Article

A Hybrid of Fully Informed Particle Swarm and Self-Adaptive Differential Evolution for Global Optimization

Shir Li Wang [1,*], Sarah Hazwani Adnan [2], Haidi Ibrahim [3], Theam Foo Ng [2,*] and Parvathy Rajendran [4,5]

[1] Faculty of Art, Computing and Creative Industry, Universiti Pendidikan Sultan Idris (UPSI), Tanjong Malim 35900, Perak, Malaysia
[2] Centre of Global Sustainability Studies (CGSS), Level 5, Hamzah Sendut Library, Universiti Sains Malaysia, Minden 11800, Pulau Pinang, Malaysia
[3] School of Electrical and Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, Nibong Tebal 14300, Pulau Pinang, Malaysia
[4] School of Aerospace Engineering, Universiti Sains Malaysia, Engineering Campus, Nibong Tebal 14300, Pulau Pinang, Malaysia
[5] Faculty of Engineering and Computing, First City University College, Bandar Utama, Petaling Jaya 47800 Selangor, Malaysia
* Correspondence: shirli_wang@fskik.upsi.edu.my (S.L.W.); tfng@usm.my (T.F.N.)

Abstract: Evolutionary computation algorithms (EC) and swarm intelligence have been widely used to solve global optimization problems. The optimal solution for an optimization problem is called by different terms in EC and swarm intelligence. It is called individual in EC and particle in swarm intelligence. Self-adaptive differential evolution (SaDE) is one of the promising variants of EC for solving global optimization problems. Adapting self-manipulating parameter values into SaDE can overcome the burden of choosing suitable parameter values to create the next best generation's individuals to achieve optimal convergence. In this paper, a fully informed particle swarm (FIPS) is hybridized with SaDE to enhance SaDE's exploitation capability while maintaining its exploration power so that it is not trapped in stagnation. The proposed hybrid is called FIPSaDE. FIPS, a variant of particle swarm optimization (PSO), aims to help solutions jump out of stagnation by gathering knowledge about its neighborhood's solutions. Each solution in the FIPS swarm is influenced by a group of solutions in its neighborhood, rather than by the best position it has visited. Indirectly, FIPS increases the diversity of the swarm. The proposed algorithm is tested on benchmark test functions from "CEC 2005 Special Session on Real-Parameter Optimization" with various properties. Experimental results show that the FIPSaDE is more effective and reasonably competent than its standalone variants, FIPS and SaDE, in solving the test functions, considering the solutions' quality.

Keywords: differential evolution; fully informed particle swarm; self-adaptive differential evolution; particle swarm optimization

check for updates

Citation: Wang, S.L.; Adnan, S.H.; Ibrahim, H.; Ng, T.F.; Rajendran, P. A Hybrid of Fully Informed Particle Swarm and Self-Adaptive Differential Evolution for Global Optimization. *Appl. Sci.* **2022**, *12*, 11367. https://doi.org/10.3390/app122211367

Academic Editor: Vincent A. Cicirello

Received: 12 September 2022
Accepted: 5 November 2022
Published: 9 November 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The real-life optimization problems are often unpredictable and dynamic, which means that the optimization problems are facing many time-varying and unimaginable constraints, such as routing problems, scheduling problems, prediction and more variants of broad application problems. The primary purpose of the optimization problems is to find the most optimum minimum or maximum value in the search space where often the problems have more than one maximum or minimum point. This type of problem is considered an NP-hard (nondeterministic polynomial time) problem, requiring high computational processing to solve it. Currently, evolutionary computation (EC) is widely used in numerous studies to solve this type of problem. The EC technique has proven to provide an effective search in a complex search to achieve optimal or near-optimal solutions and has been proven to have strong global search ability [1].

As a classic heuristic method, Particle Swarm Optimization (PSO) is one of the most used and reliable swarm intelligence techniques [2]. It has been successfully adopted into many practical applications due to its efficiency, fast optimization speed and simplicity of implementation. In the traditional PSO, each particle of the population is learning from its nearest neighbors, and this may cause the particle swarm to stagnate in the local region due to rapid convergence [1]. Unlike the traditional PSO, one of its variants, the Fully Informed Particle Swarm Optimization, FIPS, introduced by Mendes, Kennedy and Neves in 2004, has the weight contributions of all particles to have the same value. The particles are influenced by their whole neighborhood in a specific way according to different neighborhood topologies. This technique enables each particle to access the most successful solutions from the whole swarm, not necessarily the best from its nearest neighbor. Accordingly, the performance of FIPS algorithm types is also generally more dependent on the neighborhood topology [3].

Unlike PSO and its variants, Differential Evolution (DE) is also a widely used algorithm with a remarkable ability to find the optimal solution. DE relies upon its strength in handling starting initial points where multiple starting points are randomly chosen during sampling of potential solutions [4,5]. Several versions of the proposed adaptive DE focus on manipulating the parameter of DE with the introduction of a new way of controlling the value of existing parameters. The adaptive differential evolution (jDE) [6] implemented several DE strategies to control the diversity of the population. Additionally, it can self-adapt the scaling factor F and the crossover rate C_r. The parameter adaptive differential evolution (JADE) [7] relies on greedy mutation strategy (DE/current-to-*p*best) with optimal external achieve and utilizing the previously explored inferior solutions. In SaDE [8], suitable learning strategies and parameter settings are gradually self-adapted according to the previous learning experiences. Even more, the choice of learning strategy and the two essential control parameters of DE, F and C_r, are not required to be specified before the evolution phase.

In the past few years, the development of adaptive mechanisms has emerged to be one of the critical issues in the branch of the EC algorithm. Adaptive mechanism refers to the ability of the algorithm to change its behavior according to information available during its running phase. The number of works in which adaptive mechanisms are being successfully used has increased enormously over the past few decades. The applications include a wide variety of areas, such as routing problems, signal processing, optimization problem and medical fields. Furthermore, the efficiency of the adaptive mechanism mainly depends on the algorithm design and the algorithm used for adaptation.

One of the common drawbacks EC algorithms face in solving optimization problems is the lack of diversity in solutions causing a suboptimal solution. A potential approach to overcoming this drawback is a hybrid of EC and swarm intelligence. FIPS and SaDE emerge as promising EC and swarm intelligence algorithms in solving optimization problems. Therefore, they are chosen to form a hybrid in our study. FIPS and SaDE are hybridized owing to the solutions' diversity in FIPS and the optimal solution quality in SaDE. FIPS improves the solutions' diversity by gathering knowledge about its neighborhood's solutions. Each solution in FIPS receives information from all its neighbors rather than just the best one. SaDE is a simple, powerful and self-adaptive type of DE which finds an optimal solution through exploration and exploitation. SaDE minimally relies on user-specified parameters because it can self-adapt its parameters to solve optimization problems. FIPS improves the solutions through information gathering from its neighbors and maneuvering the group of solutions to move toward the optimal region, which is explored and exploited by SaDE. Therefore, in the hybrid of FIPS and SaDE, they complement each other by balancing the exploration of neighbors and exploitation of the optimal solutions.

2. Related Works

2.1. Improved Differential Evolution

DE is one of the most popular evolutionary algorithms used to solve optimization problems because it is simple, robust and computationally efficient. However, DE faces issues in convergence rate and local exploitation rate [9]. Therefore, various research efforts for its improvement are continuously carried out even though the algorithm has been introduced 25 years ago by Storn and Price [10]. Besides, surveys related DE variants are updated after a certain years to accommodate various modifications of DEs, as shown in [5,9,11–15]. The surveys show that most of the research efforts focus on improving the performance of DE through parameter settings and modifications in genetic operations consisting of initialization, differential mutation, crossover and selection. Another potential direction towards improving DE is the use of hybridization, and it has gained research attention [9,11,12].

Most of the optimization algorithms are not able to solve a variety of problems [9]. However, continuous research efforts are required to improve the algorithm through the complementary of their advantages. Hybridization is one the main research directions in improving the performance of DE because different optimization algorithms have different search behaviours and advantages [16].

It is implemented to enhance its performance and overcome its limitations such as convergence speed [11], premature convergence [9] and local minima [9]. The types of algorithms used for hybridization in DE can either belong to the same or different categories of algorithms. The work in [12] categorizes the algorithm in hybridization of DE into statistical techniques and algorithms. On the other hand, the work in [11] shows that most of the algorithms in a hybridization of DE are from the swarm intelligence algorithms. The commonly used swarm intelligence algorithms to hybridize with DE include PSO, ant colony algorithm (ACA) and artificial bee colony (ABC).

DE variants produce robust solutions owing to their exploration ability. However, they have issues related to premature convergence and local exploitation. On the other hand, one of the drawbacks of PSO is premature convergence [17]. Premature convergence is caused by improper velocity adjustment when PSO is configured by inappropriate acceleration coefficient and inertia weights [18]. Consequently, the particles move in undesired directions, causing stagnation around or being trapped in suboptimal [18]. A review of the modifications, extensions and hybridization of the PSO algorithm and their applications is presented in [19].

Most of the hybrid variants of DE is formed with PSO [11] and an extensive survey about the hybrid can be found in [9]. A hybrid of advanced DE (ADE) and (PSO) (APSO), namely, AHDEPSO, was proposed to solve unconstrained optimization problems in [9]. Given that a hybrid of DE and PSO offers complementary properties and has the effect of balancing the exploration and exploitation phases, the hybrid of their advanced variants has attracted more research attention. The research focusing on how to combine PSO and DE is still an open problem [16].

The premature convergence of PSO in the optimization process can be improved based on four different methods: parameter settings, neighborhood topology, learning strategy and hybridization [20]. Neighbourhood topology enhances PSO's exploration capability and different topologies affected solve different optimization problems [20]. On the other hand, hybridization is used to complement the weakness of intelligent algorithms by combining their helpful features. Therefore, FIPS, instead of the original PSO, is hybridized with DE to improve the performance of the optimization algorithms.

The self-adaptive mutation differential evolution algorithm based on particle swarm optimization (DEPSO) [21] uses balance to improve the optimization. The hybrid uses the selection probability of mutation strategy to decide between the modified DE/rand/1 and the PSO's mutation strategy. The modified mutation strategy with the elite archive strategy, called DE/e-rand/1, is proposed in DEPSO. The framework of DEPSO is still based on DE. Unlike DEPSO, our proposed FIPSaDE is based on the framework of PSO. The hybridization of DEPSO focuses on selecting mutation strategies from DE and PSO to generate the mutant vector. Therefore, its hybridization involves the mutation phase only. On the other hand, the hybridization in FIPSaDE involves integrating DE's algorithmic operations, from the mutation to selection phases, into the framework of PSO before the velocity and position updating.

Dash et al. proposed HDEPSO [22] to solve various benchmark functions and an optimization problem focusing on the effectiveness of the sharp edge FIR filter (SEFIRF). The proposed framework is similar to FIPSaDE, whereby, DE's mutation, crossover and selection are integrated with the best particles of PSO to enhance global searching ability. However, the settings of F and C_r were not adaptive. F is fixed as a constant and C_r refers to an arbitrary number between $[0, 1]$. In contrast, the settings of F and C_r in FIPSaDE are adaptive.

For the case of the hybrid approach, it may cause more uncertainties in parameter setting in view of the fact that a user needs to realize how to set parameters for at least two algorithms [23]. The task of parameter setting in solving optimization problem is problem-dependent, user-dependent and algorithm-dependent. The task becomes more complex when the algorithms of the hybrid are from different branches of machine learning. Therefore, researchers start to focus on the possibility of applying adaptation methods for users that have minimum knowledge about the algorithms, parameters settings and the problems. DE variants with varying adaptation levels have shown promising results in solving optimization problems and its popularity in setting DE's parameters can be found in the aforementioned reviews.

For example, the work in [24] showed the use of adaptation to tune the configurations of F, C_r, and mutation strategy at different stages of the evolution. Another work demonstrating the use of adaptation in controlling parameters F, C_r is the Success-History-based adaptive DE (SHADE) algorithm [25]. SHADE uses the nearest spatial neighborhood-based modification to the adaptation process of the parameters described above. Do et al. [26] also proposed an adaptive mechanism to determine the parameters F and F, C_r with the mutation and selection processes determined by the best individual-based mutation and elitist selection techniques. The adaptation in population sizing is summarized in the review by Piotrowski [27]. These research studies have shown the use of adaptation to control DE's various parameters.

The adaptability trend to set DE's parameters, from partial adaptiveness to full adaptiveness, is increasing. Since the adaptive DE variants have shown promising results, its use to form the hybrid can reduce the complexity of parameter setting because the number of algorithms that require user-specified parameters has declined. Therefore, we use a adaptive DE called SaDE in [8] to form the hybrid of DE and PSO in the current work. SaDE adapts the control parameter F and C_r based on the mutation strategy $DE/rand/1/binstrategy$.

2.2. Particle Swarm Optimization

PSO is another branch of EC that operates based on swarm-based intelligence. Its implementation is simple and easy, but it also suffers from premature convergence [18,28]. Therefore, various modifications are applied to the standard PSO to overcome its drawback. In [18], the researchers categorize the modification of PSO into four strategies: Modification of the PSO's controlling parameters (1); (2) Hybridization with other meta-heuristic algorithms such as GA and DE; (3) Cooperation; (4) Multi-swarm techniques. The work in [9,18] reflects that a hybrid of PSO and DE has gained popularity in recent years as the strategy to improve EC's performance in solving optimization problems. The performance of PSO is affected by its neighbourhood topology, where the particles within the neighbour topology communicate with each other and share information. FIPS is the PSO that relies on the particle's best positions of its neighbors in updating its velocity to prevent it from being stuck at local optimum. The hybrid variants of DE and PSO have shown better quality of solutions and computation efficiency than the original PSO [18]. Therefore, a hybrid of the adaptive SaDE and FIPS is formed to complement their weakness in solving optimization problems in our current work. Brief descriptions of SaDE and FIPS are provided in Sections 2.3 and 2.4.

2.3. Self-Adaptive Differential Evolution (SaDE)

SaDE is a variant of DE that can produce better results than the traditional DE algorithms [2]. This algorithm has been applied in various optimization problems [2–4,6,7]. In SaDE, two out of three critical control parameters of DE (i.e., F and C_r) are manipulated for DE improvement. Since these parameters are used as the external fixed parameters, these control parameters are not deemed to be evolving entities in the earliest EA algorithm. Later on, it was determined that certain parameters could be changed during the evolution process in order to reach the desired level of convergence [8].

The population size N_p is not favored as a chosen parameter because its value is not sensitive to the efficiency and robustness of the DE algorithm [8]. N_p is held as a user-specified parameter, whereas the F value is set to be between (0, 2] for different individuals, with a normal distribution of mean 0.5 and standard deviation of 0.3 [29]. Instead of the commonly used (0, 1] for F values, the range between (0, 2] was chosen to maintain both small F values for local search and large F values for global search. C_r would initially be naturally distributed in a range with a mean of C_{r_m} and a standard deviation, *std*, of 0.1. C_{r_m} value is set to 0.5 as a starting point where the values will be held for several generations before being replaced by a new value with a similar normal distribution for the next generation.

Throughout each generation, better C_r values associated with trial vectors that are able to reach the next generation will be registered. The mean value for the normal distribution of C_r is recalculated based on all observed values corresponding to the effective trial vectors during the cycle. In order to prevent potentially inappropriate long-term accumulated results, successful C_r values captured during the recalculation of the normal distribution mean are not saved.

2.4. Fully Informed Particle Swarm (FIPS)

The FIPS algorithm that was developed by Mendes et al. in 2004 [1] is a variant of PSO in which each particle is influenced by all of its K-neighbors. This idea is in contrast to the standard PSO algorithm, in which the particle is drawn to the best location it has visited as well as the best position found by the particle in its neighborhood that manages to produce the best result. The algorithm is motivated by how individuals in human society are influenced by a statistical summary of their world [30].

Each particle in the FIPS swarm is affected by a group of particles from its surrounding neighborhood, which is not necessarily to the best particle's location [1]. The velocity equation for FIPS is modified based on canonical PSO. Each particle inside the swarm is affected by the achievements of all its neighbors, rather than pointing to just one best particle performance from its neighbors [30]. FIPS's velocity and position updates are defined by Equations (1) and (2), respectively.

$$v_i(t+1) = wv_i + \sum_{m=1}^{|N_i|} \gamma_m(t) \frac{y_m(t) - x_i(t)}{|N_i|} \tag{1}$$

$$x_i(t+1) \;=\; x_i(t) + v_i(t+1) \tag{2}$$

where N_i refers to the set of particles in particle i's neighborhood. $\gamma_m(t)$ is in the range of $U(0, c_1 + c_2)^D$ and D is dimension of the problem. $y_m(t)$ refer to the best position previously visited by m. Coefficients c_1, c_2 and w represent cognitive, social and inertia weight, respectively. There are five variants of the algorithm, as stated by Mendes et al. [1], which are FIPS, wFIPS, wdFIPS, Self and wSelf:

- FIPS refers to a fully informed particle swarm optimization algorithm with w returning a constant value and the number of neighbourhood contributions being the same;
- wFIPS refers to a type of FIPS algorithm in which each neighbor's input is weighted by the goodness of its previous best particle;
- wdFIPS refers to a type of FIPS algorithm where the contribution of each neighbor is weighted by its distance in the search space from the target particle;
- Self refers to a FIPS algorithm in which the previous best particle received half the weight;
- wSelf refers to a FIPS algorithm where the previous best particle received half the weight and the contribution of each neighbor was weighted by the goodness of its previous best particle.

The swarm optimization algorithm is a type of algorithm that is greatly influenced by the neighborhood. As a result, the efficiency of the FIPS algorithm is generally even more dependent on its neighborhood topology. Ten topology types selected by Cleghorn and Engelbrecht [30] to be implemented on FIPS include All, Ring, Four Clusters, Pyramid, Square, UAll, UFour Clusters, UPyramid and USquare. Topology with the "U" prefix refers to a similar topology without the prefix, but with the particle's own index omitted from the neighborhood.

3. Methodology

FIPS and SaDE are hybridized owing to the solutions' diversity in FIPS and the quality of optimal solution in SaDE. One of the common drawbacks of using optimization algorithms is the lack of diversity leading to a suboptimal solution [31]. The use of FIPS could improve the drawback. In the FIPS, the neighbors around a solution become the source of influence to improve the solution's diversity. DE is remarkable in finding the optimal solution in a group of solutions. However, choosing suitable control parameter values is tricky as the values need to vary according to each problem. SaDE has the advantage that the user does not need to determine the optimal parameters settings, and time complexity does not increase as the rules for applying SaDE are simple [29]. SaDE performs well in finding the optimal solution in a group of solutions through self-adapting its parameters. Each particle in FIPS receives information from all of its neighbors rather than just the best one. The particle may sometimes trap in local optima, and the FIPS topology network may help to monitor and maneuver the particle's position. Therefore, the operation in FIPS improves the solutions through the exploration of its neighbors and guides the group of solutions moving toward the optimal region.

A hybridization between SaDE and FIPS, called FIPSaDE, could improve the performance in solving complex optimization problems. The main process of FIPSaDE is structured according to the usual structure of the PSO algorithm. The solution found in FIPSaDE is called *individual* or *solution* interchangeably. The FIPS parameter is updated each time before the mutation process of the SaDE algorithm and again during the selection process in controlling the quality of individuals chosen for the next generation. The process is repeated iteratively until the algorithm reaches the optimum value. The presence of the FIPS process creates a disturbance in the population in which the FIPS is focused on moving the solution to explore its surrounding. This helps in maintaining the diversity of the population created and producing an excellent optimal solution. The pseudocode of FIPSaDE is given in Algorithm 1.

In FIPSaDE, a swarm of individuals flies in a D-dimensional search space seek an optimal solution. Each individual i possesses a current velocity vector $v_i = [v_{i1}, v_{i2}, \ldots, v_{iD}]$ and a current position vector $x_i = [x_{i1}, x_{i2}, \ldots, x_{iD}]$, where D is the number of dimensions. The FIPSaDE process starts by randomly initializing v_i and x_i.

Algorithm 1: The Pseudocode of the FIPSaDE for Solving a Minimization Problem

```
*****************************Start of FIPS******************************
Initialization
Define the swarm size $N_p$
for each individual $i \, \epsilon [1 \ldots N_p]$ do
  Randomly generate $x_i$ and $v_i$.
  Evaluate the fitness of $x_i$ denoting it as $f(x_i)$.
  Set $Pbest_i = x_i$ and $f(Pbest_i) = f(x_i)$.
end for
Set $Gbest = Pbest_1$ and $f(Gbest) = f(Pbest_1)$.
for each individual $i \, \epsilon [1 \ldots N_p]$ do
  if $f(Pbest_i) < f(Gbest)$ then
    $f(Gbest) = f(Pbest_i)$
  end if
end for
while $t <$ maximum iterations do
  for each individual $i \, \epsilon [1 \ldots N_p]$ do
    -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.Start of SaDE-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
    Generate vector $[x_i, F_i, C_{r_i}]$.
    Update the parameters $F_i$ and $C_{r_i}$ based on Equations (3) and (4).
    Mutate $x_i$ based on Equation (5).
    Crossover $x_i$ based on Equation (6).
    Select $x_i$ based on Equation (7).
    -.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.End of SaDE-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-.-
    Evaluate its velocity $v_i(t+1)$ based on Equation (1).
    Update the position $x_i(t+1)$ of the particle based on Equation (2).
    if $f(x_i(t+1)) < f(Pbest_i)$ then
      $Pbest_i = x_i(t+1)$
      $f(Pbest_i) = f(x_i(t+1))$
    end if
    if $f(Pbest_i) < f(Gbest)$ then
      $Gbest = Pbest_i$
      $f(Gbest) = f(Pbest_i)$
    end if
  end for
  $t = t + 1$
end while
return $Gbest$
*****************************End of FIPS******************************
```

In each iteration t, FIPSaDE uses a self-adapting mechanism of two control parameters, i.e., F and C_r. Each individual i is extended with control parameter values F and C_r as a vector $[x_i, F_i, C_{r_i}]$. New control parameters $F_{i,(t+1)}$ and $C_{r_i,(t+1)}$ are calculated before the mutation operator based on Equations (3) and (4). Therefore, the parameters influence the mutation, crossover and selection operations of the new vector $x_{i,(t+1)}$.

$$F_{i,\,(t+1)} = \begin{cases} F_l + rand_1 * F_u & \text{if } rand_2 < \tau_1 \\ F_{i,\,t} & \text{otherwise} \end{cases} \tag{3}$$

$$C_{r_i,\,(t+1)} = \begin{cases} rand_3 & \text{if } rand_4 < \tau_2 \\ C_{r_i,\,t} & \text{otherwise} \end{cases} \tag{4}$$

where $rand_j$, for $j \epsilon \{1,2,3,4\}$ are uniform random values within the range [0, 1]. The parameters τ_1, τ_2, F_l, F_u are fixed to values 0.1, 0.1, 0.1, 0.9, respectively in [8].

In the mutation process, for each target vector x_i, a mutant vector q_i is generated according to Equation (5)

$$q_{i,(t+1)} = x_{r_1,t} + F_{i,t} * (x_{r_2,t} - x_{r_3,t}) \tag{5}$$

with randomly chosen indexes $r_1, r_2, r_3 \epsilon [1, N_p]$.

Then, a crossover operator forms a trial vector $p_i = (p_{i1}, p_{i2}, \ldots p_{iD})$, with the target vector is mixed with the mutated vector using Equation (6).

$$p_{ij,(t+1)} = \begin{cases} q_{ij,(t+1)} & \text{if } (rand_j(0,1) \leq C_{r_i,t}) \text{ or } (j = j_{rand}) \\ x_{ij,t} & \text{otherwise} \end{cases} \tag{6}$$

for $i = 1,2,\ldots N_p$ and $j = 1,2,\ldots D$. Index $j_{rand} \epsilon \{1, N_p\}$ is a randomly chosen integer that is responsible for the trial vector containing at least one component from the mutant vector.

In the selection process, a greedy selection scheme for a minimization is used and it is shown in Equation (7).

$$x_{i,(t+1)} = \begin{cases} p_{i,(t+1)} & \text{if } f(p_{i,(t+1)}) \leq f(x_{i,t}) \\ x_{i,t} & \text{otherwise} \end{cases} \tag{7}$$

In each iteration t, the best position that has been found by individual i, $Pbest_i = [Pbest_{i1}, Pbest_{i2}, \ldots, Pbest_{iD}]$ and the best position that has been found by the whole swarm $Gbest = [Gbest_1, Gbest_2, \ldots, Gbest_D]$ guide individual i to update its velocity and position by Equations (1) and (2).

The FIPSaDE is effective in traversing through swarm spaces while avoiding getting stuck inside local search using the information collected from individuals' neighborhood. The individuals of each population will contain a control parameter adapted from SaDE and also a traversing parameter from FIPS in order to exploit the swarm space that is searching for the best global point, while avoiding getting trapped inside a local point. If some population suffers from slow and premature convergence or if the population becomes stagnant, the velocity parameter and position parameter from each individual will be able to help the population to evolve out into a better point. Experimental results validated the effectiveness and strength of our proposed model.

4. Experimental Setup

The performance of FIPSaDE is evaluated against the 25 benchmark functions from "CEC 2005 Special Session on Real-Parameter Optimization" (CEC 2005) [32], namely, F1–F25. The benchmark functions consist of 5 unimodal functions, 7 basic multimodal functions, 2 expanded multimodal functions and 11 composition functions. The details of the functions are described in Table 1. The number of variables for each function is represented by D and the ranges of variable search are represented by S. All functions are a minimization of problems with their best minimum solutions can refer to [32]. All experiments were conducted in the Eclipse software by using the Java programming language and on a laptop featuring an Intel(R) Core (TM) i5-7200U CPU @ 2.50GHz processor with 4GB of RAM.

Table 1. Details on the benchmark functions.

Denotation	Test Function	S	Modality
F1	Shifted Sphere Function	$[-100, 100]^D$	Unimodal
F2	Shifted Schwefel's Problem 1.2	$[-100, 100]^D$	Unimodal
F3	Shifted Rotated High Conditioned Elliptic Function	$[-100, 100]^D$	Unimodal
F4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	$[-100, 100]^D$	Unimodal
F5	Schwefel's Problem 2.6 with Global Optimum on Bounds	$[-100, 100]^D$	Unimodal
F6	Shifted Rosenbrock's Function	$[-100, 100]^D$	Basic Multimodal
F7	Shifted Rotated Griewank's Function without Bounds	$[0, 600]^D$	Basic Multimodal
F8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	$[-32, 32]^D$	Basic Multimodal
F9	Shifted Rastrigin's Function	$[-5, 5]^D$	Basic Multimodal
F10	Shifted Rotated Rastrigin's Function	$[-5, 5]^D$	Basic Multimodal
F11	Shifted Rotated Weierstrass Function	$[-0.5, 0.5]^D$	Basic Multimodal
F12	Schwefel's Problem 2.13	$[-\Pi, \Pi]^D$	Basic Multimodal
F13	Shifted Expanded Griewank's plus Rosenbrock's Function	$[-5, 5]^D$	Expanded Multimodal
F14	Shifted Rotated Expanded Scaffer's $f6$ Function	$[-100, 100]^D$	Expanded Multimodal
F15	Hybrid Composition Function	$[-5, 5]^D$	Hybrid Composition
F16	Rotated Version of Hybrid Composition Function $f15$	$[-5, 5]^D$	Hybrid Composition
F17	Rotated Version of Hybrid Composition Function $f15$ with Noise in Fitness	$[-5, 5]^D$	Hybrid Composition
F18	Rotated Hybrid Composition Function	$[-5, 5]^D$	Hybrid Composition
F19	Rotated Hybrid Composition Function with Narrow Basin Global Optimum	$[-5, 5]^D$	Hybrid Composition
F20	Rotated Hybrid Composition Function with Global Optimum on the Bounds	$[-5, 5]^D$	Hybrid Composition
F21	Rotated Hybrid Composition Function	$[-5, 5]^D$	Hybrid Composition
F22	Rotated Hybrid Composition Function with High Condition Number Matrix	$[-5, 5]^D$	Hybrid Composition
F23	Non-continuous Rotated Hybrid Composition Function	$[-5, 5]^D$	Hybrid Composition
F24	Rotated Hybrid Composition Function	$[-5, 5]^D$	Hybrid Composition
F25	Rotated Hybrid Composition Function without Bounds	Initialize population in $[2, 5]^D$ but no exact search range set	Hybrid Composition

In this experiment, the performance of FIPSaDE is studied by comparing the algorithm with four known algorithms, which are FIPS by Mendes et al. [1], DE by Storn and Price [10], SaDE by Brest et al. [8] and DE-PSO by Pant et al. [31]. FIPS, DE and SaDE were selected as these algorithms can be considered as the parent or the ancestor of FIPSaDE. As for DE-PSO, it was selected as the algorithm is a hybrid algorithm that features a similar structure and concept as FIPSaDE. The parameters for the algorithms are shown in Tables 2 and 3. The optimization test for all algorithms is restricted to a maximum number of function evaluation (MAX_FEs), which are 5×10^4 for both $10D$ and $50D$ problem, and 3×10^4 for $30D$ problem. A success-threshold of 10^{-14} was administered for the experiments. This means the evolutionary processes are terminated if the best-fitness, $f_{best} < f_{target}$ is reached, with $f_{target} = f(x*) + 10^{-14}$. Otherwise, the processes continue until they reach MAX_FEs. The evolutions for all models of DEs are run until they meet the stopping criterion and repeated by using the k-th seed numbers, $k = 1, 2, \ldots, K$, with K referring to the maximum number of runs.

Table 2. Parameter settings of DE, PSO, DE-PSO, SaDE and FIPS.

Parameter	DE	PSO	DE-PSO	SaDE	FIPS
Population size, N_p ($D = 10, D = 30, D = 50$)	$(25, 30, 50)$	$(25, 30, 50)$	$(25, 30, 50)$	$(25, 30, 50)$	$(25, 30, 50)$
MAX_FEs ($D = 10, D = 30, D = 50$)	$(5 \times 10^4, 3 \times 10^4, 5 \times 10^4)$	$(5 \times 10^4, 3 \times 10^4, 5 \times 10^4)$	$(5 \times 10^4, 3 \times 10^4, 5 \times 10^4)$	$(5 \times 10^4, 3 \times 10^4, 5 \times 10^4)$	$(5 \times 10^4, 3 \times 10^4, 5 \times 10^4)$
Crossover rate, Cr	0.1	-	0.95	$N\ (0.5 \pm 0.1)$	-
Scale factor, F	0.5	-	0.9	$F = (0, 2]$ with $N(0.5 \pm 0.3)$	-
Mutation strategy	de/rand/1/bin	-	de/rand/1/bin	de/rand/1/bin,	-
Acceleration rate (c_1, c_2)	-	$(2.05, 2.05)$	$(2.0, 2.0)$	-	$(2.05, 2.05)$
Inertial weight, w	-	0.7298	0.9	-	0.7298

Table 3. Parameter settings of FIPSaDE.

Parameter	Dimension		
	10	**30**	**50**
Population size, N_p	25	30	50
MAX_FEs	5×10^4	3×10^4	5×10^4
K runs per case	30	30	20
Crossover rate, Cr	$N(0.5 \pm 0.1)$		
Scale factor, F	$F = (0, 2]$ with $N\ (0.5 \pm 0.3)$		
Learning period	50		
Acceleration rate, c_1	2.05		
Acceleration rate, c_2	2.05		
Inertial weight, w	0.7298		
Neighborhood Type	Self		
Neighborhood Topology	Four cluster		
Symmetric	Asymmetric		

5. Results and Analysis

The difference between current fitness value and optimum value, also known as error value, is used to compare the algorithms' performance. The average errors of the independent runs of all algorithms for 10D, 30D and 50D problems are summarized in Tables 4–6. The last rows in Tables 4–6 show the frequency of having the best result for the algorithms, h_{win}. Based on the results in Tables 4–6, DE cannot find the solutions for functions F15–F17 because its solutions have undefined numeric results for the functions that are divided by zero.

When the settings are 10D and N is 25 (10D25N), SaDE, DE-PSO and FIPSaDE have the highest h_{win}, which is 6. This means the three algorithms have compatible performances in solving the problems. An interesting finding from the comparison of h_{win} for SaDE, DE-PSO and FIPSaDE for the setting of 10D25N is that they have the lowest error for different functions. This finding may indicate that they perform well for different characteristics of problems. As the combinations of D and N increase to 30D30N and 50D50N, FIPSaDE shows distinctive performances compared to the other algorithms by having the highest h_{win} in both settings. FIPSaDE has $h_{win} = 9$ and $h_{win} = 12$ in 30D30N and 50D50N, respectively. In other words, FIPSaDE's performances improves with the increase of problem dimensionality and population size. In contrast, the performances of FIPS are the worst among the algorithms for the three settings. FIPS, DE and SaDE show deterioration when the settings increase from 10D25N to 50D50N, with the highest deterioration being observed in FIPS. DE-PSO, which has similar characteristics to FIPSaDE, shows moderate performances regardless of the settings.

Table 4. Results of average error values for 10D test problem and $N = 25$ (10D25N).

Function	FIPS	DE	SaDE	DE-PSO	FIPSaDE
F1	1.6913E+00	5.6843E-14	**1.8948E-15**	5.6843E-14	7.7548E-11
F2	1.7168E+00	2.5199E-01	**5.6843E-14**	1.8182E+01	9.2022E-11
F3	4.8917E+03	6.3587E+05	3.3801E+04	5.8865E+04	**3.0103E+01**
F4	1.0672E+01	2.0546E+02	**7.0025E-11**	2.1796E+01	9.2033E-11
F5	**0.0000E+00**	9.1538E+02	6.6696E-13	2.3041E-12	8.9070E-11
F6	5.0970E+01	2.6962E+00	1.2781E+00	1.3425E+05	**5.3154E-01**
F7	1.2671E+03	**7.6374E-02**	1.2670E+03	1.8818E-01	1.2670E+03
F8	2.0217E+01	2.0227E+01	2.0384E+01	**2.0081E+01**	2.0110E+01
F9	8.0601E+00	**5.6843E-14**	2.6532E-01	**5.6843E-14**	4.9748E-01
F10	8.6778E+00	1.6527E+01	6.9076E+00	1.0315E+01	**5.8371E+00**
F11	**1.7263E+00**	6.9768E+00	3.0350E+00	1.7695E+00	2.3054E+00
F12	2.8017E+02	**2.4992E+01**	5.6407E+02	1.5613E+03	3.0828E+02
F13	5.6892E-01	**2.6653E-02**	8.8579E-01	4.2952E-01	3.2816E-01
F14	2.1172E+00	3.0997E+00	3.1387E+00	2.2248E+00	**1.9601E+00**
F15	**1.8014E+02**	-	2.8100E+02	1.9472E+02	3.1617E+02
F16	**1.0019E+02**	-	1.0218E+02	1.1402E+02	1.0514E+02
F17	1.0719E+02	1.0722E+03	1.1966E+02	1.4561E+02	**1.0136E+02**
F18	7.6687E+02	1.6996E+03	**7.0154E+02**	7.8021E+02	7.8756E+02
F19	7.7130E+02	1.8139E+03	**6.5847E+02**	7.5198E+02	7.2585E+02
F20	7.4034E+02	1.7606E+03	6.8165E+02	7.6896E+02	**6.4081E+02**
F21	6.9422E+02	1.2334E+03	7.0185E+02	**5.4388E+02**	7.3634E+02
F22	7.1363E+02	1.2253E+03	7.7433E+02	**7.4833E+02**	7.5392E+02
F23	7.8007E+02	1.2661E+03	8.5139E+02	**7.9163E+02**	7.9806E+02
F24	2.8126E+02	7.4243E+02	**2.0000E+02**	5.3977E+02	2.2654E+02
F25	1.7506E+03	7.4292E+02	1.7507E+03	**5.4948E+02**	1.7495E+03
h_{win}	4	4	**6**	**6**	**6**

Table 5. Results of average error values for 30D test problem and $N = 30$ (30D30N).

Function	FIPS	DE	SaDE	DE-PSO	FIPSaDE
F1	1.0003E+03	**1.3264E-14**	1.0232E-13	1.3873E+02	6.6317E-14
F2	7.4895E+03	8.1760E+03	5.9137E+02	3.3581E+02	**2.3457E-12**
F3	9.6860E+06	1.2073E+07	3.0102E+06	1.4827E+06	**8.9114E+04**
F4	1.3059E+04	3.2880E+04	8.8330E+03	4.2988E+02	**3.2211E+01**
F5	5.0737E+03	1.0061E+04	4.8601E+03	**1.9268E+03**	3.1928E+03
F6	4.0001E+07	9.1524E+00	1.4404E+02	4.5495E+06	**1.1960E+00**
F7	4.6963E+03	**1.5056E-02**	4.6963E+03	2.4008E-02	4.6963E+03
F8	2.0904E+01	2.0861E+01	2.1049E+01	**2.0349E+01**	2.0887E+01
F9	9.1200E+01	**1.5158E-14**	1.8278E+01	2.2971E+01	1.0267E+01
F10	1.1387E+02	1.8811E+02	1.1745E+02	6.2888E+01	**4.1795E+01**
F11	2.2143E+01	3.5289E+01	3.5379E+01	2.1214E+01	**2.0232E+01**
F12	9.3414E+04	**5.5912E+03**	1.6091E+04	1.8567E+04	6.4080E+03
F13	6.8397E+00	**2.5628E-01**	1.2032E+01	2.1437E+00	2.3846E+00
F14	**1.1821E+01**	1.2741E+01	1.3538E+01	1.2031E+01	1.1989E+01
F15	4.8635E+02	-	**3.2749E+02**	3.9304E+02	3.3594E+02
F16	2.4964E+02	-	1.8945E+02	2.5133E+02	**1.2952E+02**
F17	2.4655E+02	-	2.9869E+02	2.9779E+02	**1.4873E+02**
F18	9.1085E+02	1.2349E+03	**9.0809E+02**	9.0925E+02	9.1217E+02
F19	9.0797E+02	1.2258E+03	**9.0760E+02**	9.1250E+02	9.1158E+02
F20	9.1196E+02	1.2834E+03	9.1304E+02	9.0907E+02	**9.0881E+02**
F21	9.3243E+02	1.4813E+03	**5.0000E+02**	6.6047E+02	5.3210E+02
F22	8.9686E+02	1.6756E+03	9.4560E+02	**8.5655E+02**	8.9115E+02
F23	**5.9064E+02**	1.5085E+03	6.1697E+02	8.2535E+02	5.9916E+02
F24	4.1547E+02	1.4310E+03	**2.0000E+02**	6.4398E+02	2.2529E+02
F25	1.6313E+03	1.4382E+03	1.6448E+03	**2.8872E+02**	1.6171E+03
h_{win}	2	5	5	4	**9**

Table 6. Results of average error values for 50D test problem and $N = 50$ (50D50N).

Function	FIPS	DE	SaDE	DE-PSO	FIPSaDE
F1	2.9123E+03	**5.6843E-14**	5.1044E-10	6.6018E+02	8.2423E-14
F2	1.9747E+04	2.6854E+04	4.0747E+03	1.8025E+03	**2.7853E-12**
F3	4.6307E+07	2.7198E+07	7.2181E+06	4.5704E+06	**9.4374E+04**
F4	2.9802E+04	9.4951E+04	2.7156E+04	2.5763E+03	**8.5264E+01**
F5	1.2415E+04	2.2436E+04	9.7093E+03	6.1975E+03	**5.5761E+03**
F6	1.1081E+08	1.7165E+01	3.4873E+02	4.1990E+07	**1.7940E+00**
F7	6.1953E+03	**3.5798E-04**	6.1953E+03	1.1065E-02	6.1953E+03
F8	2.1079E+01	2.1056E+01	2.1194E+01	**2.1009E+01**	2.1084E+01
F9	1.9239E+02	**5.6843E-14**	9.1163E+01	8.3410E+01	1.5024E+01
F10	2.5784E+02	4.8818E+02	2.8407E+02	1.3299E+02	**9.3924E+01**
F11	4.4393E+01	5.7573E+03	7.0201E+01	4.5260E+01	**3.8998E+01**
F12	4.2248E+05	2.3101E+04	7.0532E+04	1.0593E+05	**1.6649E+04**
F13	1.5238E+01	**5.4804E-01**	2.7075E+01	4.4169E+00	4.7292E+00
F14	2.1285E+01	2.2126E+01	2.3325E+01	**2.0844E+01**	2.1229E+01
F15	4.5708E+02	-	**2.8346E+02**	3.8232E+02	3.0640E+02
F16	2.3026E+02	-	1.9536E+02	1.9984E+02	**7.8014E+01**
F17	2.5320E+02	-	3.1450E+02	1.5794E+02	**7.1639E+01**
F18	1.0033E+03	1.4072E+03	9.3804E+02	**9.2262E+02**	9.5502E+02
F19	9.7666E+02	1.3165E+03	9.4219E+02	**9.1941E+02**	9.4735E+02
F20	9.7761E+02	1.3865E+03	9.4763E+02	**9.1692E+02**	9.4149E+02
F21	1.2067E+03	7.9999E+02	6.6371E+02	8.8581E+02	**6.3703E+02**
F22	9.5024E+02	1.2092E+03	9.5214E+02	**9.0646E+02**	9.2008E+02
F23	**6.3049E+02**	7.5279E+02	6.7595E+02	9.2719E+02	6.9500E+02
F24	8.5524E+02	1.2819E+03	2.0000E+02	5.6974E+02	**2.0000E+02**
F25	1.6662E+03	1.2397E+03	1.6795E+03	**2.1600E+02**	1.6539E+03
h_{win}	1	4	1	7	**12**

FIPSaDE with its original variants, that is, FIPS and SaDE are further analyzed based on the best, the mean and standard deviation of f_{best} for different configurations of problem dimensionality and population size for all functions. The f_{best} obtained by FIPSaDE, FIPS and SaDE for all functions are shown in Tables 7–9, summarizing the best, the mean and standard deviation for f_{best} over a number of runs, which can be either 20 runs or 30 runs. The best, mean and standard deviation results of f_{best} are denoted as *best*-f_{best}, *mean*-f_{best} and *std*-f_{best}, respectively. For each table, the algorithm producing the best results for the same function is bolded and its frequency, h_{win} is calculated at the bottom.

For the setting of 10D25N, FIPSaDE has the highest h_{win} for *best*-f_{best}, *mean*-f_{best} and *std*-f_{best}. FIPS and SaDE have similar h_{win} values *best*-f_{best}, *mean*-f_{best} for the same setting. When the setting increases to 30D30N and 50D50N, FIPSaDE is still associated with the highest h_{win} for *best*-f_{best}, *mean*-f_{best} and *std*-f_{best}. Therefore, FIPSaDE shows better results for *best*-f_{best}, *mean*-f_{best} and *std*-f_{best} with the increase of problem dimensionality and population size.

FIPS's h_{win} values for *best*-f_{best}, *mean*-f_{best} deteriorate from 10D25N to 50D50N. Based on h_{win} for *std*-f_{best}, FIPS has consistent values that are 6 to 7, regardless of the problem dimensionality and population size. The finding indicates that the frequency of FIPS producing the highest *best*-f_{best} and *mean*-f_{best} among the algorithms deteriorates with the increase of problem dimensionality and population size. Additionally, FIPS's frequency in producing the lowest variants of solutions is consistent across the problem dimensionality and population size.

SaDE shows a decreasing trend as FIPS for *best*-f_{best}, *mean*-f_{best} when the settings change from 10D25N to 50D50N. However, the decreasing trend is not as drastic as FIPS. Based on the comparisons of *std*-f_{best}, SaDE and FIPS have similar values for the settings, except 10D25N. SaDE's *std*-f_{best} = 10 is higher than FIPS for the lower problem dimensionality and population size, 10D25N. The finding indicates that SaDE's frequency in producing lowest variations of solutions is slightly lower at high problem dimensionality and population size.

Table 7. *Best-f_{best}*, *mean-f_{best}* and *std-f_{best}* obtained by FIPS, SaDE and FIPSaDE algorithms for the settings of 10*D*25*N*.

Function	*Best-f_{best}*			*Mean-f_{best}*			*Std-f_{best}*		
	FIPS	**SaDE**	**FIPSaDE**	**FIPS**	**SaDE**	**FIPSaDE**	**FIPS**	**SaDE**	**FIPSaDE**
F1	**−450.00**	**−450.00**	**−450.00**	−448.31	**−450.00**	**−450.00**	2.70	**0.00**	**0.00**
F2	**−450.00**	**−450.00**	**−450.00**	−448.28	**−450.00**	**−450.00**	2.71	**0.00**	**0.00**
F3	−449.99	3085.63	**−450.00**	−439.33	33350.51	**−450.00**	15.16	27,313.82	**0.00**
F4	−449.99	**−450.00**	**−450.00**	−439.33	**−450.00**	**−450.00**	15.16	**0.00**	**0.00**
F5	**−310.00**	**−310.00**	**−310.00**	**−310.00**	**−310.00**	**−310.00**	**0.00**	**0.00**	**0.00**
F6	390.30	**390.00**	**390.00**	440.97	391.28	**390.53**	86.73	1.66	**1.38**
F7	**1087.05**	**1087.05**	**1087.05**	**1087.05**	**1087.05**	**1087.05**	0.03	**0.00**	**0.00**
F8	−119.88	−119.78	**−120.00**	−119.78	−119.62	**−119.89**	**0.05**	0.09	0.10
F9	−329.01	**−330.00**	**−330.00**	−321.94	**−329.73**	−329.50	3.60	0.52	**0.51**
F10	−327.02	**−328.01**	**−328.01**	−321.13	−323.09	**−324.16**	3.47	**2.57**	3.21
F11	90.81	90.67	**90.54**	**91.73**	93.03	92.31	**0.68**	1.82	1.30
F12	**−460.00**	**−460.00**	**−460.00**	**−179.83**	104.07	−151.72	**678.16**	1064.47	702.17
F13	−129.72	−129.46	**−129.82**	−129.43	−129.11	**−129.67**	0.15	0.21	**0.08**
F14	−298.72	−297.77	**−299.74**	−297.88	−296.86	**−298.04**	0.45	**0.38**	0.61
F15	**121.83**	178.14	160.45	**300.14**	401.00	436.17	**129.47**	148.90	140.78
F16	**120.00**	209.94	211.25	**220.19**	222.18	225.14	20.73	7.33	**6.98**
F17	207.88	214.00	**201.17**	227.19	239.66	**221.36**	**8.66**	17.61	8.77
F18	**310.00**	**310.00**	**310.00**	776.87	**711.54**	7979.56	239.17	226.16	**169.84**
F19	**310.00**	**310.00**	**310.00**	781.30	**668.47**	735.85	242.09	**219.37**	219.42
F20	**310.00**	**310.00**	**310.00**	750.34	913.31	**650.81**	277.37	346.97	**230.60**
F21	**660.00**	**660.00**	**660.00**	**1054.22**	1061.85	1096.34	275.44	276.19	**220.53**
F22	**660.91**	1109.44	1084.87	**1073.63**	1134.33	1113.92	133.25	**22.27**	29.15
F23	**914.00**	919.47	919.47	**1140.07**	1211.39	1158.06	**178.64**	254.86	271.52
F24	**460.00**	**460.00**	**460.00**	533.32	**460.00**	486.54	130.48	**0.00**	145.39
F25	**92.99**	1991.62	1995.92	**1948.74**	1020.66	2009.47	344.45	6.99	**5.36**
h_{win}	15	14	**20**	10	9	**13**	6	10	**14**

Table 8. *Best-f_{best}*, *mean-f_{best}* and *std-f_{best}* obtained by FIPS, SaDE and FIPSaDE algorithms for the settings of 30*D*30*N*.

Function	*Best-f_{best}*			*Mean-f_{best}*			*Std-f_{best}*		
	FIPS	**SaDE**	**FIPSaDE**	**FIPS**	**SaDE**	**FIPSaDE**	**FIPS**	**SaDE**	**FIPSaDE**
F1	−297.49	**−450.00**	**−450.00**	550.32	**−450.00**	**−450.00**	474.17	**0.00**	**0.00**
F2	2925.32	−334.80	**−450.00**	7039.46	141.37	**−450.00**	2090.36	394.74	**0.00**
F3	4,662,113.70	807,037.41	**19,291.79**	9,685,580.44	3,009,720.45	**88,664.15**	2,937,448.59	1,504,001.37	**63,037.56**
F4	3522.98	3339.36	**−450.00**	12,609.08	8382.96	**−417.79**	3289.58	3329.53	**68.06**
F5	2752.59	2851.22	**1828.14**	4763.71	4550.14	**2882.76**	1143.85	1175.35	**534.97**
F6	1,311,610.25	393.12	**390.00**	40,001,127.07	534.04	**391.20**	41,151,416.49	195.57	**1.86**
F7	**4516.29**	**4516.29**	**4516.29**	**4516.29**	**4516.29**	**4516.29**	**0.00**	**0.00**	**0.00**
F8	−119.18	−119.08	**−119.24**	−119.10	−118.95	**−119.11**	**0.05**	0.06	**0.05**
F9	−274.12	−325.88	**−329.01**	−238.80	−311.72	**−319.73**	15.94	7.87	**4.57**
F10	−257.68	−303.46	**−307.12**	−216.13	−212.55	**−288.20**	19.61	52.41	**9.80**
F11	108.32	110.59	**104.48**	112.14	125.38	**110.23**	**1.49**	5.28	2.56
F12	44,816.64	1924.29	**−457.54**	92,954.31	15,631.41	**5948.00**	35,286.01	13,703.43	**9314.81**
F13	−125.85	−120.51	**−128.55**	−123.16	−117.97	**−127.62**	1.59	1.15	**0.59**
F14	−288.87	−286.68	**−289.13**	**−288.18**	−286.46	−288.01	0.33	**0.13**	0.50
F15	518.41	**230.34**	320.00	606.35	**447.49**	455.94	**45.94**	106.58	97.05
F16	227.57	179.27	**165.78**	369.64	309.45	**249.52**	151.59	138.42	**125.34**
F17	238.75	297.00	**172.70**	366.55	418.69	**268.73**	141.11	**92.36**	109.60
F18	916.04	**810.00**	915.42	920.85	**918.09**	922.17	6.09	21.18	**5.81**
F19	851.22	**810.00**	915.96	917.97	**917.60**	921.58	19.54	20.83	**4.63**
F20	915.69	915.39	**810.00**	921.96	923.04	**918.81**	7.84	**4.70**	20.93
F21	953.46	**860.00**	**860.00**	1292.43	**860.00**	892.10	260.10	**0.00**	131.12
F22	1213.63	1245.75	**1209.77**	1256.86	1305.60	**1251.15**	**22.58**	41.09	23.76
F23	901.56	896.74	**894.16**	**950.64**	976.97	959.16	**97.57**	133.89	158.86
F24	550.61	**460.00**	**460.00**	675.47	**460.00**	485.29	76.41	**0.00**	138.53
F25	1880.42	1891.12	**1868.00**	1891.27	1904.77	**1877.14**	5.85	6.17	**4.40**
h_{win}	1	7	**22**	3	7	**18**	6	6	**16**

Table 9. *Best-*f_{best}, *mean-*f_{best} and *std-*f_{best} obtained by FIPS, SaDE and FIPSaDE algorithms for the settings of 50D50N.

Function	*Best-*f_{best}			*Mean-*f_{best}			*Std-*f_{best}		
	FIPS	**SaDE**	**FIPSaDE**	**FIPS**	**SaDE**	**FIPSaDE**	**FIPS**	**SaDE**	**FIPSaDE**
F1	1153.25	450.00	**−450.00**	2462.33	450.00	**−450.00**	1044.25	**0.00**	**0.00**
F2	12,312.68	967.61	**−450.00**	19,296.90	3624.73	**−450.00**	2864.28	1944.03	**0.00**
F3	19,742,966.18	**1.55**	26,213.65	46,306,427.65	6,442,716.38	**93,924.17**	18,623,890.22	2,656,381.58	**54,148.58**
F4	18,353.52	11,796.54	**−448.82**	29,351.56	26,706.41	**−364.74**	7366.24	7776.96	**94.60**
F5	8344.21	7515.12	**3823.55**	12,104.89	9399.31	**5266.12**	1713.64	1378.09	**924.29**
F6	13,540,381.90	490.65	**390.00**	110,811,270.60	738.73	**391.79**	55,887,725.95	277.30	**2.03**
F7	**6015.32**	**6015.32**	**6015.32**	**6015.32**	**6015.32**	**6015.32**	**0.00**	**0.00**	**0.00**
F8	**−118.99**	118.75	−118.97	**−118.92**	118.81	−118.92	**0.03**	0.04	**0.03**
F9	−174.99	208.21	**−326.02**	−137.61	238.84	**−314.98**	24.75	15.11	**4.92**
F10	−136.98	0.90	**−261.35**	−72.16	60.49	**−236.08**	36.26	73.04	**16.82**
F11	129.40	156.93	**120.76**	134.39	160.20	**129.00**	2.63	**1.75**	3.64
F12	241,193.33	19,055.20	**−154.39**	422,016.01	70,072.02	**16,188.50**	115,690.78	40,227.17	**11,162.45**
F13	−116.34	99.05	**−126.80**	−114.76	102.93	**−125.27**	1.39	2.25	**0.94**
F14	−279.28	276.38	**−280.33**	−278.71	276.68	**−278.77**	0.30	**0.16**	0.53
F15	426.38	**320.00**	323.00	577.08	**403.46**	426.40	**38.77**	81.82	86.84
F16	244.32	191.77	**160.24**	350.26	315.36	**198.01**	105.18	78.92	**37.68**
F17	283.90	378.02	**170.74**	373.20	434.50	**191.64**	102.40	56.01	**11.83**
F18	935.54	**810.00**	938.12	1013.28	**984.04**	965.02	41.81	34.89	**17.86**
F19	942.46	935.18	**929.32**	986.66	**952.19**	957.35	28.70	**11.17**	15.08
F20	917.00	939.19	**916.08**	987.61	957.63	**951.49**	37.23	15.20	**14.64**
F21	1412.69	**860.00**	**860.00**	1566.71	1023.71	**997.03**	**37.54**	258.81	246.13
F22	1288.45	1268.20	**1243.66**	1310.24	1312.14	**1280.08**	**9.76**	29.34	24.74
F23	928.47	899.18	**899.12**	**990.49**	1035.95	1055.00	**105.67**	203.07	216.18
F24	728.10	**460.00**	**460.00**	1115.24	**460.00**	**460.00**	301.48	**0.00**	**0.00**
F25	1912.16	1915.96	**1902.75**	1926.19	1939.45	**1913.91**	**5.80**	9.62	7.03
h_{win}	2	6	**21**	3	5	**21**	7	6	**17**

The overall findings from the comparisons show that FIPSaDE has the highest probability of producing a better solution compared to its original variants, which are FIPS and SaDE, with an increase in problem dimensionality and population size. An EC algorithm commonly uses a large population size to solve an optimization problem with high dimensionality. However, the approach may be more effective for FIPSaDE than FIPS and SaDE. Based on the comparisons of h_{win} for *best-*f_{best} and *mean-*f_{best}, FIPSaDE has the highest values, followed by FIPS and SaDE. Therefore, FIPSaDE consistently produces a group of better-quality solutions than its original variants when problem dimensionality and population size increase.

For most tested functions, FIPSaDE proves to produce better *best-*f_{best} and *mean-*f_{best} in the swarm space compared to the other algorithms as the problem space increases. FIPSaDE, a hybrid of both FIPS and SaDE, can maneuver and manage the swarm solutions more effectively. Therefore, FIPSaDE improves performance in finding the best fitness point as the problem space increases.

The variable *mean-*f_{best} for FIPSaDE, FIPS and SaDE is denoted as $\mu_{FIPSaDE}$, μ_{FIPS} and μ_{SaDE}, respectively. A t-test is performed with a significance level, α of 0.05 to evaluate the hypothesis whether $\mu_{FIPSaDE}$ of the proposed algorithm is better than μ_{FIPS} and μ_{SaDE} or not, as shown below.

$$H_0 : \mu_{FIPSaDE} \geq \mu_{FIPS}$$
$$H_1 : \mu_{FIPSaDE} < \mu_{FIPS}$$

$$H_0 : \mu_{FIPSaDE} \geq \mu_{SaDE}$$
$$H_1 : \mu_{FIPSaDE} < \mu_{SaDE}$$

The results of hypothesis tests for the paired FIPSaDE-FIPS and FIPS-SaDE based on different settings of problem dimensionality and population size are shown in Table 10. If the p-values are less α, they are bolded, indicating that the associated H_0 is rejected. As a result, there is enough evidence to accept H_1. The p-values associated with rejecting H_0 are bolded and its frequency is denoted as $h_{(-H_0)}$. At the bottom of the table, $h_{(-H_0)}$ shows the frequency H_0 being rejected for different settings. For the setting $10D25N$, $\mu_{FIPSaDE}$ is significantly better than μ_{FIPS} and μ_{SaDE} in 10 and 9 functions, respectively. When the setting increases from $10D25N$ to $50D50N$, the values $h_{(-H_0)}$ for FIPSaDE-FIPS increase from 10 to 18 and 22. Therefore, the frequency that FIPSaDE is significantly better than FIPS increases. When similar comparisons are made for FIPSaDE-SaDE, an increasing trend is observed, but it is less obvious. The findings about $h_{(-H_0)}$ show that the performances of FIPSaDE are significantly better than FIPS and SaDE when the problem dimensionality and population size increase.

Table 10. Hypothesis test between $\mu_{FIPSaDE}$, μ_{FIPS} and μ_{SaDE} for the settings of $10D25N$, $30D30N$ and $50D50N$.

Function	**10*D*25*N***				**30*D*30*N***				**50*D*50*N***			
	***t*-Value**		***p*-Value**		***t*-Value**		***p*-Value**		***t*-Value**		***p*-Value**	
	FIPSaDE -FIPS	**FIPSaDE -SaDE**	**FIPSaDE -FIPS**	**FIPSaDE -SaDE**	**FIPSaDE -FIPS**	**FIPSaDE -SaDE**	**FIPSaDE -FIPS**	**FIPSaDE -SaDE**	**FIPSaDE -FIPS**	**FIPSaDE -SaDE**	**FIPSaDE -FIPS**	**FIPSaDE -SaDE**
F1	−3.44	28.07	**0.00**	1.00	−11.55	5.60	**0.00**	1.00	−12.47	−7.20E+12	**0.00**	**0.00**
F2	−3.47	42.75	**0.00**	1.00	−19.62	−8.21	**0.00**	**0.00**	−30.83	−9.37	**0.00**	**0.00**
F3	−3.86	−6.78	**0.00**	**0.00**	−17.89	−10.63	**0.00**	**0.00**	−11.10	−10.69	**0.00**	**0.00**
F4	−3.86	0.31	**0.00**	0.62	−21.69	−14.47	**0.00**	**0.00**	−18.04	−15.57	**0.00**	**0.00**
F5	66.38	63.86	1.00	1.00	−8.16	−7.07	**0.00**	**0.00**	−15.71	−11.14	**0.00**	**0.00**
F6	−3.18	−1.90	**0.00**	**0.03**	−5.32	−4.00	**0.00**	**0.00**	−8.87	−5.60	**0.00**	**0.00**
F7	−1.00	0.00	0.16	0.50	−18.08	−18.08	**0.00**	**0.00**	−1.76	−4.18	**0.04**	**0.00**
F8	−5.46	−11.41	**0.00**	**0.00**	−1.37	−11.75	0.09	**0.00**	0.51	−2.05E+04	0.69	**0.00**
F9	−11.41	1.76	**0.00**	0.96	−26.73	−4.82	**0.00**	**0.00**	−31.44	−155.85	**0.00**	**0.00**
F10	−3.49	−1.43	**0.00**	0.08	−18.01	−7.77	**0.00**	**0.00**	−18.34	−17.69	**0.00**	**0.00**
F11	2.16	−1.79	0.98	**0.04**	−3.53	−14.13	**0.00**	**0.00**	−5.37	−34.53	**0.00**	**0.00**
F12	0.16	−1.10	0.56	0.14	−13.06	−3.20	**0.00**	**0.00**	−15.62	−5.77	**0.00**	**0.00**
F13	−7.76	−13.78	**0.00**	**0.00**	−14.38	−40.84	**0.00**	**0.00**	−27.94	−417.89	**0.00**	**0.00**
F14	−1.14	−8.98	0.13	**0.00**	1.52	−16.31	0.93	**0.00**	−0.41	−4.44E+03	0.34	**0.00**
F15	3.90	0.94	1.00	0.82	−6.45	0.32	**0.00**	0.63	−7.09	0.86	**0.00**	0.80
F16	1.24	1.60	0.89	0.94	−3.34	−1.76	**0.00**	**0.04**	−6.09	−6.00	**0.00**	**0.00**
F17	−2.59	−5.09	**0.01**	**0.00**	−3.00	−5.73	**0.00**	**0.00**	−7.88	−18.97	**0.00**	**0.00**
F18	0.39	1.67	0.65	0.95	0.86	1.02	0.80	0.84	−4.65	1.94	**0.00**	0.97
F19	−0.76	1.19	0.22	0.88	0.98	1.02	0.84	0.84	−4.04	1.23	**0.00**	0.89
F20	−1.51	−3.45	0.07	**0.00**	−0.77	−1.08	0.22	0.14	−4.04	−1.30	**0.00**	0.10
F21	0.65	0.53	0.74	0.70	−7.53	1.34	**0.00**	0.91	−10.23	−0.33	**0.00**	0.37
F22	1.62	−3.05	0.94	**0.00**	−0.96	−6.28	0.17	**0.00**	−5.07	−3.74	**0.00**	**0.00**
F23	0.30	−0.78	0.62	0.62	0.25	−0.47	0.60	0.32	1.20	0.29	0.88	0.61
F24	−1.30	1.00	0.10	0.84	−6.58	1.00	**0.00**	0.84	−9.72	−1.01	**0.00**	0.16
F25	0.97	−0.74	0.83	0.23	−10.57	−19.98	**0.00**	**0.00**	−6.02	−9.59	**0.00**	**0.00**
$h_{(-H_0)}$	-	-	10	9	-	-	18	17	-	-	22	18

6. Conclusions

In this study, a hybrid of FIPS and SaDE called FIPSaDE is proposed, and its performance is validated by running the algorithm against the benchmark functions while also being compared to their respective original version, the SaDE and FIPS as well as it variants, such as DE and DE-PSO. The self-adaptation strategy of SaDE is adapted and maneuvered by the FIPS particle swarm, preventing the solutions from being trapped in the local region. Each algorithm can adaptively adjust the parameter values while the swarm is searching for the best solution needed. Based on different configurations of problem dimensionality and population sizes, the FIPSaDE algorithm consistently has the highest frequency of having lowest average errors than FIPS, DE, SaDE and DE-PSO. The frequency analysis of h_{win} and the hypothesis test show that FIPSaDE performs better than its respective original versions in terms of the best and mean of f_{best} as the problems' dimensionality increases. Future research will investigate the strength of proposed algorithms with other benchmark test functions. Since the current FIPSaDE is only tested on one topology, that is, Four Clusters, the impact of the other four other FIPS topologies, namely, All, Ring, Pyramid, Square, should be investigated in the future for their effects on the hybrid's performances. Moreover, various performance metrics could be applied to strengthen the comparison among the algorithms. The study of the convergence profiles and coverage curve of the proposed algorithm as compared to other algorithms could also be investigated in the future. Furthermore, investigation should be conducted to broaden the comparison among the hybridization methods.

Author Contributions: Conceptualization, S.L.W., S.H.A. and T.F.N.; Data curation, S.L.W., S.H.A., H.I. and P.R.; Formal analysis, S.L.W., S.H.A. and P.R.; Funding acquisition, T.F.N.; Investigation, S.L.W. and S.H.A.; Methodology, S.L.W., S.H.A., H.I. and T.F.N.; Project administration, T.F.N.; Resources, S.H.A., H.I. and P.R.; Software, S.H.A., H.I. and P.R.; Supervision, S.L.W., H.I. and T.F.N.; Validation, S.L.W. and T.F.N.; Visualization, S.L.W., S.H.A. and P.R.; Writing—Original draft, S.L.W., S.H.A., H.I. and T.F.N.; Writing—Review and editing, S.L.W., H.I. and T.F.N. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Higher Education Malaysia under the Fundamental Research Grant Scheme (FRGS) with grant number FRGS/1/2022/ICT02/UPSI/02/1.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article. Further inquires can be directed to the corresponding author.

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

Abbreviations

The following abbreviations are used in this manuscript:

ABC	Artificial bee colony
ACA	Ant colony algorithm
CGSS	Centre of Global Sustainability Studies
DE	Differential evolution
EC	Evolutionary computation
FIPS	Fully informed particle swarm
FIPSaDE	FIPS-SaDE
GUI	Graphical user interface
NP	nondeterministic polynomial
PSO	Particle swarm optimization
UPSI	Universiti Pendidikan Sultan Idris
USM	Universiti Sains Malaysia
SaDE	Self-adaptive differential evolution

References

1. Mendes, R.; Kennedy, J.; Neves, J. The fully informed particle swarm: Simpler, maybe better. *IEEE Trans. Evol. Comput.* **2004**, *8*, 204–210. [CrossRef]
2. Zhalechian, M.; Tavakkoli-Moghaddam, R.; Rahimi, Y. A self-adaptive evolutionary algorithm for a fuzzy multi-objective hub location problem: An integration of responsiveness and social responsibility. *Eng. Appl. Artif. Intell.* **2017**, *62*, 1–16. [CrossRef]
3. Surekha, P.; Archana, N.; Sumathi, S. Unit commitment and economic load dispatch using self adaptive differential evolution. *Wseas Trans. Power Syst.* **2012**, *7*, 159–171.
4. Chen, Y.; Mahalex, V.; Chen, Y.; He, R.; Liu, X. Optimal satellite orbit design for prioritized multiple targets with threshold observation time using self-adaptive differential evolution. *J. Aerosp. Eng.* **2015**, *28*, 04014066. [CrossRef]
5. Das, S.; Mullick, S.S.; Suganthan, P.N. Recent advances in differential evolution–an updated survey. *Swarm Evol. Comput.* **2016**, *27*, 1–30. [CrossRef]
6. Al-Anzi, F.S.; Allahverdi, A. A self-adaptive differential evolution heuristic for two-stage assembly scheduling problem to minimize maximum lateness with setup times. *Eur. J. Oper. Res.* **2007**, *182*, 80–94. [CrossRef]
7. Ali, M.; Ahn, C.W. An optimized watermarking technique based on self-adaptive DE in DWT-SVD transform domain. *Signal Process.* **2014**, *94*, 545–556. [CrossRef]
8. Brest, J.; Greiner, S.; Boskovic, B.; Mernik, M.; Zumer, V. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Trans. Evol. Comput.* **2006**, *10*, 646–657. [CrossRef]
9. Parouha, R.P.; Verma, P. A systematic overview of developments in differential evolution and particle swarm optimization with their advanced suggestion. *Appl. Intell.* **2022**, *52*, 10448–10492. [CrossRef]

10. Storn, R.; Price, K. Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
11. Bilal; Pant, M.; Zaheer, H.; Garcia-Hernandez, L.; Abraham, A. Differential Evolution: A review of more than two decades of research. *Eng. Appl. Artif. Intell.* **2020**, *90*, 103479. [CrossRef]
12. Wang, S.L.; Morsidi, F.; Ng, T.F.; Budiman, H.; Neoh, S.C. Insights into the effects of control parameters and mutation strategy on self-adaptive ensemble-based differential evolution. *Inf. Sci.* **2020**, *514*, 203–233. [CrossRef]
13. Al-Dabbagh, R.D.; Neri, F.; Idris, N.; Baba, M.S. Algorithmic design issues in adaptive differential evolution schemes: Review and taxonomy. *Swarm Evol. Comput.* **2018**, *43*, 284–311. [CrossRef]
14. Das, S.; Suganthan, P.N. Differential evolution: A survey of the state-of-the-art. *IEEE Trans. Evol. Comput.* **2011**, *15*, 4–31. [CrossRef]
15. Neri, F.; Tirronen, V. Recent advances in differential evolution: A survey and experimental analysis. *Artif. Intell. Rev.* **2010**, *33*, 61–106. [CrossRef]
16. Parouha, R.P.; Verma, P. An innovative hybrid algorithm for bound-unconstrained optimization problems and applications. *J. Intell. Manuf.* **2022**, *33*, 1273–1336. [CrossRef]
17. Tao, X.; Guo, W.; Li, Q.; Ren, C.; Liu, R. Multiple scale self-adaptive cooperation mutation strategy-based particle swarm optimization. *Appl. Soft Comput.* **2020**, *89*, 106124. [CrossRef]
18. Shami, T.M.; El-Saleh, A.A.; Alswaitti, M.; Al-Tashi, Q.; Summakieh, M.A.; Mirjalili, S. Particle Swarm Optimization: A Comprehensive Survey. *IEEE Access* **2022**, *10*, 10031–10061. [CrossRef]
19. Jain, M.; Saihjpal, V.; Singh, N.; Singh, S.B. An Overview of Variants and Advancements of PSO Algorithm. *Appl. Sci.* **2022**, *12*, 8392. [CrossRef]
20. Chen, Y.; Liang, J.; Wu, Y.; He, B.; Lin, L.; Wang, Y. Self-Regulating and Self-Perception Particle Swarm Optimization with Mutation Mechanism. *J. Intell. Robot. Syst.* **2022**, *105*, 1–21. [CrossRef]
21. Wang, S.; Li, Y.; Yang, H. Self-adaptive mutation differential evolution algorithm based on particle swarm optimization. *Appl. Soft Comput.* **2019**, *81*, 105496. [CrossRef]
22. Dash, J.; Dam, B.; Swain, R. Design and implementation of sharp edge FIR filters using hybrid differential evolution particle swarm optimization. *Aeu-Int. J. Electron. Commun.* **2020**, *114*, 153019. [CrossRef]
23. Yang, X.S. Nature-inspired optimization algorithms: Challenges and open problems. *J. Comput. Sci.* **2020**, *46*, 101104. [CrossRef]
24. Mallipeddi, R.; Suganthan, P.N.; Pan, Q.K.; Tasgetiren, M.F. Differential evolution algorithm with ensemble of parameters and mutation strategies. *Appl. Soft Comput.* **2011**, *11*, 1679–1696. [CrossRef]
25. Ghosh, A.; Das, S.; Das, A.K.; Senkerik, R.; Viktorin, A.; Zelinka, I.; Masegosa, A.D. Using spatial neighborhoods for parameter adaptation: An improved success history based differential evolution. *Swarm Evol. Comput.* **2022**, *71*, 101057. [CrossRef]
26. Do, D.T.; Lee, S.; Lee, J. A modified differential evolution algorithm for tensegrity structures. *Compos. Struct.* **2016**, *158*, 11–19. [CrossRef]
27. Piotrowski, A.P. Review of differential evolution population size. *Swarm Evol. Comput.* **2017**, *32*, 1–24. [CrossRef]
28. Borowska, B. Learning Competitive Swarm Optimization. *Entropy* **2022**, *24*, 283. [CrossRef]
29. Qin, A.K.; Huang, V.L.; Suganthan, P.N. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Trans. Evol. Comput.* **2009**, *13*, 398–417. [CrossRef]
30. Cleghorn, C.W.; Engelbrecht, A. Fully informed particle swarm optimizer: Convergence analysis. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, Japan, 25–28 May 2015; pp. 164–170. [CrossRef]
31. Pant, M.; Thangaraj, R.; Grosan, C.; Abraham, A. Hybrid differential evolution-particle swarm optimization algorithm for solving global optimization problems. In Proceedings of the 2008 Third International Conference on Digital Information Management, London, UK, 13–16 November 2008; pp. 18–24.
32. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.P.; Auger, A.; Tiwari, S. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. *KanGAL Rep.* **2005**, *2005005*, 2005.