Time Adaptive Dual Particle Swarm Optimization

Ulysse Côté Allard*, Gabriel Dubé*, Richard Khoury,
Luc Lamontagne, Benoit Gosselin and François Laviolette

Abstract— **This paper presents a novel particle swarm optimization (PSO) algorithm that combines the strengths of several PSO variants into a single competitive algorithm. This novel algorithm, named Time Adaptive Dual Particle Swarm Optimization (TAD-PSO), is comprised of two specialized populations, with one focusing on exploration of the search space and the other on exploitation. The main population, specialized in exploration, uses orthogonal learning to create information-rich exemplars which intelligently guide particle movement throughout the search space. The auxiliary population uses a PSO variant known for its very fast convergence speed, and thus very high performance on unimodal problems. This population is specialized in exploitation of the interesting local minima. The main population size decays linearly, to foster exploration early and convergence in the later stages of the optimization procedure. Additionally, TAD-PSO does not have the topological structure of the swarm as an algorithm hyper-parameter, making it a fast and simple algorithm to apply to new problems. TAD-PSO was tested extensively and compared to 6 widely used PSO variants on 19 benchmark problems, for 10, 30 and 100 dimensions. TAD-PSO consistently ranked first in each dimensional space, making it a competitive optimization algorithm on both unimodal and multimodal problems.**

I. Introduction

The problem of constrained optimization has been an active area of research for decades, from purely mathematical paradigms [1] to optimization algorithms inspired by evolution [2], [3], the movements of animal swarms [4], [5], [6] and other animal behavior [7], [8], [9].

Particle Swarm Optimization (PSO) algorithms [10], [11] are loosely based on the behavior of a flock of birds collaboratively seeking the best areas to feed. The movement of a particle is guided by its individual knowledge (the best location it has personally visited), and the knowledge of the swarm (the best position collectively found by the swarm). However, problems can arise from this duality (see Sec. III).

Some PSO algorithms, in particular Comprehensive Learning Particle Swarm Optimizer (CLPSO) [12] and Orthogonal Learning Particle Swarm Optimizer (OLPSO) [13], solve this by combining the individual and collective information into a single "exemplar". CLPSO exemplars are typically poor in information, but have the advantage of partially removing the dependence on the swarm topology (see Sec. II and III). Meanwhile, OLPSO exemplars are rich in information, at the cost of function evaluations.

Ulysse Côté Allard and Benoit Gosselin are with the Depart. of Computer and Electrical Engineering; Gabriel Dubé, Richard Khoury, Luc Lamontagne and François Laviolette are with the Depart. of Computer Science and Software Engineering; Université Laval, Québec, Québec, G1V 0A6, Canada. Contact author email: ulysse.cote-allard.1@ulaval.ca
*These authors contributed equally to this work.

Another issue faced by PSO algorithms is the inherent trade-off between the exploration of the search space and the exploitation of a local minimum. Algorithms that converge rapidly on unimodal problems will tend to do so prematurely when applied to multimodal functions [14]. On the other hand, algorithms that explore a wider portion of the search space will waste function evaluations and exhibit slow convergence when applied to unimodal functions [14].

To minimize the impacts of the trade-off problem, this paper presents a novel algorithm, TAD-PSO, which employs two populations. One is specialized for the exploration of the search space, the other for the exploitation. Specifically, the main population leverages exemplars that combine the CLPSO and OLPSO methods of exemplar construction. The second is an auxiliary population based on Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients [15] (HPSO-TVAC) designed for rapid convergence. This provides comprehensive inspection of the promising local minima found by the main population.

The dual population approach of the algorithm aims at being able to solve complex and high-dimensional problems efficiently. The over-evaluation problem stemming from the OLPSO exemplar construction is addressed via a decaying population (see Sec. IV). In other words, each population is specialized for either of the two tasks of exploration or exploitation. The main population addresses complex and highly multimodal functions, while the auxiliary population addresses simple and unimodal functions. This dual population approach seems ideal for application to a wide variety of optimization problems, which are often mixtures, to varying degrees, of these two types of problems.

The remainder of the paper is organized as follows; the original PSO algorithm is presented in Sec. II. Sec. III describes relevant variants of the PSO algorithm (CLPSO, OLPSO, HPSO-TVAC). A novel algorithm, TAD-PSO, is presented in Sec. IV. Penultimately, a comprehensive experimental evaluation of TAD-PSO is conducted on 19 benchmarks in high dimensions (Sec. V). Finally, conclusions and future works are given in Sec. VI.

II. Particle Swarm Optimization

PSO [10], [16], [11] is one of the most popular optimization algorithms based on animal swarm behavior. The standard PSO algorithm emulates swarm behaviors by using simple rules to move the particles around a D-dimensional search space. "Particle" is used to refer to each minimum-seeking entity, while "swarm" is the collection of the aforementioned particles. The ith particle of the swarm at iteration time t is defined by a position vector $\mathbf{X}^{i,t} = [\mathbf{X}_1^{i,t}, \mathbf{X}_2^{i,t}, ..., \mathbf{X}_D^{i,t}]$ and a velocity vector $\mathbf{V}^{i,t} = [\mathbf{V}_1^{i,t}, \mathbf{V}_2^{i,t}, ..., \mathbf{V}_D^{i,t}]$. Additionally, each particle keeps track of its personal best visited position, denoted by $\mathbf{Pb}^{i,t}$, and the global best position, denoted by $\mathbf{Gb}^{i,t}$, found by the particle and its neighbors.

As this paper focuses on minimization problems, the best visited position is defined as the position associated with the lowest function value ("best fitness"). The vectors $\mathbf{X}^{i,t}$ and $\mathbf{V}^{i,t}$ are initialized randomly and updated each generation by the following rules:

$$\mathbf{V}_d^{i,t+1} = w^t\mathbf{V}_d^{i,t} + c_1\mathbf{r}_{1,d}(\mathbf{Pb}_d^{i,t} - \mathbf{X}_d^{i,t}) + c_2\mathbf{r}_{2,d}(\mathbf{Gb}_d^{i,t} - \mathbf{X}_d^{i,t}) \tag{1}$$

$$\mathbf{X}^{i,t+1} = \mathbf{X}^{i,t} + \mathbf{V}^{i,t+1}, \tag{2}$$

where d is the considered dimension of the vector; w^t is the "inertia weight" at iteration t, a scalar that determines the amount of velocity that should be conserved from the previous generation. The inertia weight decreases linearly from $w_{\max}$ to $w_{\min}$ as the iteration count t increases to the maximum number of iterations $\mathrm{maxiter}$, following the formula

$$w^t = w_{\min} + (w_{\max} - w_{\min})\frac{\mathrm{maxiter} - t}{\mathrm{maxiter}}. \tag{3}$$

This update rule has proven to significantly boost performance of PSO [17], [18], [15]. c_1 and c_2 are positive scalars serving the same role as the inertia weight for their respective component; $\mathbf{r}_{1,d}$ and $\mathbf{r}_{2,d}$ are two random numbers uniformly and independently sampled from $[0, 1]$. The term $c_1\mathbf{r}_{1,d}(\mathbf{Pb}_d^{i,t} - \mathbf{X}_d^{i,t})$ is referred to as the "cognitive component", as the guidance it offers comes from the particle's own personal knowledge. The term $c_2\mathbf{r}_{2,d}(\mathbf{Gb}_d^{i,t} - \mathbf{X}_d^{i,t})$ is referred to as the "social component", as it draws knowledge from the neighboring particles. Both the personal neighborhood best positions are named "exemplars" (i.e. locations in the search space to which a particle is attracted).

What is referred to as the "neighborhood" of a particle depends on the swarm's topological structure. Previous work has shown that, generally speaking, a small neighborhood is auspicious to solving complex or multimodal optimization problems, whereas more connected swarms are better suited for simpler or unimodal problems [19], [14], [11]. Two of the more commonly featured topologies are the global topology, in which the neighborhood of a particle is the entire swarm, and the ring topology, in which the neighborhood of the ith particle is comprised of only the $(i-1)$, i and $(i+1)$th particles. In this paper, algorithms will default to the global topology. Exceptions are identified with an "-L", and employ a ring topology instead.

The last two important concepts in the standard PSO pertain to the search boundary range. Firstly, any function evaluation (FE) outside of the search space is wasted, as such a point would be an inadmissible solution. The problem of managing particles that leave or want to leave the search space is known as "bound handling". Although many solutions exist [20], a simple and prevalent technique [13], [15], [12] is to let particles leave the search space as they please, while evaluating only the particles that represent admissible solutions. Because exemplars are always within the search space boundaries, particles that leave the search space will naturally return within a few iterations. Secondly, a maximal velocity is enforced, to ensure that particles do not behave too wildly. Several methods are proposed [21]. The most common solution [10], [13], [15], [12] is to limit the velocity in every dimension to a user-specified percentage of the search space range. Unless specified otherwise, algorithms presented in this paper utilize those two methods, albeit with different hyper-parameter values for different algorithms (which will be specified in Sec. V).

III. Background

The original PSO algorithm has been altered by many researchers [22], [23], [24], [25] since its introduction by Kennedy and Eberhart [10] in 1995. The algorithm presented in this paper combines three versions: CLPSO [12], OLPSO [13] and HPSO-TVAC [15].

A. Comprehensive Learning Particle Swarm Optimizer

When using a global topology, the social component of any particle is only influenced by the single best particle of the swarm. This tends to lead to premature convergence when optimizing multimodal functions [14]. Moreover, when optimizing functions of several variables, a particle with an unattractive fitness might still have found optimal variable values in some dimensions. In this case, even local topologies (e.g. ring [14], random [26], von Neumann [14]) often fail to properly exploit this information, as the social component still only relies on a single particle.

Furthermore, equation (1) has two other major drawbacks. The first is the possibility of wasteful back-and-forth motions, occurring when a particle is midway between its personal best location and its neighborhood's best location. This is called the "Oscillation" problem [27], [13]. A second issue occurs when the combined influence of a particle's historical and neighborhood experiences deteriorates some component of the velocity vector, causing the particle to fly in the wrong direction in some dimensions. This is called the "Two steps forward, one step back" problem [28], [13].

To solve these two problems, the velocity update equation utilized by the Comprehensive Learning PSO (CLPSO) algorithm builds a single exemplar which replaces both the social and the cognitive components of the original PSO. Additionally, as CLPSO also allows any particle to be part of any other particle's exemplar, the issues of premature convergence and sub-optimal use of information in high-dimensional spaces are also considerably lessened [12].

The velocity update equation of CLPSO is as follows:

$$\mathbf{V}_d^{i,t+1} = w^t\mathbf{V}_d^{i,t} + c\,\mathbf{r}_d(\mathbf{Ecl}_d^{i,t} - \mathbf{X}_d^{i,t}), \tag{4}$$

where $\mathbf{Ecl}^{i,t}$ is the exemplar followed by the ith particle (at iteration t) and obtained via Algorithm 1. While the topological structure of CLPSO is global, equation (4) emulates a local topology to a certain extent. Indeed, by construction, it is highly unlikely that an exemplar will only depend on the swarm's best particle. Rather, it is expected that a certain number of the exemplar's dimensions will be independent to the global best. This reduces the risk of premature convergence.

Algorithm 1 CLPSO exemplar construction

```
1:  procedure Ecl$^i$
2:      exemplar ← an all-zero length $D$ array
3:      for $d = 1, \ldots, D$ do
4:          if $\mathbf{Pc}_i \leq \mathrm{rand}()$ then
5:              exemplar$[d] \leftarrow i$
6:          else
7:              $I_a \leftarrow$ Choose one from $\{1, \ldots, s\} \backslash \{i\}$
8:              $I_b \leftarrow$ Choose one from $\{1, \ldots, s\} \backslash \{i, I_a\}$
9:              $f_a \leftarrow \mathrm{Particles}[I_a].\mathrm{getBestFitness}()$
10:             $f_b \leftarrow \mathrm{Particles}[I_b].\mathrm{getBestFitness}()$
11:             if $f_a \leq f_b$ then
12:                 exemplar$[d] \leftarrow I_a$
13:             else
14:                 exemplar$[d] \leftarrow I_b$
        return exemplar
```

Note that in Algorithm 1, procedure $\mathrm{rand}()$ returns a random number uniformly sampled from $[0, 1]$; procedure $\mathrm{Particles}[i]$ returns the ith particle of the swarm; procedure $\mathrm{getBestFitness}()$ returns the best fitness ever found by the particle. Finally, $\mathbf{Pc}_i$ is the "learning probability" of particle i, as introduced in CLPSO's article [12]. This probability is calculated once for each particle at the initialization of the optimization procedure, using the following formula

$$\mathbf{Pc}_i = .05 + .45 \times \frac{\exp\left(\frac{10(i-1)}{s-1}\right) - 1}{\exp(10) - 1}, \tag{5}$$

where i is the index of the creature, and s is the population size of the swarm. For all dimensions, $\mathbf{Pc}_i$ is the probability that a particle will learn from a personal best other than its own. Finally, a detail was omitted from Algorithm 1 in order to avoid cluttering. If all dimensions of the exemplar are equal to i (i.e. the exemplar is simply the particle's own personal best), randomly choose a single dimension to learn from another particle's best location.

Each exemplar contains a list of particle indexes, rather than a list of variable values. When updating the velocity of a particle, each index needs to be replaced by the corresponding particle's personal best value in the corresponding dimension. This way, the exemplars will automatically update whenever any particle finds a new personal best position. This avoids using outdated information, which would inhibit learning. The authors of [12] suggest creating a new exemplar whenever seven generations go by without a particle improving its best fitness. This number G of generations between exemplar constructions is the "refreshing gap".

B. Orthogonal Learning Particle Swarm Optimizer

Orthogonal Learning PSO (OLPSO) [13] appropriates Orthogonal Experimental Design (OED) [29] to combine the personal and neighborhood experiences of each particle into a single exemplar. Using this method, which can be applied to any swarm topology, the "Oscillation" and "Two steps forward, one step back" problems are solved, while maintaining or improving performance over CLPSO, as the quality of the produced exemplars is higher [13].

Let $\mathbf{Pb}^i$ be the personal best location of the ith particle, and $\mathbf{Gb}^i$ be its neighborhood best location, two D-dimensional vectors. Optimally, the exemplar should contain the best search information from each of these two locations. For each dimension d, the goal is thus to find which of $\mathbf{Pb}^i_d$ or $\mathbf{Gb}^i_d$ should be the exemplar's value in that dimension. In other words, for a given dimension, a choice has to be made between the personal and neighborhood knowledge when guiding a particle. The brute force approach would be to evaluate the fitness for each of the 2^D combinations of values from $\mathbf{Pb}^i$ and $\mathbf{Gb}^i$. However, a better approach would be to use OED to predict the best combination using a representative sample of at most $2D$ combinations. The sample is built using an orthogonal array, while the best combination is determined by applying factor analysis [30] on the results of the experiment.

An orthogonal array with N factors and Q levels is denoted $L_M(Q^N)$, where M is the required size of the sample. In the context of building an exemplar from two D-dimensional vectors, the factors are the dimensions (1 to D), while the levels are the two possible values for that variable: personal best or neighborhood best. These levels are denoted respectively 0 and 1. Therefore, we are interested in orthogonal arrays of the form $L_M(2^D)$, where the sample size M is given by $M = 2^{\lceil \log_2(D+1) \rceil} \leq 2D$.

The following is an example of a suitable $L_8(2^5)$ array

$$L_8(2^5) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

Each row of $L_M(2^D)$ represents a D-dimensional sample. The dth variable of the sample should take its value from the personal best if the dth element of the row is 0, and from the neighborhood best if it is 1.

The procedure for constructing an $L_M(2^D)$ orthogonal array is given in Algorithm 2, adapted from [13], [31]. The algorithm will in general return an array with more than D columns. As any combination of columns of an orthogonal array is still an orthogonal array [13], it suffices to eliminate columns beyond the Dth.

Given the $M \leq 2D$ sample points defined by $L_M(2^D)$, and given the associated fitness values $f_1, \ldots, f_M$, the factor analysis predicts the best possible combination of levels; for each dimension, whether the exemplar variable value should come from the personal best or the neighborhood best. Define

$$z_{md0} = L_M(2^D)[m][d],$$

Algorithm 2 Construction of an $L_M(2^D)$ orthogonal array

```
1:  procedure $L_M(2^D)$
2:      $u \leftarrow \lceil \log_2(D+1) \rceil$
3:      $M \leftarrow 2^u$
4:      $C \leftarrow M - 1$
5:      $L \leftarrow$ an empty $M \times C$ array
6:      for $a = 1, \ldots, M$ do
7:          for $k = 1, \ldots, u$ do
8:              $b \leftarrow 2^{k-1}$
9:              $L[a][b] \leftarrow \left(\left\lceil \frac{a-1}{2^{u-k}} \right\rceil\right) \mod 2$
10:     for $a = 1, \ldots, M$ do
11:         for $k = 2, \ldots, u$ do
12:             $b \leftarrow 2^{k-1}$
13:             for $s = 1, \ldots, b-1$ do
14:                 $L[a][b+s] \leftarrow (L[a][s] + L[a][b]) \mod 2$
        return $L$
```

i.e., 0 if the value of the dth variable of the mth sample point was taken from level 0, and 1 if it was taken from level 1. Define $z_{md1} = 1 - z_{md0}$. The effect of the level q on the dth variable is denoted S_{dq} and is calculated as follows [13]:

$$S_{dq} = \frac{\sum_{m=1}^{M} f_m z_{mdq}}{\sum_{m=1}^{M} z_{mdq}}. \tag{6}$$

In other words, S_{dq} is the mean of the function values for the sample points where the dth variable was taken from level q. For instance, S_{d0} gives an approximation of the function value that should be expected if the dth variable is set to be taken from the personal best location of the particle. As such, for a minimization problem, variable d of the exemplar should be taken from the personal best location if $S_{d0} < S_{d1}$, and from the neighborhood best otherwise. The exemplar is constructed by applying this reasoning to each of the D dimensions. To make full use of the FEs required by Algorithm 3, and to avoid using a sub-par exemplar, the position obtained by factor analysis is also evaluated and compared in fitness to the sample combinations. If the constructed position is inferior to the best sample combination, the best sample combination is instead taken as the exemplar. The factor analysis and construction of the exemplar is summarized in Algorithm 3.

An exemplar is not constructed for every particle every generation, due to the extremely high cost of doing so. Rather, a new exemplar is constructed only if there has been no significant improvement to a particle's personal best location for a number of generations (see the refreshing gap in CLPSO, Sec. III-A). In addition, note that, similar to CLPSO, the exemplar is a list of levels, rather than an actual search space location. Every time the exemplar needs to be utilized, each level needs to be replaced by the corresponding variable value: from the personal best if the level is 0, and from the neighborhood best otherwise. This way, the exemplar can be reconstructed every iteration without any FE, and will naturally evolve as new best locations are found by the particle and its neighbors.

Algorithm 3 OLPSO exemplar construction

```
1:  procedure $\mathbf{Eol}^i$
2:      $L \leftarrow L_M(2^D)$
3:      $M \leftarrow L$'s number of rows
4:      $F \leftarrow$ an empty length $M$ array
5:      for $m = 1, \ldots, M$ do
6:          $X \leftarrow$ an empty length $D$ array
7:          for $d = 1, \ldots, D$ do
8:              if $L[m][d] = 0$ then
9:                  $X[d] \leftarrow \mathbf{Pb}^i[d]$
10:             else
11:                 $X[d] \leftarrow \mathbf{Gb}^i[d]$
12:         $F[m] \leftarrow f(X)$
13:     exemplar $\leftarrow$ an empty length $D$ array
14:     $X \leftarrow$ an empty length $D$ array
15:     for $d = 1, \ldots, D$ do
16:         $S \leftarrow 0$
17:         for $m = 1, \ldots, M$ do
18:             if $L[m][d] = 0$ then
19:                 $S \leftarrow S + F[m]$
20:             else
21:                 $S \leftarrow S - F[m]$
22:         if $S < 0$ then
23:             exemplar$[d] \leftarrow 0$
24:             $X[d] \leftarrow \mathbf{Pb}^i[d]$
25:         else
26:             exemplar$[d] \leftarrow 1$
27:             $X[d] \leftarrow \mathbf{Gb}^i[d]$
28:     if $f(X) > \min F$ then
29:         $i \leftarrow \operatorname{argmin} F$
30:         exemplar $\leftarrow$ row $i$ of $L$ (columns 1 to $D$)
    return exemplar
```

OLPSO uses this exemplar to update the velocity, rather than both the personal best and neighborhood best. Let $\mathbf{Eol}^{i,t}$ be the exemplar constructed from $\mathbf{Pb}^{i,t}$ and $\mathbf{Gb}^{i,t}$ at iteration t. The update rule in OLPSO is the following :

$$\mathbf{V}_d^{i,t} = w^t \mathbf{V}_d^{i,t} + c_1 \mathbf{r}_{1,d} (\mathbf{Eol}_d^{i,t} - \mathbf{X}_d^{i,t}) \tag{7}$$

C. Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients

1) TVAC: In the original PSO [10], the inertia factor is held constant throughout the generations. A strong inertia weight is desired early, as it fosters wandering of the particles and exploration of the search space [15]. However, a strong inertia is inadequate in the late stages of optimization, where wandering should be limited in order to promote convergence [15]. Recall that formula (1) uses a linearly decaying inertia weight to help achieve this. This begs the question of whether the cognitive and social parameters c_1 and c_2 should be made to vary similarly with time. [15] argues that a high cognitive parameter value c_1 leads to wandering, while a high social parameter value c_2 promotes convergence. Therefore, the cognitive parameter should be high in early generations and progressively get lower, while the social parameter should start low and increase with time. This promotes global search in the early stages while preventing premature convergence, and promotes convergence when nearing the maximal number of iterations. Time-Varying Acceleration Coefficients PSO (PSO-TVAC) [15] consists of updating the velocity with (1) while linearly varying the coefficients. The update rules are:

$$w^t = w_{\min} + (w_{\max} - w_{\min})\frac{\text{maxiter} - t}{\text{maxiter}} \tag{8}$$

$$c_1^t = c_{1\min} + (c_{1\max} - c_{1\min})\frac{\text{maxiter} - t}{\text{maxiter}} \tag{9}$$

$$c_2^t = c_{2\max} - (c_{2\max} - c_{2\min})\frac{\text{maxiter} - t}{\text{maxiter}} \tag{10}$$

$$\mathbf{V}_d^{i,t} = w^t\mathbf{V}_d^{i,t} + c_1^t\mathbf{r}_{1,d}(\mathbf{Pb}_d^{i,t} - \mathbf{X}_d^{i,t}) + c_2^t\mathbf{r}_{2,d}(\mathbf{Gb}_d^{i,t} - \mathbf{X}_d^{i,t}). \tag{11}$$

The maximal values $c_{1\max}$ and $c_{2\max}$ should be 2.5, while the minimal values $c_{1\min}$ and $c_{2\min}$ should be 0.5 [15]. PSO-TVAC provides increased performance when compared to original PSO [15].

2) HPSO: It has been observed that swarms tend to lack diversity in the later stages of optimization [15]. This causes superfluous FEs and premature convergence, and therefore hinders the quality of the final solution. [15] proposes a mutation operation to foster diversity and therefore enhance the global search capability of the swarm and prevent premature convergence. The mutation operation consists in attributing a random value to some dimension of the velocity vector. As is typical of evolutionary algorithms [32], the mutation operation is applied only in certain situations, or randomly. HPSO [15] applies this mutation operation when the velocity value in a given dimension is 0, that is when the particle has stopped moving in this dimension. Additionally, HPSO removes the inertia term in (1), leading to the following velocity update rule:

$$\mathbf{V}_d^{i,t+1} = c_1\mathbf{r}_{1,d}(\mathbf{Pb}_d^{i} - \mathbf{X}_d^{i,t}) + c_2\mathbf{r}_{2,d}(\mathbf{Gb}_d^{i,t} - \mathbf{X}_d^{i,t}) \tag{12}$$

If the updated velocity value $\mathbf{V}_d^{i,t+1}$ is 0 (to some precision), the mutation operation is applied. The velocity value is replaced using the following rules

$$v^t = \left(v_{\min} + (v_{\max} - v_{\min})\frac{\text{maxiter} - t}{\text{maxiter}}\right)\mathbf{Vmax}_d \tag{13}$$

$$\mathbf{V}_d^{i,t+1} = \mathbf{r}_{3,d}v^t \tag{14}$$

where v^t is the "reinitialization velocity", $\mathbf{r}_{3,d}$ is a number uniformly sampled from $[0, 1]$, $\mathbf{Vmax}_d$ is the maximal velocity, and $v_{\min}$ and $v_{\max}$ are constants, typically 0 and 1. Finally, with probability 0.5, $\mathbf{V}_d^{i,t+1}$ is multiplied by -1.

By granting a random velocity to the dimensions in which the particles have stagnated, a more diverse swarm is achieved. The reinitialization velocity is decreased linearly with time, once again to favor global search early and convergence in the later stages of the optimization procedure. [15] combines this HPSO update rule to TVAC to obtain a highly competitive PSO algorithm, HPSO-TVAC.

IV. Time Adaptive Dual Particle Swarm Optimization

Hyper-parameter selection is an important issue in PSO. Indeed, because non-convex optimization problems are usually computationally heavy to solve, it is often impractical to run an algorithm to completion multiple times in order to find the best parameter values. This issue is most salient for the choice of a topological structure, which can have a high impact on the final solution (as shown in the experimental sections of several articles [13], [12], [14]).

As mentioned in Sec. III-A, CLPSO emulates a local topology to a certain extent, even though every particle possesses full knowledge of the swarm. Therefore, CLPSO combines advantages from both global and local topologies, without the difficult task of choosing a topology. However, considering that any particle can be applied in the construction of CLPSO's exemplars with little to no quality requirement, these exemplars tend to be relatively poor in useful information. On the other hand, while OLPSO's exemplars are costly to generate, they are rich in information.

The idea behind Time Adaptive Dual Particle Swarm Optimization (TAD-PSO) is to combine the exemplar construction methods from OLPSO and CLPSO in order to produce information-rich exemplars while also removing the topological dependency of OLPSO. To limit the high cost of generating OLPSO-type exemplars, most noticeable when particles converge and new exemplars are required every few generations, TAD-PSO utilizes a decaying population size. To compensate for the loss of particles in the decaying population, a second population, based on a variant of HPSO-TVAC, is used. Finally, due to the velocity update of this new algorithm, which depends on a single information-rich exemplar, the previously mentioned "Oscillation" and "Two steps forward, one step back" problems are also prevented.

The decaying population is named the "main population" (MP), whereas the other population is referred to as the "auxiliary population" (AP). The remainder of this section describes in detail the behavior of both populations.

A. Main population

The purpose of the main population is exploration of the search space. To that end, the construction of an exemplar for the TAD-PSO main population is based on Algorithm 3, where $\mathbf{Gb}^i$ is replaced by $\mathbf{Ecl}^i$. The ith TAD-PSO particle's exemplar at iteration time t will be denoted $\mathbf{Etad}^{i,t}$. Due to Algorithm 3 already considering the personal best position of a particle, the learning probabilities ($\mathbf{Pc}_i$) from Algorithm 1 is set to a value (e.g. 42) that prohibits the utilization of the particle's personal best. Without this consideration, when constructing $\mathbf{Etad}^{i,t}$, some sample combinations (see Algorithm 3) would be the same, leading to wasted FEs and loss of information. The velocity update rule is as follows:

$$\mathbf{V}_d^{i,t} = w^t\mathbf{V}_d^{i,t} + c\,\mathbf{r}_d(\mathbf{Etad}_d^{i,t} - \mathbf{X}_d^{i,t}). \tag{15}$$

Note that, as the swarm starts converging around a minimum, the local topography of a multimodal function will start to resemble that of a unimodal function[1]. In that context, the information-rich exemplars lose their value, despite their high construction cost. Less costly velocity updates (updates better suited for optimizing unimodal functions) could be used in lieu of (15) when convergence is underway. This is the task of the AP, described in Sec. IV-B. For this reason, the MP is oblivious to the AP; the AP particles are not considered when creating a MP exemplar. Otherwise, particles from the MP would be disproportionally attracted to the current general position of the highly convergent AP, which would lessen its exploration abilities.

Because computing $\mathbf{Etad}^{i,t}$ requires up to $2D$ FEs, like OLPSO, TAD-PSO applies a refreshing gap of five generations. However, this mechanic favors an increase in superfluous FEs in the later stages of convergence, when improving becomes harder. To prevent this, the main population's size decreases linearly with time, until all particles become inactive. Once inactive, particles are neither chosen as exemplars nor updated, but remain as an exemplar if they were chosen prior to their deactivation. The particle chosen to be deactivated is the one exhibiting the worst personal best fitness value. The auxiliary population, described next, compensates for the diminishing number of particles.

B. Auxiliary population

HPSO-TVAC was reported by [13] to be the best PSO algorithm for optimizing unimodal functions. The auxiliary population follows HPSO-TVAC with two simple modifications. First, considering that the mutation operation from HPSO aims at increasing diversity and slowing convergence, it is not applied in the auxiliary population's velocity update rule. Indeed, diversity and global search capability is already the primary goal of the main population and would thus be redundant here. Secondly, the inertia component is added back to the HPSO velocity update rule, to avoid fast stagnation due to the removal of the mutation operation. In other words, the AP follows the PSO-TVAC update rules (11) with varying coefficients. While PSO-TVAC is ill-suited for multimodal problems, as it tends to get stuck in local minima [15], high quality solutions on unimodal problems are rapidly and consistently found even for its mutated version, HPSO-TVAC [15], [13], [23]. The AP is fully informed of the MP, so that it can exploit the interesting positions it finds. The goal of the auxiliary population is thus to compensate for the main population's weakness; convergence. PSO-TVAC also has the advantage of being simple to implement and having fewer hyper-parameters than HPSO-TVAC. The size of the auxiliary population remains constant.

Note that versions of TAD-PSO using local topologies were implemented and tested, but performed poorly, as expected, when compared to the algorithms presented in this paper. Due to space concerns, these results are omitted.

V. Experimentations and Comparisons

While the "No Free Lunch" theorem [33], [34] states that no optimization algorithm can perform better than all others on all possible functions, the quality of an optimization algorithm cannot be judged any other way than from its ability to solve a wide range of optimization problems [35]. With that in mind, TAD-PSO is tested on 19 benchmark functions, listed in Table I, and compared to the PSO variants on which it is based (see Table II).

A. Benchmark functions

The benchmark functions are listed in Table I. Tests were done using 10, 30 and 100 search space dimensions (i.e. $D = 10, 30, 100$). For tests in 10-dimensional search spaces, a maximum of 5×10^4 function evaluations were allowed. In 30 and 100-dimensional search spaces, this number was increased to 2×10^5 and 5×10^5 respectively. For rotated and shifted functions, M is a $D \times D$ orthonormal matrix and $\mathbf{c}$ is a D-dimensional vector, both randomly and uniformly generated once per simulation.

B. PSO Variants

The tested PSO variants are presented in Table II. Hyper-parameter values for each algorithm follow the recommendations of their respective paper. w is the inertia weight; c_1 and c_2 are respectively the cognitive and social parameters (simply c for the variants that use a single exemplar for the velocity update); G is the refreshing gap; R_d is the range of the search space in the dth dimension; $\mathbf{Vmax}_d$ is the maximal velocity in the dth dimension; s is the swarm size.

The first two variants are the basic PSO with linearly decreasing inertia weight, respectively using a global and local topology (ring). The third variant is CLPSO, as presented in Sec. III-A. The fourth and fifth variants are the OLPSO, respectively using a global and local topology (ring) (Sec. III-B). The sixth variant is HPSO-TVAC, presented in Sec III-C. Note that PSO-TVAC is omitted, as the algorithm alone performs poorly when compared to HPSO-TVAC [15]. The novel algorithm presented in this paper, TAD-PSO, is the final tested variant (Sec. IV). As every listed algorithm is stochastic, the reported results are averaged from 30 independent runs.

TABLE II

PSO variants to be compared

Algorithm	Parameters	Reference
PSO-G	$w = .9 \sim .4$, $c_1 = c_2 = 2$, $\text{Vmax}_d = R_d/5$, $s = 30, 40, 50$	[17]
PSO-L	$w = .9 \sim .4$, $c_1 = c_2 = 2$, $\text{Vmax}_d = R_d/5$, $s = 30, 40, 50$	[14]
CLPSO	$w = .9 \sim .4$, $c = 1.49445$, $G = 7$, $\text{Vmax}_d = R_d/5$, $s = 30, 40, 50$	[12]
HPSO-TVAC	$c_1 = 2.5 \sim .5$, $c_2 = .5 \sim 2.5$, $\text{Vmax}_d = R_d/5$, $s = 30, 40, 50$	[15]
OLPSO-G	$w = .9 \sim .4$, $c = 2$, $G = 5$, $\text{Vmax}_d = R_d/5$, $s = 30, 40, 50$	[13]
OLPSO-L	$w = .9 \sim .4$, $c = 2$, $G = 5$, $\text{Vmax}_d = R_d/5$, $s = 30, 40, 50$	[13]
TAD-PSO	MP: $w = .9 \sim .4$, $c = 2$, $G = 5$, $\text{Vmax}_d = R_d/5$, $s = 37, 75, 120$ AP: $c_1 = 2.5 \sim .5$, $c_2 = .5 \sim 2.5$, $\text{Vmax}_d = R_d/5$, $s = 13, 25, 40$	-

As TAD-PSO employs two populations, a swarm size of 40 was found to be simply too small for finding satisfactory

[1]We assume that the functions to be optimized are sufficiently well-behaved. For example, the function $f(x) = \left|x \sin\left(\frac{1}{x}\right)\right| + |x|$ (with $f(0)$ defined as 0) does not exhibit this desired behavior around $x = 0$, despite having a single global optimum.

TABLE I

THE 19 BENCHMARK FUNCTIONS USED FOR PERFORMANCE COMPARISON.

	Benchmark Problem	Search Range (R)	Global Opt. $\mathbf{x}$	$f_{\min}$	Name
Unimodal	$f_1(\mathbf{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-10, 10]^D$	$\{0\}^D$	0	Schwefel's P1.2 [23]
	$f_2(\mathbf{x}) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	$[-10, 10]^D$	$\{1\}^D$	0	Rosenbrock [36]*
	$f_3(\mathbf{x}) = \sum_{i=1}^{D} (10^6)^{(i-1)/(D-1)} x_i^2$	$[-100, 100]^D$	$\{0\}^D$	0	Elliptic [36]
	$f_4(\mathbf{x}) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	$\{0\}^D$	0	Sphere [13]
Multimodal	$f_5(\mathbf{x}) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$	$[-5.12, 5.12]^D$	$\{0\}^D$	0	Rastrigin [36]
	$f_6(\mathbf{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^D$	$\{0\}^D$	0	Ackley [36]
	$f_7(\mathbf{x}) = 418.982887272 \times D - \sum_{i=1}^{D} x_i \sin(\sqrt{\lvert x_i\rvert})$	$[-500, 500]^D$	$\{420.96\}^D$	0	Schwefel [23]
	$f_8(\mathbf{x}) = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i\rvert$	$[-10, 10]^D$	$\{0\}^D$	0	Alpine [23]
	$f_9(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	$\{0\}^D$	0	Griewank [36]
	$f_{10}(\mathbf{x}) = \frac{\pi}{D}\left(10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2[1 + 10\sin^2(\pi y_{i+1})] + (y_D - 1)^2\right) + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$	$[-50, 50]^D$	$\{0\}^D$	0	Generalized Penalized [23]
	Where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, x_i > a \\ 0, -a \leq x_i \leq a \\ k(-x_i - a)^m, x_i < -a \end{cases}$				
Rotated and Shifted	$f_{11}(\mathbf{x}) = 418.982887272 \times D - \sum_{i=1}^{D} z_i$, where $z_i = \begin{cases} y_i \sin\sqrt{\lvert y_i\rvert}, \lvert y_i\rvert \leq 500.0 \\ -.001(\lvert y_i\rvert - 500)^2, \text{otherwise} \end{cases}$ and $\mathbf{y} = \mathbf{y}' + \{420.96\}^D$, $\mathbf{y}' = M\left(\mathbf{x} - \{420.96\}^D\right)$	$[-500, 500]^D$	$\{420.96\}^D$	0	Rotated Schwefel [12]
	$f_{12}(\mathbf{x}) = \sum_{i=1}^{D-1}\left[100(y_{i+1} - y_i^2)^2 + (y_i - 1)^2\right]$, where $\mathbf{y} = M\mathbf{x}$	$[-10, 10]^D$	$\{0\}^D$	0	Rotated Rosenbrock [36]
	$f_{13}(\mathbf{x}) = \sum_{i=1}^{D}[y_i^2 - 10\cos(2\pi y_i) + 10]$, where $\mathbf{y} = M\mathbf{x}$	$[-5.12, 5.12]^D$	$\{0\}^D$	0	Rotated Rastrigin [36]
	$f_{14}(\mathbf{x}) = \sum_{i=1}^{D-1}\left[100(y_{i+1} - y_i^2)^2 + (y_i - 1)^2\right]$, where $\mathbf{y} = \mathbf{x} - \mathbf{c}$	$[-10, 10]^D$	$\{1\}^D + \mathbf{c}$	0	Shifted Rosenbrock [36]
	$f_{15}(\mathbf{x}) = \sum_{i=1}^{D}[y_i^2 - 10\cos(2\pi y_i) + 10]$, where $\mathbf{y} = \mathbf{x} - \mathbf{c}$	$[-5.12, 5.12]^D$	$\mathbf{c}$	0	Shifted Rastrigin [36]
	$f_{16}(\mathbf{x}) = \sum_{i=1}^{D}[y_i^2 - 10\cos(2\pi y_i) + 10]$, where $\mathbf{y} = M(\mathbf{x} - \mathbf{c})$	$[-5.12, 5.12]^D$	$\mathbf{c}$	0	Shifted Rotated Rastrigin [36]
	$f_{17}(\mathbf{x}) = 0.5 + \left\lvert\sum_{i=1}^{D} y_i^2 - D\right\rvert^{1/4} + \frac{1}{D}\left(0.5\sum_{i=1}^{D} y_i^2 + \sum_{i=1}^{D} y_i\right)$, where $\mathbf{y} = M(\mathbf{x} - \mathbf{c})$	$[-100, 100]^D$	$\{-1\}^D + \mathbf{c}$	0	Shifted Rotated HappyCat [36]
	$f_{18}(\mathbf{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1}(w_i - 1)^2\left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_D - 1)^2\left[1 + \sin^2(2\pi w_D)\right]$, where $w_i = 1 + \frac{y_i - 1}{4}$ and $\mathbf{y} = M(\mathbf{x} - \mathbf{c})$	$[-100, 100]^D$	$\{1\}^D + \mathbf{c}$	0	Shifted Rotated Levy [36]
	$f_{19}(\mathbf{x}) = 0.5 + \left\lvert\left(\sum_{i=1}^{D} y_i^2\right)^2 - \left(\sum_{i=1}^{D} y_i\right)^2\right\rvert^{1/4} + \frac{1}{D}\left(0.5\sum_{i=1}^{D} y_i^2 + \sum_{i=1}^{D} y_i\right)$, where $\mathbf{y} = M(\mathbf{x} - \mathbf{c})$	$[-100, 100]^D$	$\mathbf{c}$	0	Shifted Rotated HGBat [36]

*Rosenbrock function can be considered multimodal in high dimensions [36]

solutions. Furthermore, keeping the linear decay of the main population in mind, a larger than average population size was selected for the main population. We used a main population size of 37 for 10 dimensions, 75 for 30 dimensions, and 120 for 100 dimensions, with respective sizes of 13, 25 and 40 for the auxiliary population. In an effort not to skew the experimental results in favor of TAD-PSO, the TAD-PSO population sizes were settled upon using a benchmark problem other than those reported in this article. While no population size was suggested in the corresponding papers for 100-dimensional problems, [37] finds that bigger swarms are better suited for solving complex problems, such as high-dimensional ones. Therefore, the population sizes for all algorithms except TAD-PSO were increased to 50 for the 100-dimensional experiments, and reduced to 30 for 10 dimensions. Note that all PSO variants were also tested using the TAD-PSO population sizes, but this led to poor results for all but the TAD-PSO algorithm. For space concerns, these results are not presented in this paper.

C. Experimental results

Table III presents the average performance over 30 runs of each algorithm on each benchmark function, in 30 dimensions. TAD-PSO ranks first in most cases, and never falls below third place, on average always returning a competitive solution. Table IV presents similar results in 100 dimensions, where TAD-PSO returns the best solutions for most functions, although it does fall below third rank once. For space considerations, the results in 10 dimensions are not presented in this paper, but are available at `http://graal.ift.ulaval.ca/public/10D.pdf`. Overall, TAD-PSO has a clear edge on the other algorithms. This edge is at its best on the rotated and shifted versions of the benchmark functions, which seems to indicate that TAD-PSO has better chance to perform well on real life problems.

Also, observe that OLPSO-G and OLPSO-L appear to have complementary performances. Indeed, OLPSO-G generally outranks OLPSO-L on unimodal and unrotated functions, while the opposite seems to be true for multimodal and rotated functions, which are more challenging to optimize. This is related to the fact that OLPSO solutions depend on the chosen topology. In contrast, TAD-PSO was designed not to depend on topology and performs uniformly well on both unimodal and multimodal functions.

As suggested in [38], a two-step procedure is used to compare TAD-PSO with the six other algorithms. First, Friedman's test is used to rank the algorithms amongst each other and test the null hypothesis that the algorithms are all equivalent. In every case, the null hypothesis was rejected ($p < 0.05$). The second step consists of a post-hoc test, the null hypothesis of which is that the mean results of the control method (TAD-PSO) is equal to the mean results of each of the other PSO variants (i.e. six tests in total).

TABLE III

BENCHMARK RESULT COMPARISONS OF PSOS ON 19 GLOBAL OPTIMIZATION PROBLEMS (30 DIMENSIONS)

Function		PSO-G	PSO-L	CLPSO	HPSO-TVAC	OLPSO-G	OLPSO-L	TAD-PSO
Schwefel's P1.2	Mean	1.012	2.991	8.123	**9.736e-10**	3.250e-3	0.823	1.032e-4
f_1	SD	0.353	1.055	1.946	**1.662e-9**	5.562e-3	0.684	1.841e-4
	Rank	5	6	7	**1**	3	4	2
Rosenbrock	Mean	53.055	55.865	14.536	6.509	21.610	3.640	**0.021**
f_2	SD	33.507	23.640	9.79	5.125	30.465	6.420	**0.021**
	Rank	6	7	4	3	5	2	**1**
Elliptic	Mean	40.780	126.296	6.671e-12	1.455e-19	**1.068e-69**	3.175e-33	1.472e-62
f_3	SD	22.676	46.415	5.046e-12	6.345e-20	**2.742e-69**	1.093e-32	3.455e-62
	Rank	6	7	5	4	**1**	3	2
Sphere	Mean	0.920	8.402	2.419e-12	8.328e-21	**6.990e-72**	1.624e-35	2.128e-63
f_4	SD	0.641	3.522	1.132e-12	3.396e-21	**1.741e-71**	5.162e-35	8.885e-63
	Rank	6	7	5	4	**1**	3	2
Rastrigin	Mean	48.426	19.707	9.471e-5	0.365	3.051	1.895e-15	**0.0**
f_5	SD	10.620	4.353	1.128e-4	0.748	1.703	1.020e-14	**0.0**
	Rank	7	6	3	4	5	2	**1**
Ackley	Mean	3.126	1.951	5.050e-7	6.522e-11	5.566e-15	6.869e-15	**4.974e-15**
f_6	SD	0.576	0.450	1.345e-7	1.157e-11	1.760e-15	8.862e-16	**1.740e-15**
	Rank	7	6	5	4	2	3	**1**
Schwefel	Mean	5631.682	5320.552	2.581e-10	1396.914	435.606	2.583e-4	**1.213e-12**
f_7	SD	652.454	644.091	1.714e-10	258.064	265.949	2.156e-4	**1.182e-12**
	Rank	7	6	2	5	4	3	**1**
Alpine	Mean	1.742	0.597	6.878e-4	2.784e-10	**2.598e-15**	7.979e-8	5.727e-12
f_8	SD	0.832	0.342	2.029e-4	1.597e-10	**2.202e-15**	2.862e-7	3.083e-11
	Rank	7	6	5	3	**1**	4	2
Griewank	Mean	0.701	1.070	6.816e-9	0.015	4.503e-3	**0.0**	**0.0**
f_9	SD	0.159	0.039	6.545e-9	0.019	0.010	**0.0**	**0.0**
	Rank	6	7	3	5	4	**1**	**1**
Generalized Penalized	Mean	3.704e-31	**0.0**	**0.0**	5.664e-31	1.215e-32	**0.0**	**0.0**
f_{10}	SD	6.389e-31	**0.0**	**0.0**	9.110e-31	4.544e-32	**0.0**	**0.0**
	Rank	6	**1**	**1**	7	5	**1**	**1**
Rotated Schwefel	Mean	2205.323	3676.135	3208.419	5385.639	688.816	433.874	**159.444**
f_{11}	SD	647.201	976.877	384.556	662.571	441.160	486.607	**235.615**
	Rank	4	6	5	7	3	2	**1**
Rotated Rosenbrock	Mean	74.403	51.407	44.847	**11.653**	28.366	22.758	20.067
f_{12}	SD	57.820	18.033	5.630	**8.875**	10.1663	11.639	3.170
	Rank	7	6	5	**1**	4	3	2
Rotated Rastrigin	Mean	64.094	51.655	78.979	25.814	38.568	44.740	**5.845**
f_{13}	SD	19.643	12.082	9.006	6.936	8.868	10.239	**3.242**
	Rank	6	5	7	2	3	4	**1**
Shifted Rosenbrock	Mean	63.512	101.083	56.093	4.812	11.601	1.462	**0.038**
f_{14}	SD	61.965	87.208	20.043	11.892	23.620	3.868	**0.069**
	Rank	6	7	5	3	4	2	**1**
Shifted Rastrigin	Mean	100.851	63.089	3.273	0.929	6.832	0.265	**0.0**
f_{15}	SD	21.350	15.242	0.985	1.333	2.447	0.440	**0.0**
	Rank	7	6	4	3	5	2	**1**
Shifted Rotated Rastrigin	Mean	166.826	109.473	75.110	119.592	48.486	53.560	**6.314**
f_{16}	SD	32.494	22.604	7.638	25.990	11.150	12.469	**1.720**
	Rank	7	5	4	6	2	3	**1**
Shifted Rotated Happy Cat	Mean	0.557	0.361	0.150	0.487	0.212	0.199	0.128
f_{17}	SD	0.151	0.081	5.213e-2	0.207	5.558e-2	4.802e-2	1.756e-2
	Rank	7	5	2	6	4	3	1
Shifted Rotated Levy	Mean	2727.888	304.904	95.196	2057.611	2.466	0.517	3.330e-2
f_{18}	SD	1707.314	516.719	79.863	977.051	9.778	1.603	9.012e-2
	Rank	7	5	4	6	3	2	1
Shifted Rotated HGBat	Mean	0.493	1.496	0.209	0.476	0.408	0.323	0.385
f_{19}	SD	0.258	2.164	7.456e-2	0.247	0.142	3.339	3.762e-2
	Rank	6	7	1	5	4	2	3
Ave. rank		6.316	6.000	4.053	4.158	3.316	2.579	**1.369**
Final rank		7	6	4	5	3	2	**1**
Algorithms		PSO-G	PSO-L	CLPSO	HPSO-TVAC	OLPSO-G	OLPSO-L	TAD-PSO

Note that changing the number of dimensions changes the underlying functions in a profound enough way [39], [13] that it significantly alters the optimization performance of the various algorithms [12], [23], [15], [20]. With that in mind, functions in the different dimensions can be considered to be distinct benchmark functions. As such, the statistical tests can be applied to the concatenation of the results in the different dimensions. This triples the data points from 19 to 57, greatly increasing the power of the statistical tests. As can be seen in Table V, the null hypothesis, under these conditions, is rejected every time, meaning that TAD-PSO overall performs significantly better than all the other algorithms.

TABLE IV

BENCHMARK RESULT COMPARISONS OF PSOS ON 19 GLOBAL OPTIMIZATION PROBLEMS (100 DIMENSIONS)

Function		PSO-G	PSO-L	CLPSO	HPSO-TVAC	OLPSO-G	OLPSO-L	TAD-PSO
Schwefel's P1.2 f_1	Mean	118.532	62.458	1051.594	**0.157**	244.837	475.505	91.964
	SD	21.095	13.763	86.545	**0.097**	65.929	128.818	31.589
	Rank	4	2	7	**1**	5	6	3
Rosenbrock f_2	Mean	633.846	756.390	191.807	113.688	46.267	3.213	**0.166**
	SD	137.158	108.231	52.727	38.956	54.048	12.352	**0.174**
	Rank	6	7	5	4	3	2	**1**
Elliptic f_3	Mean	976.225	2036.556	6.251e-7	8.849e-17	**2.442e-44**	2.489e-19	4.282e-37
	SD	229.203	296.693	1.561e-7	2.916e-17	**9.298e-44**	3.877e-19	1.245e-36
	Rank	6	7	5	4	1	3	2
Sphere f_4	Mean	78.290	330.663	2.335e-7	2.413e-17	**6.784e-47**	1.751e-21	1.861e-38
	SD	19.391	49.578	5.860e-8	8.693e-18	**1.715e-46**	3.848e-21	4.676e-38
	Rank	6	7	5	4	**1**	3	2
Rastrigin f_5	Mean	224.640	126.230	142.416	11.542	13.830	0.133	**0.0**
	SD	33.143	15.243	9.993	8.081	6.183	0.425	**0.0**
	Rank	7	5	6	3	4	2	**1**
Ackley f_6	Mean	7.248	4.633	1.002e-4	1.976e-9	**9.711e-15**	1.533e-9	6.241e-14
	SD	0.668	0.240	9.679e-6	4.340e-10	**3.299e-15**	8.286e-10	1.013e-13
	Rank	7	6	5	4	**1**	3	2
Schwefel f_7	Mean	23174.825	22936.395	771.640	4267.070	502.047	51.323	**9.944e-12**
	SD	1827.283	1380.293	377.452	598.737	226.545	58.690	**6.366e-12**
	Rank	7	6	4	5	3	2	**1**
Alpine f_8	Mean	24.499	9.075	0.085	3.620e-8	**1.130e-14**	8.931e-4	1.064e-4
	SD	4.563	2.548	0.025	2.985e-8	**8.935e-15**	7.004e-4	6.641e-5
	Rank	7	6	5	2	**1**	4	3
Griewank f_9	Mean	1.821	3.955	4.495e-7	0.011	0.001	**0.0**	**0.0**
	SD	0.173	0.489	2.541e-7	0.013	0.007	**0.0**	**0.0**
	Rank	6	7	3	5	4	**1**	**1**
Generalized Penalized f_{10}	Mean	0.996	5.597e-32	**0.0**	5.387e-31	6.219e-32	**0.0**	**0.0**
	SD	2.540	3.014e-31	**0.0**	7.072e-31	3.021e-31	**0.0**	**0.0**
	Rank	7	4	**1**	6	5	**1**	**1**
Rotated Schwefel f_{11}	Mean	2205.323	3676.135	18563.120	19887.658	1040.930	**339.398**	575.033
	SD	647.201	976.877	616.851	3073.267	769.272	**318.779**	627.482
	Rank	4	5	6	7	3	**1**	2
Rotated Rosenbrock f_{12}	Mean	662.811	803.915	97.184	**81.361**	122.521	169.373	131.502
	SD	147.654	150.490	2.142	**45.348**	26.917	47.979	52.283
	Rank	6	7	2	**1**	3	5	4
Rotated Rastrigin f_{13}	Mean	237.936	307.793	430.041	71.754	104.504	155.127	**11.174**
	SD	43.782	47.917	22.097	13.534	21.374	28.639	**3.118**
	Rank	5	6	7	2	3	4	**1**
Shifted Rosenbrock f_{14}	Mean	633.873	3512.113	248.419	107.850	35.497	7.813	**1.196**
	SD	343.957	1159.389	64.070	47.962	54.836	20.464	**1.226**
	Rank	6	7	5	4	3	2	**1**
Shifted Rastrigin f_{15}	Mean	741.799	521.107	180.615	29.116	37.444	1.459	**0.531**
	SD	102.652	44.060	12.429	25.712	8.778	0.986	**0.559**
	Rank	7	6	5	3	4	2	**1**
Shifted Rotated Rastrigin f_{16}	Mean	795.473	731.826	468.640	304.366	160.613	179.744	**16.541**
	SD	127.527	71.654	25.879	49.098	27.785	33.770	**4.211**
	Rank	7	6	5	4	2	3	**1**
Shifted Rotated Happy Cat f_{17}	Mean	0.619	5.244	0.408	0.569	0.367	0.269	**0.227**
	SD	0.108	0.564	5.490e-2	0.161	5.694e-2	4.767e-2	**2.640e-2**
	Rank	6	7	4	5	3	2	**1**
Shifted Rotated Levy f_{18}	Mean	26982.031	23197.826	12068.214	14265.778	295.680	528.595	**9.769e-2**
	SD	4538.699	4538.699	2473.329	1697.539	366.699	611.434	**0.204**
	Rank	7	6	4	5	2	3	**1**
Shifted Rotated HGBat f_{19}	Mean	0.655	289.460	**0.277**	0.506	0.484	0.402	0.457
	SD	0.255	48.718	**0.103**	0.239	0.129	3.134e-2	1.525e-2
	Rank	6	7	**1**	5	4	2	3
Ave. rank		6.158	6.000	4.474	3.895	2.895	2.684	**1.685**
Final rank		7	6	5	4	3	2	**1**
Algorithms		PSO-G	PSO-L	CLPSO	HPSO-TVAC	OLPSO-G	OLPSO-L	TAD-PSO

TABLE V

AVERAGE RANKING FROM FRIEDMAN'S TEST AND HOLM NULL HYPOTHESIS TESTING

		PSO-G	PSO-L	CLPSO	HPSO-TVAC	OLPSO-G	OLPSO-L	TAD-PSO
10D& 30D& 100D	Rank	6.187	5.904	4.009	3.947	3.263	3.009	**1.658**
	h	**1**	**1**	**1**	**1**	**1**	**1**	-
10D	Rank	6.158	5.763	4.316	3.526	3.395	3.184	**1.658**
	h	**1**	**1**	**1**	**1**	**1**	**1**	-
30D	Rank	6.263	5.895	4.053	4.158	3.263	2.921	**1.447**
	h	**1**	**1**	**1**	**1**	**1**	**1**	-
100D	Rank	6.211	6.053	4.579	3.368	3.000	2.921	**1.868**
	h	**1**	**1**	**1**	0	0	0	-

*Reject null hypothesis ($h = 1$), accept null hypothesis ($h = 0$).

VI. CONCLUSION

This paper presented a novel algorithm, Time Adaptive Dual Particle Swarm Optimization (TAD-PSO), which makes use of key concepts from OLPSO, CLPSO and HPSO-TVAC. TAD-PSO employs two populations, with information being shared only from the main to the auxiliary population. The main population takes inspiration from CLPSO and OLPSO, using orthogonal learning to find the most interesting directions to explore in every dimension. The auxiliary population is a PSO-TVAC swarm, chosen for its high performance on unimodal problems. The decaying main population size achieves the trade-off between exploration of the search space and exploitation of the interesting local minima.

Comprehensive experimental tests were conducted on 19 benchmark problems of three types: unimodal, multimodal, and shifted/rotated functions. The results show that TAD-PSO achieves better performance than any of the PSO algorithms it is based on, while still being intuitive and easy to implement.

Future works will focus on developing a heuristic for handling the decay of the main population, and will compare TAD-PSO to a broader range of optimization algorithms.

References

[1] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[2] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, and J.-J. Li, "Distributed evolutionary algorithms and their models: A survey of the state-of-the-art," *Applied Soft Computing*, vol. 34, pp. 286–300, 2015.

[3] Y. Yuan, H. Xu, B. Wang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2016.

[4] X.-S. Yang, "Bat algorithm for multi-objective optimisation," *International Journal of Bio-Inspired Computation*, vol. 3, no. 5, pp. 267–274, 2011.

[5] G.-G. Wang, A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "A new hybrid method based on krill herd and cuckoo search for global optimisation tasks," *International Journal of Bio-Inspired Computation*, vol. 8, no. 5, pp. 286–299, 2016.

[6] X.-S. Yang and S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and Applications*, vol. 24, no. 1, pp. 169–174, 2014.

[7] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015.

[8] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[9] S.-C. Chu, P.-W. Tsai, and J.-S. Pan, "Cat swarm optimization," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2006, pp. 854–858.

[10] R. C. Eberhart, J. Kennedy *et al.*, "A new optimizer using particle swarm theory," in *Proceedings of the sixth international symposium on micro machine and human science*, vol. 1. New York, NY, 1995, pp. 39–43.

[11] K.-L. Du and M. Swamy, "Particle swarm optimization," in *Search and Optimization by Metaheuristics*. Springer, 2016, pp. 153–173.

[12] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE transactions on evolutionary computation*, vol. 10, no. 3, pp. 281–295, 2006.

[13] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.

[14] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," *Congress on evolutionary computation: proceedings*, 2002.

[15] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on evolutionary computation*, vol. 8, no. 3, pp. 240–255, 2004.

[16] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.

[17] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*. IEEE, 1998, pp. 69–73.

[18] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999.

[19] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 3. IEEE, 1999.

[20] S. Helwig, J. Branke, and S. Mostaghim, "Experimental analysis of bound handling techniques in particle swarm optimization," *IEEE Transactions on Evolutionary computation*, vol. 17, no. 2, pp. 259–271, 2013.

[21] J. Barrera, O. Álvarez Bajo, J. J. Flores, and C. A. Coello Coello, "Limiting the velocity in the particle swarm optimization algorithm," *Computación y Sistemas*, vol. 20, no. 4, 2016.

[22] M. A. M. De Oca, T. Stutzle, M. Birattari, and M. Dorigo, "Frankenstein's pso: a composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.

[23] Q. Qin, S. Cheng, Q. Zhang, Y. Wei, and Y. Shi, "Multiple strategies based orthogonal design particle swarm optimizer for numerical optimization," *Computers & Operations Research*, vol. 60, pp. 91–110, 2015.

[24] M. R. Tanweer, S. Suresh, and N. Sundararajan, "Self regulating particle swarm optimization algorithm," *Information Sciences*, vol. 294, pp. 182–202, 2015.

[25] Z. Beheshti and S. M. Shamsuddin, "Non-parametric particle swarm optimization for global optimization," *Applied Soft Computing*, vol. 28, pp. 345–359, 2015.

[26] Q. Ni and J. Deng, "A new logistic dynamic particle swarm optimization algorithm based on random topology," *The Scientific World Journal*, vol. 2013, 2013.

[27] K. E. Parsopoulos and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Transactions on evolutionary computation*, vol. 8, no. 3, pp. 211–224, 2004.

[28] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE transactions on evolutionary computation*, vol. 8, no. 3, pp. 225–239, 2004.

[29] D. C. Montgomery, *Design and analysis of experiments*. John Wiley & Sons, 2008.

[30] K.-T. Fang and Y. Wang, *Number-theoretic methods in statistics*. CRC Press, 1993, vol. 51.

[31] J. Kennedy, "Stereotyping: improving particle swarm performance with cluster analysis stereotyping: improving particle swarm performance with cluster analysis," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 2, 2000, pp. 1507–1512.

[32] T. Back, *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford university press, 1996.

[33] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[34] K.-L. Du and M. Swamy, "Introduction," in *Search and Optimization by Metaheuristics*. Springer, 2016, pp. 1–28.

[35] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE transactions on evolutionary computation*, vol. 8, no. 3, pp. 204–210, 2004.

[36] N. Awad, M. Ali, J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2017 special session and competition on single objective real-parameter numerical optimization," 2016.

[37] D. Chen and C. Zhao, "Particle swarm optimization with adaptive population size and its application," *Applied Soft Computing*, vol. 9, no. 1, pp. 39–48, 2009.

[38] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine learning research*, vol. 7, no. Jan, pp. 1–30, 2006.

[39] D. L. Donoho *et al.*, "High-dimensional data analysis: The curses and blessings of dimensionality," *AMS Math Challenges Lecture*, vol. 1, p. 32, 2000.