitness comparison between DECELSO and the compared algorithms on the 1000-D CEC'2013 problems with 3×10^6 fitness evaluations.

F	Quality	DGCELSO	TPLSO	SPLSO	LLSO	CSO	SLPSO	DECC-GDG	DECC-DG2	DECC-RDG	DECC-RDG2
F_1	Median	0.00×10^0	3.21×10^{-18}	1.17×10^{-19}	4.02×10^{-22}	7.92×10^{-12}	1.03×10^{-17}	7.06×10^0	3.46×10^0	2.04×10^{-2}	2.96×10^{-2}
	Mean	0.00×10^0	3.81×10^{-18}	1.18×10^{-19}	4.28×10^{-22}	7.88×10^{-12}	1.65×10^{-17}	7.43×10^0	6.31×10^0	3.51×10^{-2}	1.08×10^{-1}
	Std	0.00×10^0	1.57×10^{-18}	1.04×10^{-20}	1.29×10^{-22}	1.19×10^{-12}	3.25×10^{-17}	9.38×10^{-1}	7.78×10^0	3.88×10^{-2}	2.08×10^{-1}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$9.63 \times 10^{-7+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_2	Median	8.61×10^2	1.30×10^3	9.64×10^2	1.14×10^3	8.58×10^3	2.09×10^3	1.43×10^3	7.81×10^3	7.81×10^3	7.69×10^3
	Mean	8.77×10^2	1.34×10^3	1.06×10^3	1.14×10^3	8.58×10^3	2.10×10^3	1.43×10^3	7.88×10^3	7.74×10^3	7.74×10^3
	Std	4.28×10^1	1.75×10^2	4.38×10^2	5.00×10^1	1.76×10^2	1.61×10^2	2.43×10^1	4.07×10^2	3.47×10^2	3.56×10^2
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_3	Median	2.16×10^1	2.22×10^1	2.16×10^1	2.16×10^1	2.16×10^1	2.16×10^1	2.15×10^1	2.15×10^1	2.14×10^1	2.15×10^1
	Mean	2.16×10^1	2.31×10^1	2.16×10^1	2.16×10^1	2.16×10^1	2.16×10^1	2.15×10^1	2.15×10^1	2.14×10^1	2.15×10^1
	Std	6.26×10^{-3}	1.72×10^0	7.11×10^{-15}	7.11×10^{-15}	7.11×10^{-15}	2.37×10^{-1}	3.00×10^{-2}	4.23×10^{-2}	4.82×10^{-2}	4.90×10^{-2}
	p-value	-	$3.49 \times 10^{-3+}$	$2.61 \times 10^{-4-}$	$2.61 \times 10^{-4-}$	$2.07 \times 10^{-6-}$	$7.15 \times 10^{-1=}$	$4.65 \times 10^{-1=}$	$4.65 \times 10^{-1=}$	$7.15 \times 10^{-1=}$	$7.15 \times 10^{-1=}$
F_4	Median	2.55×10^9	4.23×10^9	9.14×10^9	6.40×10^9	1.22×10^{10}	4.28×10^9	4.15×10^{11}	8.12×10^{10}	7.45×10^{10}	6.10×10^{10}
	Mean	2.52×10^9	4.27×10^9	9.41×10^9	6.55×10^9	1.35×10^{10}	4.33×10^9	4.20×10^{11}	7.79×10^{10}	7.16×10^{10}	6.78×10^{10}
	Std	6.55×10^8	1.03×10^9	1.86×10^9	1.40×10^9	3.12×10^9	9.91×10^8	7.75×10^{10}	2.19×10^{10}	1.92×10^{10}	2.32×10^{10}
	p-value	-	$1.44 \times 10^{-1=}$	$1.02 \times 10^{-3+}$	$1.06 \times 10^{-2+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_5	Median	7.83×10^5	6.80×10^5	6.43×10^5	6.51×10^5	5.90×10^5	8.89×10^5	8.62×10^6	6.10×10^6	5.81×10^6	5.72×10^6
	Mean	7.91×10^5	6.79×10^5	6.30×10^5	6.56×10^5	5.97×10^5	8.90×10^5	8.66×10^6	6.06×10^6	5.72×10^6	5.67×10^6
	Std	1.03×10^5	1.10×10^5	1.00×10^5	1.01×10^5	1.03×10^5	1.31×10^5	2.80×10^5	2.40×10^5	4.24×10^5	3.61×10^5
	p-value	-	$1.06 \times 10^{-2-}$	$1.18 \times 10^{-5-}$	$3.49 \times 10^{-3-}$	$2.61 \times 10^{-4-}$	$2.85 \times 10^{-2+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_6	Median	1.06×10^6	1.17×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6
	Mean	1.06×10^6	1.22×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6	1.06×10^6
	Std	1.27×10^3	1.61×10^5	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0	3.00×10^3	0.00×10^0	0.00×10^0	0.00×10^0
	p-value	-	$5.90 \times 10^{-5+}$	$4.65 \times 10^{-1=}$	$1.18 \times 10^{-5-}$	$2.73 \times 10^{-1=}$	$2.61 \times 10^{-4-}$	$4.65 \times 10^{-1=}$	$1.44 \times 10^{-1=}$	$2.73 \times 10^{-1=}$	$2.85 \times 10^{-2-}$
F_7	Median	7.93×10^4	1.22×106	5.42×106	1.70×106	5.45×106	1.47×106	7.45×108	7.36×107	2.84×108	8.36×107
	Mean	9.71×10^4	1.24×106	5.50×106	1.87×106	5.81×106	1.58×106	7.67×108	7.79×107	3.65×108	8.25×107
	Std	5.54×10^4	5.05×105	2.23×106	1.08×106	3.04×106	7.53×105	1.32×108	2.73×107	2.63×108	2.06×107
	p-value	-	$1.18 \times 10-5^{+}$	$1.18 \times 10-5^{+}$	$4.32 \times 10-8^{+}$	$2.61 \times 10-4^{+}$	$3.19 \times 10-7^{+}$	$4.32 \times 10-8^{+}$	$4.32 \times 10-8^{+}$	$4.32 \times 10-8^{+}$	$4.32 \times 10-8^{+}$
F_8	Median	5.58×10^{13}	7.07×10^{13}	1.56×10^{14}	1.37×10^{14}	2.43×10^{14}	9.65×10^{13}	1.70×10^{16}	9.35×10^{15}	6.96×10^{15}	5.83×10^{15}
	Mean	6.15×10^{13}	7.28×10^{13}	1.55×10^{14}	1.36×10^{14}	2.46×10^{14}	1.09×10^{14}	1.65×10^{16}	9.32×10^{15}	6.95×10^{15}	6.38×10^{15}
	Std	2.08×10^{13}	4.02×10^{13}	2.92×10^{13}	3.39×10^{13}	8.71×10^{13}	5.44×10^{13}	4.49×10^{15}	2.71×10^{15}	1.64×10^{15}	1.99×10^{15}
	p-value	-	$5.90 \times 10^{-5+}$	$1.44 \times 10^{-1=}$	$2.85 \times 10^{-2-}$	$1.18 \times 10^{-5-}$	$3.49 \times 10^{-3-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$

Table 4. Fitness comparison between DECELSO and the compared algorithms on the 1000-D CEC'2013 problems with 3 × 10^6 fitness evaluations.

F	Quality	DGCELSO	TPLSO	SPLSO	LLSO	CSO	SLPSO	DECC-GDG	DECC-DG2	DECC-RDG	DECC-RDG2
F_9	Median	4.67×10^{7}	4.52×10^{7}	7.23×10^{7}	1.11×10^{8}	5.94×10^{7}	8.05×10^{7}	5.62×10^{8}	5.55×10^{8}	5.40×10^{8}	5.32×10^{8}
	Mean	4.47×10^{7}	4.28×10^{7}	8.08×10^{7}	1.29×10^{8}	6.08×10^{7}	7.99×10^{7}	5.61×10^{8}	5.59×10^{8}	5.38×10^{8}	5.31×10^{8}
	Std	1.37×10^{7}	7.49×10^{6}	2.21×10^{7}	8.85×10^{7}	1.29×10^{7}	1.18×10^{7}	3.24×10^{7}	2.93×10^{7}	3.03×10^{7}	2.33×10^{7}
	p-value	-	$4.65 \times 10^{-1=}$	$3.19 \times 10^{-7+}$	$4.32 \times 10^{-8+}$	$2.61 \times 10^{-4+}$	$3.19 \times 10^{-7+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_{10}	Median	9.40×10^{7}	9.44×10^{7}	9.40×10^{7}	9.41×10^{7}	9.41×10^{7}	9.37×10^{7}	9.46×10^{7}	9.46×10^{7}	9.46×10^{7}	9.45×10^{7}
	Mean	9.40×10^{7}	9.52×10^{7}	9.39×10^{7}	9.41×10^{7}	9.40×10^{7}	9.27×10^{7}	9.46×10^{7}	9.46×10^{7}	9.46×10^{7}	9.45×10^{7}
	Std	2.95×10^{5}	1.70×10^{6}	2.18×10^{5}	2.23×10^{5}	2.14×10^{5}	1.99×10^{6}	2.57×10^{5}	2.51×10^{5}	1.98×10^{5}	2.78×10^{5}
	p-value	-	$1.02 \times 10^{-3+}$	$6.79 \times 10^{-2=}$	$1.18 \times 10^{-5+}$	$2.07 \times 10^{-6+}$	$1.02 \times 10^{-3-}$	$3.19 \times 10^{-7+}$	$3.19 \times 10^{-7+}$	$4.32 \times 10^{-8+}$	$2.07 \times 10^{-6+}$
F_{11}	Median	6.44×10^{7}	1.88×10^{8}	9.22×10^{11}	9.23×10^{11}	9.26×10^{11}	9.38×10^{11}	6.80×10^{8}	1.99×10^{10}	5.75×10^{8}	1.33×10^{10}
	Mean	7.14×10^{7}	1.83×10^{8}	9.27×10^{11}	9.28×10^{11}	9.29×10^{11}	9.34×10^{11}	6.84×10^{8}	2.52×10^{10}	5.68×10^{8}	1.49×10^{10}
	Std	2.45×10^{7}	5.62×10^{7}	9.35×10^{9}	9.68×10^{9}	9.63×10^{9}	8.96×10^{9}	1.09×10^{8}	1.38×10^{10}	9.23×10^{7}	7.57×10^{9}
	p-value	-	$3.49 \times 10^{-3+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$2.61 \times 10^{-4+}$	$4.32 \times 10^{-8+}$	$3.49 \times 10^{-3+}$	$4.32 \times 10^{-8+}$
F_{12}	Median	1.12×10^{3}	2.19×10^{3}	1.03×10^{3}	1.80×10^{3}	1.04×10^{3}	1.76×10^{3}	5.54×10^{3}	5.42×10^{3}	4.28×10^{3}	4.25×10^{3}
	Mean	1.14×10^{3}	2.13×10^{3}	1.05×10^{3}	1.82×10^{3}	1.08×10^{3}	1.77×10^{3}	5.51×10^{3}	5.59×10^{3}	4.34×10^{3}	4.30×10^{3}
	Std	9.96×10^{1}	2.72×10^{2}	5.45×10^{1}	1.52×10^{2}	7.45×10^{1}	1.69×10^{2}	3.67×10^{2}	7.64×10^{2}	3.24×10^{2}	2.48×10^{2}
	p-value	-	$4.32 \times 10^{-8+}$	$2.85 \times 10^{-2-}$	$4.32 \times 10^{-8+}$	$1.00 \times 10^{0=}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$
F_{13}	Median	4.89×10^{7}	2.01×10^{8}	1.20×10^{9}	2.98×10^{8}	7.08×10^{8}	4.01×10^{8}	1.56×10^{9}	1.43×10^{9}	2.87×10^{9}	7.08×10^{8}
	Mean	6.40×10^{7}	2.21×10^{8}	1.20×10^{9}	3.42×10^{8}	7.48×10^{8}	5.20×10^{8}	1.50×10^{9}	1.47×10^{9}	2.98×10^{9}	7.17×10^{8}
	Std	5.35×10^{7}	1.24×10^{8}	4.91×10^{8}	1.42×10^{8}	2.85×10^{8}	4.85×10^{8}	3.35×10^{8}	3.46×10^{8}	7.23×10^{8}	1.57×10^{8}
	p-value	-	$3.19 \times 10^{-7+}$	$1.00 \times 10^{0=}$	$4.32 \times 10^{-8+}$	$1.02 \times 10^{-3+}$	$4.32 \times 10^{-8+}$	$1.44 \times 10^{-1=}$	$2.73 \times 10^{-1=}$	$4.32 \times 10^{-8+}$	$3.49 \times 10^{-3+}$
F_{14}	Median	1.77×10^{7}	5.86×10^{7}	5.19×10^{9}	8.06×10^{7}	2.90×10^{9}	1.51×10^{8}	4.45×10^{9}	4.54×10^{9}	2.23×10^{9}	2.50×10^{9}
	Mean	1.78×10^{7}	6.05×10^{7}	8.31×10^{9}	1.59×10^{8}	3.67×10^{9}	2.51×10^{8}	5.28×10^{9}	4.58×10^{9}	2.78×10^{9}	3.33×10^{9}
	Std	2.62×10^{6}	1.34×10^{7}	6.56×10^{9}	2.27×10^{8}	3.32×10^{9}	2.25×10^{8}	3.84×10^{9}	1.83×10^{9}	1.85×10^{9}	2.09×10^{9}
	p-value	-	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$2.07 \times 10^{-6+}$	$3.19 \times 10^{-7+}$	$2.85 \times 10^{-2+}$	$6.79 \times 10^{-2=}$	$1.00 \times 10^{0=}$	$6.79 \times 10^{-2=}$
F_{15}	Median	3.53×10^{7}	1.29×10^{7}	4.13×10^{7}	4.58×10^{6}	7.60×10^{7}	5.99×10^{7}	8.60×10^{6}	8.82×10^{6}	7.75×10^{6}	8.04×10^{6}
	Mean	3.54×10^{7}	1.26×10^{7}	4.13×10^{7}	4.59×10^{6}	7.61×10^{7}	6.03×10^{7}	8.98×10^{6}	8.95×10^{6}	7.96×10^{6}	8.07×10^{6}
	Std	7.60×10^{6}	1.36×10^{6}	3.05×10^{6}	3.22×10^{5}	6.14×10^{6}	6.54×10^{6}	8.90×10^{5}	9.38×10^{5}	9.30×10^{5}	9.78×10^{5}
	p-value	-	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8+}$	$4.32 \times 10^{-8+}$	$1.18 \times 10^{-5+}$	$4.65 \times 10^{-1=}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$	$4.32 \times 10^{-8-}$
	w/t/l		11/2/2	8/4/3	12/0/3	11/2/2	11/2/2	11/3/1	10/4/1	11/3/1	11/2/2
Rank		2.73	4.47	4.87	4.13	5.80	5.00	8.13	7.40	6.47	6.00

In Table 2, the comparison results on the CEC'2010 set are summarized as follows. (1) From the perspective of the Friedman test, as shown in the last row, it is found that the proposed DGCELSO has the lowest rank value, which is much smaller than those of the compared algorithms. This means that DGCELSO achieves the best overall performance and shows great superiority to the compared algorithms. (2) With respect to the Wilcoxon rank-sum test, as shown in the second last row, it is observed that DGCELSO performs significantly better than the compared algorithms on at least 14 problems. In particular, competed with the four cooperative coevolutionary evolutionary algorithms, DGCELSO presents significant superiority to them on at least 16 problems and only shows inferiority in at most four problems. In comparison with the five holistic large-scale PSO variants, DGCELSO is significantly superior to SLPSO on 18 problems, achieves much better performance than TPLSO on 16 problems, outperforms both LLSO and CSO on 15 problems, and beats SPLSO down on 14 problems. The superiority of DGCELSO to the five holistic large-scale PSOs demonstrates the effectiveness of the proposed DGCEL strategy.

In Table 4, we summarize the comparison results on the CEC'2013 set as follows. (1) From the perspective of the Friedman test, as shown in the last row, it is found that the rank value of the proposed DGCELSO is still the lowest among the ten algorithms, and such a rank is still much smaller than those of the nine compared algorithms. This demonstrates that DGCELSO still achieves the best overall performance on the complicated CEC'2013 benchmark set and shows great dominance to the compared algorithms. (2) With respect to the Wilcoxon rank-sum test, as shown in the second to last row, it is observed that except for SPLSO, DGCELSO shows significantly better performance than the other eight compared algorithms on at least 10 problems and shows inferiority on at most three problems. Competed with SPLSO, DGCELSO beats it on eight problems and is defeated on only three problems. The superiority of DGCELSO to the compared algorithms on the CEC'2013 benchmark set demonstrates that it is promising for complicated large-scale optimization problems.

The above experiments demonstrated the effectiveness of the proposed DGCELSO. To further demonstrate its efficiency in solving large-scale optimization problems, we conduct experiments on the two large-scale benchmark sets to investigate the convergence speed of the proposed DGCELSO in comparison with the nine compared methods. In this experiment, the maximum number of fitness evaluations is set as 5×10^6. Figures 3 and 4 show the convergence comparison results on the CEC'2010 and the CEC'2013 benchmark sets, respectively.

In Figure 3, on the CEC'2010 benchmark set, the following findings can be obtained. (1) At first glance, it is found that the proposed DGCELSO obviously obtains faster convergence along with better solutions than all the nine compared algorithms on nine problems (F_1, F_4, F_7, F_9, F_{11}, F_{12}, F_{14}, F_{16}, and F_{17}). On F_3, F_{13}, F_{18}, and F_{20}, DGCELSO achieves very similar performance with some compared algorithms in terms of the solution quality but obtains much faster convergence than the associated compared algorithms. (2) More specifically, we find that DGCELSO obviously shows much better performance in both convergence speed and solution quality than the five holistic large-scale PSO variants, namely TPLSO, SPLSO, LLSO, CSO, and SLPSO on 17, 16, 15, 16, and 17, respectively. In the competition with the four cooperative coevolutionary evolutionary algorithms, namely DECC-DG, DECC-GD2, DECC-RDG, and DECC-RDG2, DGCELSO shows clear superiority in both convergence speed and solution quality on 17, 17, 17, and 15 problems, respectively.

From Figure 4, similar observations on the CEC'2013 benchmark set can be attained. (1) At first glance, it is found that the proposed DGCELSO obtains faster convergence along with better solutions than all the nine compared algorithms on six problems (F_1, F_4, F_7, F_{11}, F_{13}, and F_{14}). On F_8, F_9, and F_{12}, DGCELSO shows superiority in both convergence speed and solution quality to eight compared algorithms and is inferior to only one compared algorithm. (2) More specifically, we find that DGCELSO performs better with faster convergence speed and higher solution quality than TPLSO, SPLSO, LLSO, CSO, and SLPSO on 11, 11, 9, 12, and 10 problems, respectively. In competition with DECC-DG, DECC-GD2, DECC-RDG, and DECC-RDG2, DGCELSO presents great dominance to them on 11, 9, 11, and 12 problems, respectively.

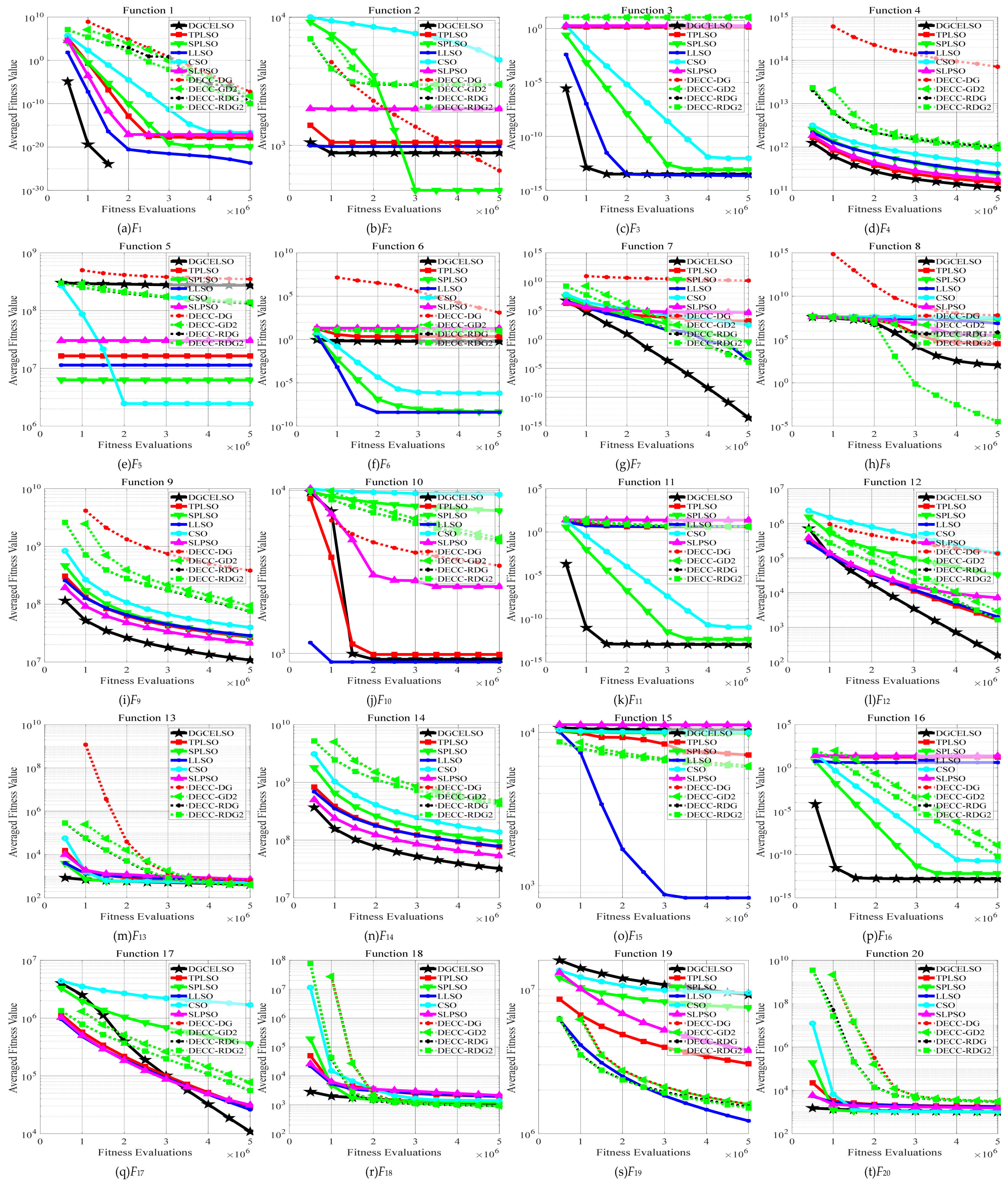


Figure 3. Convergence behavior comparison between DGCELSO and the compared algorithms on each 1000-D CEC'2010 benchmark problem.

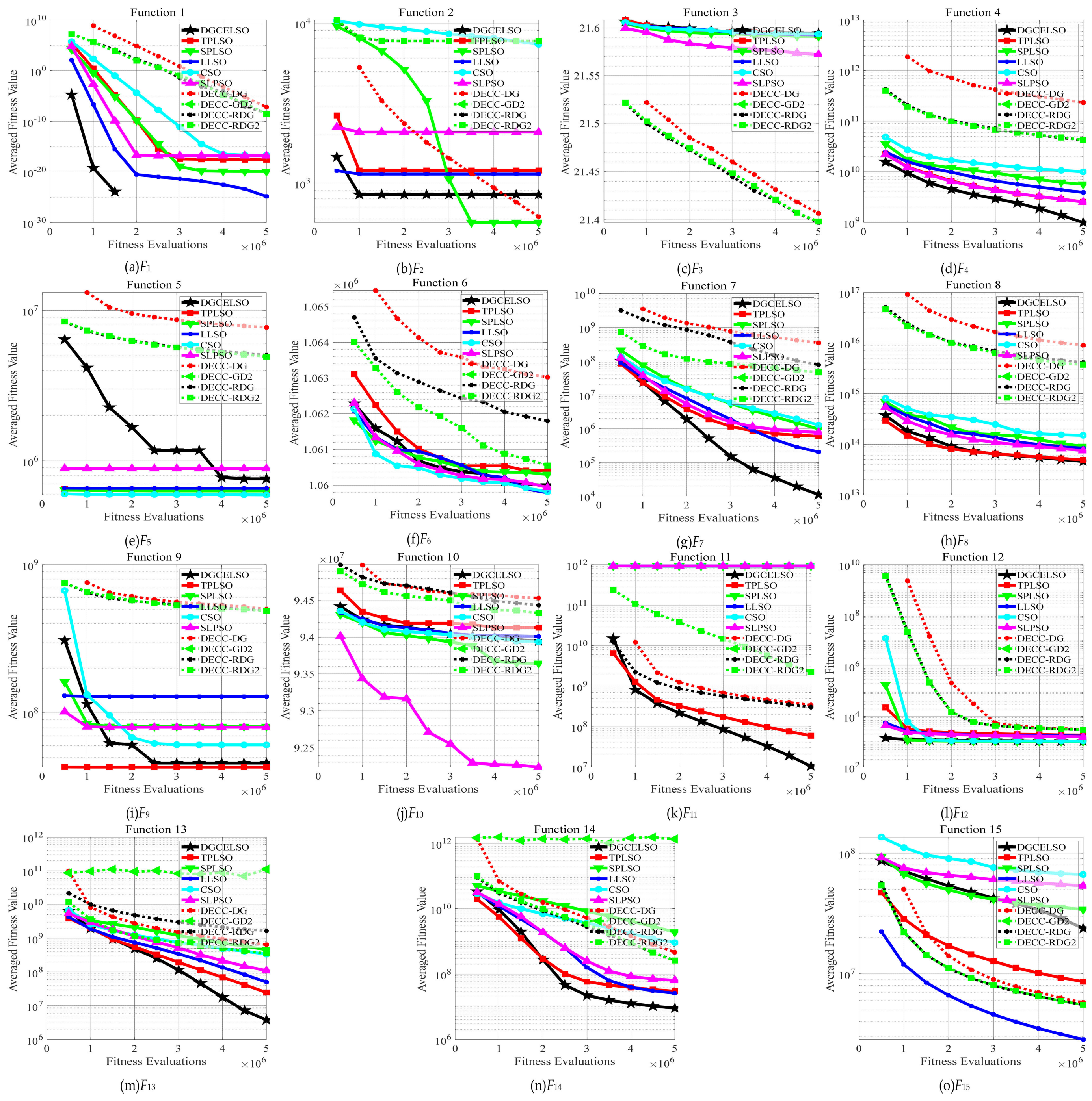


Figure 4. Convergence behavior comparison between DGCELSO and the compared algorithms on each 1000-D CEC'2013 benchmark problem.

To sum up, compared with these state-of-the-art large-scale algorithms, DGCELSO performs much better in both convergence speed and solution quality. The superiority of DGCELSO mainly benefits from the proposed DGCEL strategy, which could implicitly assemble useful information embedded in elite particles to guide the evolution of the swarm. In particular, the superiority of DGCELSO to the five holistic large-scale PSOs, which also adopt elite particles in the current swarm to direct the evolution of the swarm, demonstrates that the assembly of evolutionary information in elites is effective. Such assembly not only improves the learning diversity of particles due to the random selection of guiding exemplars from the elites but also promotes the learning effectiveness of particles because each updated particle could learn from multiple different elites with the help of the dimension group-based learning. As a result, DGCELSO could compromise search intensification and diversification well to explore and exploit the large-scale solution appropriately to locate satisfactory solutions.

4.3. Deep Investigation on DGCELSO

In this section, we conduct extensive experiments on the 1000-D CEC'2010 benchmark set to verify the effectiveness of the main components in the proposed DGCELSO.

4.3.1. Effectiveness of the Proposed DGCEL

First, we conduct experiments to investigate the effectiveness of the proposed DGCEL strategy. To this end, we first incorporate the segment-based predominance learning strategy (SPL) in SPLSO, which is the most similar work to the proposed DGCELSO, to replace the DGCEL strategy, leading to a new variant of DGCELSO, which we denote as "DGCELSO-SPL". In addition, we also develop two extreme cases of DGCELSO, where the number of dimension groups (*NDG*) is set as 1 and 1000, respectively. The former, which we denote as "DGCELSO-1", con all dimensions as a group, and thus can be considered a DGCELSO without the dimension group-based comprehensive learning, while the latter, which we denote as "DGCELSO-1000", considers each dimension as a group. This can be considered a DGCELSO by introducing the comprehensive learning strategy in CLPSO [46] to replace the dimension group-based comprehensive learning in DGCELSO. Then, we conduct experiments on the CEC'2010 benchmark set to compare the above four versions of DGCELSO. Table 5 shows the comparison results among the four versions of DGCELSO. In this table, the best results are highlighted in bold.

From Table 5, the following observations can be attained. (1) From the perspective of the Friedman test, it is found that the rank value of DGCELSO is the smallest among the four versions of DGCELSO. This demonstrates that DGCELSO achieves the best overall performance. (2) Comparing DGCELSO with DGCELSO-SPL, DGCELSO shows great superiority. This demonstrates that the proposed DGCEL strategy is much better than SPL. It should be mentioned that, like DGCEL, SPL also lets each particle learn from multiple elites in the swarm, based on the dimension group. The differences between DGCEL and SPL lie in two aspects. On the one hand, SPL lets particles learn from relatively better elites which are determined by the competition between randomly paired two particles, while DGCEL lets particles learn from absolutely better elites which are the top $tp*NP$ best particles in the swarm. On the other hand, the second exemplar in the velocity update in SPL is the mean position of the whole swarm, which is shared by all updated particles, while the second exemplar in DGCEL is also randomly selected from the elite particles. With the observed superiority of DGCEL to SPL, it is demonstrated that the exemplar selection in DGCEL is better than that in SPL. (3) Competed with DGCELSO-1 and DGCELSO-1000, DGCELSO presents great superiority. This superiority demonstrates the effectiveness of the proposed dimension group-based comprehensive learning strategy. Instead of learning from only two exemplars in DGCELSO-1, which consider all dimensions as a group, and learning from multiple exemplars dimension by dimension in DGCELSO-1000, which considers each dimension as a group, DGCELSO lets each updated particle learn from multiple exemplars based on dimension group. In this way, the potentially useful information embedded in different exemplars is more likely to be assembled in DGCELSO than in DGCELSO-1 and DGCELSO-1000.

Based on the above observations, it is found that the proposed DGCEL strategy is effective and plays a crucial role in helping DGCELSO achieve promising performance.

4.3.2. Effectiveness of the Proposed Dynamic Adjustment Schemes for Parameters

In this subsection, we conduct experiments to verify the effectiveness of the proposed dynamic adjustment schemes for the two control parameters, namely the elite ratio *tp* and the number of dimension groups *NDG*.

Table 5. Comparison results among different versions of DGCELSO on the 1000-D CEC'2010 problems.

F	**DGCELSO**	**DGCELSO-1**	**DGCELSO-1000**	**DGCELSO-SPL**
F_1	0.00×10^{0}	3.85×10^{-26}	0.00×10^{0}	1.81×10^{3}
F_2	8.88×10^{2}	1.98×10^{3}	8.70×10^{2}	1.54×10^{3}
F_3	3.18×10^{-14}	1.08×10^{0}	3.16×10^{-14}	1.97×10^{-2}
F_4	1.60×10^{11}	2.15×10^{11}	1.56×10^{11}	9.44×10^{11}
F_5	2.80×10^{8}	6.93×10^{7}	2.79×10^{8}	1.07×10^{7}
F_6	4.00×10^{-9}	1.96×10^{1}	4.00×10^{-9}	3.74×10^{-1}
F_7	2.15×10^{-5}	4.01×10^{3}	2.17×10^{-5}	6.15×10^{6}
F_8	4.36×10^{3}	6.84×10^{5}	4.26×10^{3}	3.27×10^{7}
F_9	1.77×10^{7}	3.28×10^{7}	1.77×10^{7}	1.05×10^{8}
F_{10}	9.23×10^{2}	2.02×10^{3}	9.34×10^{2}	3.63×10^{3}
F_{11}	1.10×10^{-13}	2.08×10^{1}	1.10×10^{-13}	6.66×10^{-1}
F_{12}	2.55×10^{3}	4.60×10^{3}	2.63×10^{3}	1.99×10^{5}
F_{13}	5.15×10^{2}	7.69×10^{2}	4.87×10^{2}	1.42×10^{3}
F_{14}	5.17×10^{7}	9.78×10^{7}	5.13×10^{7}	3.42×10^{8}
F_{15}	1.04×10^{4}	2.04×10^{3}	1.05×10^{4}	1.00×10^{4}
F_{16}	1.55×10^{-13}	2.92×10^{1}	2.93×10^{-2}	5.72×10^{-1}
F_{17}	6.57×10^{4}	4.30×10^{4}	7.12×10^{4}	7.10×10^{5}
F_{18}	1.31×10^{3}	2.30×10^{3}	1.33×10^{3}	2.38×10^{4}
F_{19}	1.02×10^{7}	1.33×10^{6}	1.06×10^{7}	6.52×10^{6}
F_{20}	1.08×10^{3}	1.98×10^{3}	1.08×10^{3}	2.11×10^{4}
Rank	1.80	2.90	1.90	3.40

First, we conduct experiments to investigate the effectiveness of the proposed dynamic scheme for *tp*. To this end, we first set *tp* as different fixed values from 0.1 to 0.9. Then, we compare the DGCELSO with the dynamic scheme with these DGCELSOs with different fixed *tp* values. Table 6 shows the comparison results between the DGCELSO with the dynamic scheme and the ones with different values of *tp* on the CEC'2010 benchmark set. In this table, the best results are highlighted in bold.

From Table 6, the following findings can be obtained. (1) From the perspective of the Friedman test, it is found that DGCELSO with the dynamic *tp* ranks first among all versions of DGCELSO with different settings of *tp*. This demonstrates that DGCELSO with the dynamic *tp* achieves the best overall performance. (2) More specifically, we find that DGCELSO with the dynamic strategy obtains the best results on 4 problems and its results on the other problems are very close to the best ones obtained by the DGCELSO with the associated optimal settings of *tp*. These two observations demonstrate that the dynamic strategy for *tp* is helpful in achieving good performance for DGCELSO.

Then, we conduct experiments to verify the dynamic scheme for the number of dimension groups (*NDG*). To this end, we first set *NDG* as different fixed values from 20 to 100. Subsequently, we conduct experiments on the CEC'2010 set to compare the DGCELSO with the dynamic scheme for *NDG* and the ones with different fixed values of *NDG*. Table 7 shows the comparison results among the above versions of DGCELSO. In this table, the best results are highlighted in bold.

Table 6. Comparison results between DGCELSO with the dynamic strategy for *tp* and the ones with different fixed settings of *tp* on the 1000-D CEC'2010 problems.

F	***tp* = 0.1**	***tp* = 0.2**	***tp* = 0.3**	***tp* = 0.4**	***tp* = 0.5**	***tp* = 0.6**	***tp* = 0.7**	***tp* = 0.8**	***tp* = 0.9**	***Dynamic***
F_1	9.55×10^{-3}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	0.00×10^{0}	3.31×10^{-26}	0.00×10^{0}
F_2	2.33×10^{3}	1.38×10^{3}	1.05×10^{3}	8.21×10^{2}	6.71×10^{2}	1.03×10^{3}	9.26×10^{3}	9.83×10^{3}	1.00×10^{4}	8.88×10^{2}
F_3	1.41×10^{0}	3.30×10^{-14}	3.17×10^{-14}	3.14×10^{-14}	2.98×10^{-14}	2.96×10^{-14}	2.99×10^{-14}	2.93×10^{-14}	2.98×10^{-14}	3.18×10^{-14}
F_4	6.60×10^{11}	1.64×10^{11}	1.80×10^{11}	1.89×10^{11}	2.01×10^{11}	2.24×10^{11}	2.28×10^{11}	2.52×10^{11}	2.53×10^{11}	1.60×10^{11}
F_5	5.90×10^{7}	2.64×10^{8}	2.75×10^{8}	2.76×10^{8}	2.83×10^{8}	2.79×10^{8}	2.81×10^{8}	2.82×10^{8}	2.83×10^{8}	2.80×10^{8}
F_6	1.99×10^{1}	2.00×10^{1}	1.98×10^{1}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	3.88×10^{-9}	4.00×10^{-9}	4.00×10^{-9}
F_7	1.15×10^{6}	9.66×10^{-8}	5.36×10^{-5}	8.46×10^{-3}	3.32×10^{-1}	2.98×10^{0}	1.56×10^{1}	8.23×10^{1}	3.67×10^{2}	2.15×10^{-5}
F_8	4.15×10^{7}	1.10×10^{3}	7.82×10^{3}	1.19×10^{5}	2.07×10^{6}	6.99×10^{6}	1.07×10^{7}	1.36×10^{7}	1.57×10^{7}	4.36×10^{3}
F_9	1.21×10^{8}	1.93×10^{7}	1.85×10^{7}	1.78×10^{7}	2.05×10^{7}	2.03×10^{7}	2.21×10^{7}	2.17×10^{7}	2.34×10^{7}	1.77×10^{7}
F_{10}	2.44×10^{3}	1.53×10^{3}	1.06×10^{3}	2.19×10^{3}	9.48×10^{3}	9.80×10^{3}	1.01×10^{4}	1.02×10^{4}	1.02×10^{4}	9.23×10^{2}
F_{11}	2.86×10^{1}	2.04×10^{1}	1.05×10^{1}	1.11×10^{-13}	1.09×10^{-13}	1.11×10^{-13}	1.11×10^{-13}	1.13×10^{-13}	1.15×10^{-13}	1.10×10^{-13}
F_{12}	1.68×10^{5}	1.12×10^{3}	2.48×10^{3}	7.33×10^{3}	2.81×10^{4}	1.16×10^{5}	5.74×10^{5}	1.53×10^{6}	2.08×10^{6}	2.55×10^{3}
F_{13}	1.68×10^{3}	4.27×10^{2}	5.13×10^{2}	4.12×10^{2}	6.18×10^{2}	4.30×10^{2}	4.50×10^{2}	4.88×10^{2}	5.19×10^{2}	5.15×10^{2}
F_{14}	3.74×10^{8}	5.88×10^{7}	5.39×10^{7}	5.68×10^{7}	5.82×10^{7}	6.54×10^{7}	6.97×10^{7}	8.12×10^{7}	8.92×10^{7}	5.17×10^{7}
F_{15}	2.66×10^{3}	1.08×10^{4}	1.05×10^{4}	1.04×10^{4}	1.04×10^{4}	1.04×10^{4}	1.04×10^{4}	1.04×10^{4}	1.04×10^{4}	1.04×10^{4}
F_{16}	7.57×10^{1}	5.55×10^{0}	1.62×10^{-13}	1.64×10^{-13}	1.68×10^{-13}	1.76×10^{-13}	1.79×10^{-13}	1.87×10^{-13}	1.94×10^{-13}	1.55×10^{-13}
F_{17}	5.02×10^{5}	2.01×10^{4}	5.70×10^{4}	1.86×10^{6}	3.51×10^{6}	4.29×10^{6}	4.92×10^{6}	5.16×10^{6}	5.46×10^{6}	6.57×10^{4}
F_{18}	4.17×10^{3}	1.45×10^{3}	1.35×10^{3}	1.45×10^{3}	1.09×10^{3}	1.43×10^{3}	1.16×10^{3}	1.12×10^{3}	1.16×10^{3}	1.31×10^{3}
F_{19}	2.18×10^{6}	6.26×10^{6}	1.05×10^{7}	1.20×10^{7}	1.32×10^{7}	1.39×10^{7}	1.42×10^{7}	1.50×10^{7}	1.53×10^{7}	1.02×10^{7}
F_{20}	3.09×10^{3}	1.30×10^{3}	1.19×10^{3}	1.10×10^{3}	1.06×10^{3}	1.03×10^{3}	1.02×10^{3}	9.94×10^{2}	9.86×10^{2}	1.08×10^{3}
Rank	7.75	4.98	4.53	4.23	4.85	5.35	5.9	6.28	7.73	3.43

Table 7. Comparison results between DGCELSO with the dynamic strategy for *NDG* and the ones with different fixed settings of *NDG* on the 1000-D CEC'2010 problems.

F	*NDG* = 20	*NDG* = 30	*NDG* = 40	*NDG* = 50	*NDG* = 60	*NDG* = 70	*NDG* = 80	*NDG* = 90	*NDG* = 100	*Dynamic*
F_1	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0	0.00×10^0
F_2	8.46×10^2	8.56×10^2	8.49×10^2	8.41×10^2	8.53×10^2	8.48×10^2	8.52×10^2	8.51×10^2	8.44×10^2	8.88×10^2
F_3	3.18×10^{-14}	3.18×10^{-14}	3.19×10^{-14}	3.22×10^{-14}	3.21×10^{-14}	3.24×10^{-14}	3.19×10^{-14}	3.21×10^{-14}	3.19×10^{-14}	3.18×10^{-14}
F_4	1.69×10^{11}	1.69×10^{11}	1.68×10^{11}	1.65×10^{11}	1.54×10^{11}	1.56×10^{11}	1.65×10^{11}	1.58×10^{11}	1.65×10^{11}	1.60×10^{11}
F_5	2.78×10^8	2.79×10^8	2.80×10^8	2.78×10^8	2.78×10^8	2.78×10^8	2.79×10^8	2.78×10^8	2.79×10^8	2.80×10^8
F_6	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	4.00×10^{-9}	3.88×10^{-9}	4.00×10^{-9}
F_7	2.36×10^{-5}	3.22×10^{-5}	2.64×10^{-5}	2.58×10^{-5}	2.18×10^{-5}	1.92×10^{-5}	2.22×10^{-5}	2.00×10^{-5}	2.93×10^{-5}	2.15×10^{-5}
F_8	5.49×10^3	5.20×10^3	5.15×10^3	5.12×10^3	5.14×10^3	4.98×10^3	5.07×10^3	5.01×10^3	5.07×10^3	4.36×10^3
F_9	1.80×10^7	1.78×10^7	1.73×10^7	1.81×10^7	1.74×10^7	1.76×10^7	1.78×10^7	1.74×10^7	1.82×10^7	1.77×10^7
F_{10}	8.97×10^2	8.94×10^2	8.94×10^2	8.94×10^2	8.92×10^2	8.92×10^2	9.02×10^2	9.14×10^2	8.89×10^2	9.23×10^2
F_{11}	1.11×10^{-13}	1.11×10^{-13}	1.11×10^{-13}	1.11×10^{-13}	1.11×10^{-13}	1.10×10^{-13}	1.10×10^{-13}	1.11×10^{-13}	1.11×10^{-13}	1.10×10^{-13}
F_{12}	3.13×10^3	3.13×10^3	3.24×10^3	3.18×10^3	3.24×10^3	3.21×10^3	3.11×10^3	3.14×10^3	3.28×10^3	2.55×10^3
F_{13}	4.48×10^2	5.03×10^2	5.13×10^2	4.83×10^2	5.30×10^2	5.09×10^2	4.51×10^2	4.64×10^2	4.82×10^2	5.15×10^2
F_{14}	5.26×10^7	5.26×10^7	5.24×10^7	5.16×10^7	5.16×10^7	5.07×10^7	5.23×10^7	5.18×10^7	5.24×10^7	5.17×10^7
F_{15}	1.04×10^4	1.04×10^4	1.04×10^4	1.04×10^4	1.04×10^4	1.04×10^4	1.04×10^4	1.04×10^4	1.04×10^4	1.04×10^4
F_{16}	1.59×10^{-13}	1.57×10^{-13}	1.58×10^{-13}	1.58×10^{-13}	3.85×10^{-2}	1.58×10^{-13}	2.93×10^{-2}	1.59×10^{-13}	1.59×10^{-13}	1.55×10^{-13}
F_{17}	1.19×10^5	1.20×10^5	1.19×10^5	1.20×10^5	1.28×10^5	1.22×10^5	1.28×10^5	1.34×10^5	1.35×10^5	6.57×10^4
F_{18}	1.19×10^3	1.21×10^3	1.28×10^3	1.31×10^3	1.24×10^3	1.18×10^3	1.31×10^3	1.31×10^3	1.28×10^3	1.31×10^3
F_{19}	1.10×10^7	1.09×10^7	1.12×10^7	1.10×10^7	1.12×10^7	1.13×10^7	1.12×10^7	1.12×10^7	1.11×10^7	1.02×10^7
F_{20}	1.12×10^3	1.11×10^3	1.09×10^3	1.11×10^3	1.08×10^3	1.11×10^3	1.09×10^3	1.08×10^3	1.10×10^3	1.08×10^3
Rank	6.05	6.05	6.18	5.33	5.78	4.70	5.50	5.18	6.00	4.25

From Table 7, we can obtain the following findings. (1) From the perspective of the Friedman test, it is found that the rank value of the DGCELSO with the dynamic scheme for *NDG* is the smallest among all versions of DGCELSO with different settings of *NDG*. This demonstrates that DGCELSO with the dynamic strategy achieves the best overall performance. (2) More specifically, we find that DGCELSO with the dynamic strategy obtains the best results on nine problems, while DGCELSO with fixed *NDG* obtains the best results on at most four problems. In particular, on the other 11 problems where DGCELSO with the dynamic strategy does not achieve the best results, its optimization results are very close to the best ones obtained by DGCELSO with the associated optimal *NDG*. These two observations verify the effectiveness of the dynamic strategy for *NDG*.

To sum up, the above comparative experiments demonstrated the effectiveness and efficiency of DGCELSO in solving large-scale optimization problems. In particular, the deep investigation experiments have validated that it is the proposed DGCEL strategy along with the two dynamic strategies that play a crucial role in helping DGCELSO achieve promising performance.

5. Conclusions

This paper proposed a dimension group-based comprehensive elite learning swarm optimizer (DGCELSO) to effectively solve large-scale optimization problems. Specifically, this optimizer first partitions the swarm into two exclusive sets, namely the elite set and the non-elite set. Then, the non-elite particles are updated by learning from the elite ones with the elite particles directly entering the next generation. During the update of each non-elite particle, the dimensions are separated into several dimension groups. Subsequently, for each dimension group, two elites are randomly selected from the elite set and then act as the guiding exemplars to direct the update of the dimension group. In this way, each non-elite particle could comprehensively learn from multiple elites. Moreover, not only are the guiding exemplars for different non-elite particles different, but the guiding exemplars for different dimension groups of the same non-elite particle are also likely to be different. As a result, not only could the learning diversity of particles be improved, but the learning efficiency of particles could also be promoted. To further aid the optimizer to explore and exploit the solution space properly, we designed two dynamic adjustment strategies for the associated control parameters in the proposed DGCELSO.

Experiments conducted on the 1000-D CEC'2010 and CEC'2013 large-scale benchmark sets verified the effectiveness of the proposed DGCELSO by comparing it with nine state-of-the-art large-scale methods. Experimental results demonstrate that DGCELSO achieves highly competitive or even much better performance than the compared methods in terms of both the solution quality and the convergence speed.

Author Contributions: Q.Y.: Conceptualization, supervision, methodology, formal analysis, and writing—original draft preparation. K.-X.Z.: Implementation, formal analysis, and writing—original draft preparation. X.-D.G.: Methodology, and writing—review, and editing. D.-D.X.: Writing—review and editing. Z.-Y.L.: Writing—review and editing, and funding acquisition. S.-W.J.: Writing—review and editing. J.Z.: Conceptualization and writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by the National Natural Science Foundation of China under Grant 62006124 and U20B2061, in part by the Natural Science Foundation of Jiangsu Province under Project BK20200811, in part by the Natural Science Foundation of the Jiangsu Higher Education Institutions of China under Grant 20KJB520006, in part by the National Research Foundation of Korea (NRF-2021H1D3A2A01082705), and in part by the Startup Foundation for Introducing Talent of NUIST.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Jia, Y.H.; Mei, Y.; Zhang, M. A Two-Stage Swarm Optimizer with Local Search for Water Distribution Network Optimization. *IEEE Trans. Cybern.* **2021**. [CrossRef]
2. Cao, K.; Cui, Y.; Liu, Z.; Tan, W.; Weng, J. Edge Intelligent Joint Optimization for Lifetime and Latency in Large-Scale Cyber-Physical Systems. *IEEE Internet Things J.* **2021**. [CrossRef]
3. Chen, W.N.; Tan, D.Z.; Yang, Q.; Gu, T.; Zhang, J. Ant Colony Optimization for the Control of Pollutant Spreading on Social Networks. *IEEE Trans. Cybern.* **2020**, *50*, 4053–4065. [CrossRef]
4. Zuo, T.; Zhang, Y.; Meng, K.; Tong, Z.; Dong, Z.Y.; Fu, Y. A Two-Layer Hybrid Optimization Approach for Large-Scale Offshore Wind Farm Collector System Planning. *IEEE Trans. Ind. Inform.* **2021**, *17*, 7433–7444. [CrossRef]
5. Yang, Q.; Chen, W.N.; Gu, T.; Jin, H.; Mao, W.; Zhang, J. An Adaptive Stochastic Dominant Learning Swarm Optimizer for High-Dimensional Optimization. *IEEE Trans. Cybern.* **2020**, *52*, 1960–1976. [CrossRef]
6. Omidvar, M.N.; Li, X.; Mei, Y.; Yao, X. Cooperative Co-Evolution with Differential Grouping for Large Scale Optimization. *IEEE Trans. Evol. Comput.* **2014**, *18*, 378–393. [CrossRef]
7. Tang, K.; Li, X.; Suganthan, P.; Yang, Z.; Weise, T. *Benchmark Functions for the CEC 2010 Special Session and Competition on Large-Scale Global Optimization*; Nature Inspired Computation and Applications Laboratory, University of Science and Technology of China: Hefei, China, 2009.
8. Li, X.; Tang, K.; Omidvar, M.N.; Yang, Z.; Qin, K.; China, H. *Benchmark Functions for the CEC 2013 Special Session and Competition on Large-Scale Global Optimization*; Technical Report; Evolutionary Computation and Machine Learning Group, RMIT University: Melbourne, Australia, 2013.
9. Yang, Q.; Li, Y.; Gao, X.-D.; Ma, Y.-Y.; Lu, Z.-Y.; Jeon, S.-W.; Zhang, J. An Adaptive Covariance Scaling Estimation of Distribution Algorithm. *Mathematics* **2021**, *9*, 3207. [CrossRef]
10. Yang, Q.; Chen, W.N.; Gu, T.; Zhang, H.; Yuan, H.; Kwong, S.; Zhang, J. A Distributed Swarm Optimizer with Adaptive Communication for Large-Scale Optimization. *IEEE Trans. Cybern.* **2020**, *50*, 3393–3408. [CrossRef]
11. Omidvar, M.N.; Li, X.; Yao, X. A Review of Population-Based Metaheuristics for Large-Scale Black-Box Global Optimization: Part A. *IEEE Trans. Evol. Comput.* 2021. *in press*. Available online: https://ieeexplore.ieee.org/document/9627116 (accessed on 1 January 2022).
12. Omidvar, M.N.; Li, X.; Yao, X. A Review of Population-Based Metaheuristics for Large-Scale Black-Box Global Optimization: Part B. *IEEE Trans. Evol. Comput.* 2021. *in press*. Available online: https://ieeexplore.ieee.org/document/9627138 (accessed on 1 January 2022).
13. Yang, Q.; Xie, H.; Chen, W.; Zhang, J. Multiple Parents Guided Differential Evolution for Large Scale Optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 24–29 July 2016; pp. 3549–3556.
14. Yang, Q.; Chen, W.N.; Li, Y.; Chen, C.L.P.; Xu, X.M.; Zhang, J. Multimodal Estimation of Distribution Algorithms. *IEEE Trans. Cybern.* **2017**, *47*, 636–650. [CrossRef]
15. Eberhart, R.; Kennedy, J. A New Optimizer Using Particle Swarm Theory. In Proceedings of the International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995; pp. 39–43.
16. Shi, Y.; Eberhart, R. A Modified Particle Swarm Optimizer. In Proceedings of the IEEE International Conference on Evolutionary Computation Proceedings: IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
17. Tang, J.; Liu, G.; Pan, Q. A Review on Representative Swarm Intelligence Algorithms for Solving Optimization Problems: Applications and Trends. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1627–1643. [CrossRef]
18. Ren, Z.; Zhang, A.; Wen, C.; Feng, Z. A Scatter Learning Particle Swarm Optimization Algorithm for Multimodal Problems. *IEEE Trans. Cybern.* **2014**, *44*, 1127–1140. [CrossRef]
19. Zhang, J.; Lu, Y.; Che, L.; Zhou, M. Moving-Distance-Minimized PSO for Mobile Robot Swarm. *IEEE Trans. Cybern.* **2021**. [CrossRef]
20. Villalón, C.L.C.; Dorigo, M.; Stützle, T. PSO-X: A Component-Based Framework for the Automatic Design of Particle Swarm Optimization Algorithms. *IEEE Trans. Evol. Comput.* **2021**. [CrossRef]
21. Ding, W.; Lin, C.T.; Cao, Z. Deep Neuro-Cognitive Co-Evolution for Fuzzy Attribute Reduction by Quantum Leaping PSO with Nearest-Neighbor Memeplexes. *IEEE Trans. Cybern.* **2019**, *49*, 2744–2757. [CrossRef]
22. Yang, Q.; Hua, L.; Gao, X.; Xu, D.; Lu, Z.; Jeon, S.-W.; Zhang, J. Stochastic Cognitive Dominance Leading Particle Swarm Optimization for Multimodal Problems. *Mathematics* **2022**, *10*, 761. [CrossRef]
23. Bonavolontà, F.; Noia, L.P.D.; Liccardo, A.; Tessitore, S.; Lauria, D. A PSO-MMA Method for the Parameters Estimation of Interarea Oscillations in Electrical Grids. *IEEE Trans. Instrum. Meas.* **2020**, *69*, 8853–8865. [CrossRef]
24. Lan, R.; Zhu, Y.; Lu, H.; Liu, Z.; Luo, X. A Two-Phase Learning-Based Swarm Optimizer for Large-Scale Optimization. *IEEE Trans. Cybern.* **2020**, *51*, 6284–6293. [CrossRef]
25. Yang, Q.; Chen, W.; Deng, J.D.; Li, Y.; Gu, T.; Zhang, J. A Level-Based Learning Swarm Optimizer for Large-Scale Optimization. *IEEE Trans. Evol. Comput.* **2018**, *22*, 578–594. [CrossRef]
26. Cheng, R.; Jin, Y. A Competitive Swarm Optimizer for Large Scale Optimization. *IEEE Trans. Cybern.* **2015**, *45*, 191–204. [CrossRef]

27. Mahdavi, S.; Shiri, M.E.; Rahnamayan, S. Metaheuristics in Large-Scale Global Continues Optimization: A Survey. *Inf. Sci.* **2015**, *295*, 407–428. [CrossRef]
28. Ma, X.; Li, X.; Zhang, Q.; Tang, K.; Liang, Z.; Xie, W.; Zhu, Z. A Survey on Cooperative Co-Evolutionary Algorithms. *IEEE Trans. Evol. Comput.* **2018**, *23*, 421–441. [CrossRef]
29. Li, X.; Yao, X. Cooperatively Coevolving Particle Swarms for Large Scale Optimization. *IEEE Trans. Evol. Comput.* **2011**, *16*, 210–224.
30. Yang, Q.; Chen, W.; Gu, T.; Zhang, H.; Deng, J.D.; Li, Y.; Zhang, J. Segment-Based Predominant Learning Swarm Optimizer for Large-Scale Optimization. *IEEE Trans. Cybern.* **2017**, *47*, 2896–2910. [CrossRef]
31. Xie, H.Y.; Yang, Q.; Hu, X.M.; Chen, W.N. Cross-Generation Elites Guided Particle Swarm Optimization for Large Scale Optimization. In Proceedings of the IEEE Symposium Series on Computational Intelligence, Athens, Greece, 6–9 December 2016; pp. 1–8.
32. Song, G.W.; Yang, Q.; Gao, X.D.; Ma, Y.Y.; Lu, Z.Y.; Zhang, J. An Adaptive Level-Based Learning Swarm Optimizer for Large-Scale Optimization. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Melbourne, Australia, 17–20 October 2021; pp. 152–159.
33. Potter, M.A.; De Jong, K.A. A Cooperative Co-Evolutionary Approach to Function Optimization. In Proceedings of the International Conference on Parallel Problem Solving from Nature, Berlin, Germany, 22–26 September 1994; pp. 249–257.
34. Yang, Q.; Chen, W.N.; Zhang, J. Evolution Consistency Based Decomposition for Cooperative Coevolution. *IEEE Access* **2018**, *6*, 51084–51097. [CrossRef]
35. Omidvar, M.N.; Yang, M.; Mei, Y.; Li, X.; Yao, X. DG2: A Faster and More Accurate Differential Grouping for Large-Scale Black-Box Optimization. *IEEE Trans. Evol. Comput.* **2017**, *21*, 929–942. [CrossRef]
36. Sun, Y.; Kirley, M.; Halgamuge, S.K. Extended Differential Grouping for Large Scale Global Optimization with Direct and Indirect Variable Interactions. In Proceedings of the Annual Conference on Genetic and Evolutionary Computation, Madrid, Spain, 11–15 July 2015; pp. 313–320.
37. Sun, Y.; Kirley, M.; Halgamuge, S.K. A Recursive Decomposition Method for Large Scale Continuous Optimization. *IEEE Trans. Evol. Comput.* **2017**, *22*, 647–661. [CrossRef]
38. Song, A.; Chen, W.N.; Gong, Y.J.; Luo, X.; Zhang, J. A Divide-and-Conquer Evolutionary Algorithm for Large-Scale Virtual Network Embedding. *IEEE Trans. Evol. Comput.* **2020**, *24*, 566–580. [CrossRef]
39. Deng, H.; Peng, L.; Zhang, H.; Yang, B.; Chen, Z. Ranking-Based Biased Learning Swarm Optimizer for Large-Scale Optimization. *Inf. Sci.* **2019**, *493*, 120–137. [CrossRef]
40. Wang, H.; Liang, M.; Sun, C.; Zhang, G.; Xie, L. Multiple-Strategy Learning Particle Swarm Optimization for Large-Scale Optimization Problems. *Complex Intell. Syst.* **2021**, *7*, 1–16. [CrossRef]
41. Jian, J.R.; Chen, Z.G.; Zhan, Z.H.; Zhang, J. Region Encoding Helps Evolutionary Computation Evolve Faster: A New Solution Encoding Scheme in Particle Swarm for Large-Scale Optimization. *IEEE Trans. Evol. Comput.* **2021**, *25*, 779–793. [CrossRef]
42. Kampourakis, K. *Understanding Evolution*; Cambridge University Press: Cambridge, UK, 2014.
43. Ju, X.; Liu, F. Wind Farm Layout Optimization Using Self-Informed Genetic Algorithm with Information Guided Exploitation. *Appl. Energy* **2019**, *248*, 429–445. [CrossRef]
44. Ju, X.; Liu, F.; Wang, L.; Lee, W.-J. Wind Farm Layout Optimization Based on Support Vector Regression Guided Genetic Algorithm with Consideration of Participation among Landowners. *Energy Convers. Manag.* **2019**, *196*, 1267–1281. [CrossRef]
45. Xia, X.; Gui, L.; Yu, F.; Wu, H.; Wei, B.; Zhang, Y.L.; Zhan, Z.H. Triple Archives Particle Swarm Optimization. *IEEE Trans. Cybern.* **2020**, *50*, 4862–4875. [CrossRef]
46. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
47. Gong, Y.; Li, J.; Zhou, Y.; Li, Y.; Chung, H.S.; Shi, Y.; Zhang, J. Genetic Learning Particle Swarm Optimization. *IEEE Trans. Cybern.* **2016**, *46*, 2277–2290. [CrossRef]
48. Zhan, Z.; Zhang, J.; Li, Y.; Shi, Y. Orthogonal Learning Particle Swarm Optimization. *IEEE Trans. Evol. Comput.* **2011**, *15*, 832–847. [CrossRef]
49. Van den Bergh, F.; Engelbrecht, A.P. A Cooperative Approach to Particle Swarm Optimization. *IEEE Trans. Evol. Comput.* **2004**, *8*, 225–239. [CrossRef]
50. Mei, Y.; Omidvar, M.N.; Li, X.; Yao, X. A Competitive Divide-and-Conquer Algorithm for Unconstrained Large-Scale Black-Box Optimization. *ACM Trans. Math. Softw.* **2016**, *42*, 1–24. [CrossRef]
51. Yang, M.; Zhou, A.; Li, C.; Yao, X. An Efficient Recursive Differential Grouping for Large-Scale Continuous Problems. *IEEE Trans. Evol. Comput.* **2021**, *25*, 159–171. [CrossRef]
52. Sun, Y.; Omidvar, M.N.; Kirley, M.; Li, X. Adaptive Threshold Parameter Estimation with Recursive Differential Grouping for Problem Decomposition. In Proceedings of the Genetic and Evolutionary Computation Conference, Kyoto, Japan, 15–19 July 2018.
53. Ma, X.; Huang, Z.; Li, X.; Wang, L.; Qi, Y.; Zhu, Z. Merged Differential Grouping for Large-scale Global Optimization. *IEEE Trans. Evol. Comput.* **2022**, *in press*. [CrossRef]
54. Liu, H.; Wang, Y.; Fan, N. A Hybrid Deep Grouping Algorithm for Large Scale Global Optimization. *IEEE Trans. Evol. Comput.* **2020**, *24*, 1112–1124. [CrossRef]

55. Zhang, X.Y.; Gong, Y.J.; Lin, Y.; Zhang, J.; Kwong, S.; Zhang, J. Dynamic Cooperative Coevolution for Large Scale Optimization. *IEEE Trans. Evol. Comput.* **2019**, *23*, 935–948. [CrossRef]
56. Neshat, M.; Mirjalili, S.; Sergiienko, N.Y.; Esmaeilzadeh, S.; Amini, E.; Heydari, A.; Garcia, D.A. Layout Optimisation of Offshore Wave Energy Converters Using a Novel Multi-swarm Cooperative Algorithm with Backtracking Strategy: A Case Study from Coasts of Australia. *Energy* **2022**, *239*, 122463. [CrossRef]
57. Pan, Q.K.; Gao, L.; Wang, L. An Effective Cooperative Co-Evolutionary Algorithm for Distributed Flowshop Group Scheduling Problems. *IEEE Trans. Cybern.* **2020**. [CrossRef]
58. Neshat, M.; Alexander, B.; Wagner, M. A Hybrid Cooperative Co-Evolution Algorithm Framework for Optimising Power Take off and Placements of Wave Energy Converters. *Inf. Sci.* **2020**, *534*, 218–244. [CrossRef]
59. Liang, M.; Wang, W.; Dong, C.; Zhao, D. A Cooperative Coevolutionary Optimization Design of Urban Transit Network and Operating Frequencies. *Expert Syst. Appl.* **2020**, *160*, 113736. [CrossRef]
60. Zhao, S.-Z.; Liang, J.J.; Suganthan, P.N.; Tasgetiren, M.F. Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search for Large Scale Global Optimization. In Proceedings of the IEEE Congress on Evolutionary Computation, Hong Kong, China, 1–6 June 2008; pp. 3845–3852.
61. Cheng, R. A Social Learning Particle Swarm Optimization Algorithm for Scalable Optimization. *Inf. Sci.* **2015**, *291*, 43–60. [CrossRef]
62. Mohapatra, P.; Das, K.N.; Roy, S. A Modified Competitive Swarm Optimizer for Large Scale Optimization Problems. *Appl. Soft Comput.* **2017**, *59*, 340–362. [CrossRef]
63. Li, D.; Guo, W.; Lerch, A.; Li, Y.; Wang, L.; Wu, Q. An Adaptive Particle Swarm Optimizer with Decoupled Exploration and Exploitation for Large Scale Optimization. *Swarm Evol. Comput.* **2021**, *60*, 100789. [CrossRef]
64. Lan, R.; Zhang, L.; Tang, Z.; Liu, Z.; Luo, X. A Hierarchical Sorting Swarm Optimizer for Large-Scale Optimization. *IEEE Access* **2019**, *7*, 40625–40635. [CrossRef]
65. Kong, F.; Jiang, J.; Huang, Y. An Adaptive Multi-Swarm Competition Particle Swarm Optimizer for Large-Scale Optimization. *Mathematics* **2019**, *7*, 521. [CrossRef]
66. Huang, H.; Lv, L.; Ye, S.; Hao, Z. Particle Swarm Optimization with Convergence Speed Controller for Large-Scale Numerical Optimization. *Soft Comput.* **2019**, *23*, 4421–4437. [CrossRef]
67. LaTorre, A.; Muelas, S.; Peña, J.-M. A Comprehensive Comparison of Large Scale Global Optimizers. *Inf. Sci.* **2015**, *316*, 517–549. [CrossRef]