Cooperative Micro–Particle Swarm Optimization

Konstantinos E. Parsopoulos
Department of Mathematics
University of Patras
GR–26110 Patras, Greece
kostasp@math.upatras.gr

ABSTRACT

Cooperative approaches have proved to be very useful in evolutionary computation due to their ability to solve efficiently high-dimensional complex problems through the cooperation of low–dimensional subpopulations. On the other hand, Micro–evolutionary approaches employ very small populations of just a few individuals to provide solutions rapidly. However, the small population size renders them prone to search stagnation. This paper introduces Cooperative Micro–Particle Swarm Optimization, which employs cooperative low–dimensional and low–cardinality subswarms to concurrently adapt different subcomponents of high–dimensional optimization problems. The algorithm is applied on high-dimensional instances of five widely used test problems with very promising results. Comparisons with the standard Particle Swarm Optimization algorithm are also reported and discussed.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Global optimization,Unconstrained optimization; G.3 [**Probability and Statistics**]: Probabilistic algorithms

General Terms

Algorithms, Performance, Experimentation

Keywords

Particle Swarm Optimization, Cooperative, Micro–Evolutionary Algorithms, Swarm Intelligence

1. INTRODUCTION

Evolutionary algorithms have proved to be a very useful tool in modern applications where increasingly complex optimization problems are involved. However, dimensionality of these problems often becomes prohibitive for their efficient treatment with any stochastic optimization algorithm. This problem is also known as the *curse of dimensionality*, and it decreases the probability of finding a solution under reasonable time constraints.

The aforementioned deficiency has been addressed with the introduction of Cooperative Evolutionary Algorithms (CEAs) [18]. These approaches break the original problem into subcomponents of lower dimensionality and apply a separate subpopulation to evolve each one. Subpopulations can communicate and combine information for the evaluation of their individuals with the original (high–dimensional) objective function.

This approach has been extensively studied for the case of Genetic Algorithms (GAs) [3, 17, 18, 19]. Extensive experimentation with GA–based approaches, such as the Cooperative Coevolutionary Genetic Algorithm (CCGA) [17], revealed also several deficiencies of cooperative schemes, such as the deteriorating performance in problems with correlated coordinate directions and the introduction of new local minima. This has triggered the development of performance–enhancing techniques such as the surrogate–assisted version of CCGA [13]. Cooperative approaches were also introduced for Evolution Strategies [20] and Particle Swarm Optimization (PSO) [7, 23], verifying the ongoing interest of the scientific community.

Another interesting class of algorithms consists of Micro–Evolutionary Algorithms (Micro–EAs), which are instances of typical evolutionary algorithms characterized by small population size and simple fitness functions. Micro–Genetic Algorithms (Micro–GAs), also called Tiny–GAs, have been investigated in demanding applications [11]. Recently, a Micro–PSO (μPSO) approach was proposed for tackling demanding optimization problems [8].

Search stagnation is identified as the main drawback of Micro–approaches and can be attributed to the small population size, which limits their exploration capability. Indeed, small number of individuals results in rapid collapse of the population on the best detected solution, prohibiting the algorithm from further probing of the search space. Hence, efficiency is limited especially in complex high–dimensional problems.

This deficiency is usually addressed by combining Micro–EAs with diversity–preserving schemes. Such schemes provide the algorithm with the ability to retain population diversity, thereby increasing its exploration capability. Additionally, multiple restarts combined with techniques that prevent convergence to the same solution can be used. For example, μPSO [8] is equipped with a repulsion technique based on the approach of Parsopoulos and Vrahatis [16].

GEC'09, June 12–14, 2009, Shanghai, China.

In this paper, a Cooperative Micro–Particle Swarm Optimization (COMPSO) is introduced. To the best of the author's knowledge, this is the first approach that combines the efficiency of cooperative algorithms on high–dimensional problems with the flexibility and low computational requirements of Micro–PSO. Thus, a high–dimensional problem is divided to low–dimensional subcomponents. Each subcomponent is tackled with a small low–dimensional subswarm, while the evaluation of each particle is based on an information–sharing scheme among subswarms. The algorithm is applied on high–dimensional instances of five widely used test problems and compared, in terms of the best solution value after a prespecified number of iterations, with the standard PSO.

The rest of the paper is organized as follows: Section 2 describes the PSO algorithm, while Section 3 presents the proposed COMPSO approach. Experimental results are reported in Section 4 and the paper concludes in Section 5.

2. PARTICLE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) was proposed in 1995 by Eberhart and Kennedy [5, 9] as a stochastic population–based algorithm for continuous optimization problems. Collective behavior in decentralized systems in nature constituted the main inspiration source behind its development. Bird flocks, fish schools, and animal herds, as well as human groups with social hierarchy, share properties that can be modeled with simple direct interactions among their individuals, giving rise to the field of *Swarm Intelligence* (SI).

Memory and information sharing possess a dominant role in SI models. Five fundamental principals have been identified in SI systems [6, 12]:

(1) *Proximity*: The swarm shall be able to perform simple space and time computations.

(2) *Quality*: The swarm shall be a able to respond to quality factors in the environment.

(3) *Diverse response*: The swarm shall not commit its activities along excessively narrow channels.

(4) *Stability*: The swarm shall not change its behavior under slight changes of the environment.

(5) *Adaptability*: The swarm shall be able to change its behavior whenever needed.

PSO adheres these rules, hence it is categorized as an SI algorithm.

In PSO's context, the population is called *swarm*, while its individuals are called *particles*. The particles are initialized randomly in the search space and they move iteratively with a stochastically adapted velocity (position shift). Also, each particle retains a memory of the best position it has ever encountered and shares this information with a set of other particles, called its *neighborhood*.

In the *global* PSO variant, the best position ever attained by any individual is communicated to all other particles at each iteration, i.e., the whole swarm is considered as the neighborhood of each particle. On the other hand, in the *local* variant each particle is assigned a neighborhood strictly smaller than the swarm (usually consisting of just a few particles). In this case, the best position ever attained by the particles that comprise a neighborhood is communicated among them [10].

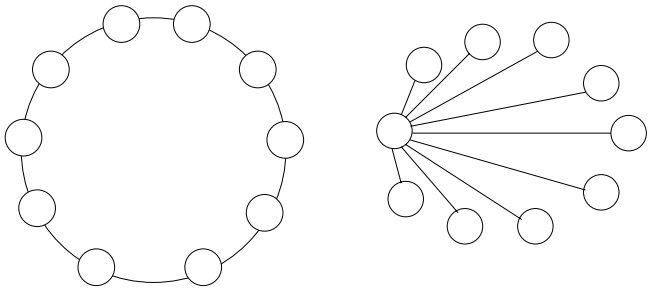

Figure 1: The ring (left) and star (right) neighborhood topology of PSO.

Neighboring particles are determined based on their indices rather than their actual distance in search space, according to their *neighborhood topology*. The simplest topologies are the *ring* and the *star* topology. In the first one, depicted in the left part of Fig. 1, particles are assumed to lie on a ring, each one having two immediate neighbors, namely the particles with its neighboring indices. The second one is depicted in the right part of Fig. 1, where each particle shares information only with the best one, i.e., the one with the overall best position. Apparently, star topology corresponds to the global PSO variant, while ring topology defines local variants.

To put it more formally, let:

$$f : S \subset \mathbb{R}^n \to \mathbb{R},$$

be the n–dimensional objective function under consideration. Then, a swarm consisting of N particles is defined as a set:

$$\mathbb{S} = \{x_1, x_2, \ldots, x_N\},$$

where the i–th particle is an n–dimensional vector:

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top \in S.$$

The velocity of this particle is also an n–dimensional vector:

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top.$$

The best previous position encountered by the i–th particle in S is denoted as:

$$p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top \in S.$$

If r is the neighborhood's radius, then the neighborhood of x_i under the ring topology is defined as the set:

$$\{x_{i-r}, x_{i-r+1}, \ldots, x_i, \ldots, x_{i+r-1}, x_{i+r}\},$$

where x_1 is considered as the immediate neighbor next to x_N.

Assume g_i to be the index of the particle that attained the best previous position among all particles in the neighborhood of x_i, and let t be the iteration counter. Then, the swarm is manipulated by the following equations [2]:

$$\begin{aligned} v_{ij}(t+1) &= \chi\Big[v_{ij}(t) + c_1 R_1\big(p_{ij}(t) - x_{ij}(t)\big) + \\ &\quad c_2 R_2\big(p_{g_i,j}(t) - x_{ij}(t)\big)\Big], \qquad (1) \\ x_{ij}(t+1) &= x_{ij}(t) + v_{ij}(t+1), \qquad (2) \end{aligned}$$

$i = 1, 2, \ldots, N$; $j = 1, 2, \ldots, n$; where χ is the *constriction coefficient*; c_1 and c_2 are the *cognitive* and *social* parameter, respectively; and R_1, R_2, are random variables uniformly distributed in $[0, 1]$.

The best positions are updated as follows:

$$p_i(t+1) = \begin{cases} x_i(t+1), & \text{if } f(x_i(t+1)) < f(p_i(t)), \\ p_i(t), & \text{otherwise.} \end{cases} \quad (3)$$

The constriction coefficient is a mechanism for controlling the magnitude of velocities. Based on the stability analysis of PSO due to Clerc and Kennedy [2], it is derived analytically through the formula:

$$\chi = \frac{2\kappa}{\left|2 - \phi - \sqrt{\phi^2 - 4\phi}\right|}, \quad (4)$$

where $\phi = c_1 + c_2$. Values received for $\phi > 4$ and $\kappa = 1$ are considered to be the most prominent settings of χ due to their good average performance [2]. A thorough theoretical analysis of the derivation of Eq. (4) can be found in [2, 22].

Initialization of swarm and velocities is usually performed randomly and uniformly in the search space, although more sophisticated techniques can enhance the overall performance of PSO [14, 15].

3. COOPERATIVE MICRO–PARTICLE SWARM OPTIMIZATION

Micro–PSO works in direct analogy to the standard PSO described in the previous section. The only difference is the swarm size, N, which is considered to be rather small. Typically, it is required that $N \leqslant 10$, with $N = 5$ being a common choice. Similarly to the rest Micro-EAs, the performance of small swarms depends heavily on problem dimension, n. Micro–PSO is expected to converge quite rapidly due to its small number of particles. Thus, the difficulty in detecting the global minimizer is related to the ratio,

$$D = \frac{n}{N}.$$

Small values of D (less than 1) imply swarm sizes larger than problem dimension. On the other hand, values higher than 1 correspond to dimensions higher than swarm size. Empirical evidence for evolutionary algorithms suggest that in most cases the problem becomes harder as D increases. According to this observation, Micro–PSO is expected to be a promising approach only in low–dimensional problems.

The problem of search stagnation and premature convergence in Micro–EAs has been tackled by introducing multi-start techniques that prevent the algorithm from converging to the same solution after restart. Huang and Mohan [8] equipped their μPSO approach with a scheme that repels the swarm away from any point belonging to a black–list of already detected solutions. Their technique has many in common with typical Tabu Search approaches, as well as with the repulsion technique of Parsopoulos and Vrahatis [16]. Nevertheless, the small amount of reported results, mostly for test problems with dimension up to $n = 500$, has proved to be inadequate to establish Micro–PSO approaches as a promising alternative.

In contrast to Micro–PSO, Cooperative PSO (CPSO) approaches have been proposed as a means for addressing high–dimensional problems efficiently [7, 23]. In CPSO, the original n–dimensional problem is divided to low–dimensional subcomponents, each one evolved with a separate subswarm.

Let $n_1, n_2, \ldots, n_K$, be K positive integers such that:

$$n = \sum_{k=1}^{K} n_k,$$

where n is the dimension of the original problem. Then, instead of using a single n–dimensional swarm, $\mathbb{S}$, as in standard PSO, K subswarms, $\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_K$, with dimensions, $n_1, n_2, \ldots, n_K$, and sizes, $N_1, N_2, \ldots, N_K$, respectively, are used. Thus, each subswarm optimizes a specific subcomponent of strictly smaller dimension than the original problem.

The update of each subswarm is exactly the same with the standard PSO described in Section 2. However, an issue arises with the evaluation of particles, since the objective function is n–dimensional while their dimension is strictly smaller than n in all subswarms. The problem is addressed by using an n–dimensional buffer vector, also called *context vector* [23]:

$$P = (P_1, P_2, \ldots, P_n)^\top,$$

where each subswarm deposits its contribution. Thus, if:

$$z^{[k]} = \left(z_1^{[k]}, z_2^{[k]}, \ldots, z_{n_k}^{[k]}\right)^\top,$$

is a contributed n_k–dimensional vector by the k–th subswarm, $\mathbb{S}_k$, $k = 1, 2, \ldots, K$, the buffer vector is defined as:

$$P = \Big(\underbrace{z_1^{[1]}, \ldots, z_{n_1}^{[1]}}_{\text{contribution of } \mathbb{S}_1}, \underbrace{z_1^{[2]}, \ldots, z_{n_2}^{[2]}}_{\text{contribution of } \mathbb{S}_2}, \ldots, \underbrace{z_1^{[K]}, \ldots, z_{n_K}^{[K]}}_{\text{contribution of } \mathbb{S}_K} \Big)^\top.$$

Then, the i–th particle of the j–th swarm,

$$x_i^{[j]} = \left(x_{i1}^{[j]}, x_{i2}^{[j]}, \ldots, x_{i,n_j}^{[j]}\right)^\top,$$

is evaluated using the buffer vector, P, by substituting the components that correspond to the contribution of the j–th swarm with the components of $x_i^{[j]}$, while the rest components remain unaffected, i.e.,

$$f\left(x_i^{[j]}\right) = f\left(P_i^{[j]}\right), \quad (5)$$

where,

$$P_i^{[j]} = \Big(z_1^{[1]}, \ldots, z_{n_1}^{[1]}, \ldots, \underbrace{x_{i1}^{[j]}, \ldots, x_{i,n_j}^{[j]}}_{\text{evaluated particle}}, \ldots, z_1^{[K]}, \ldots, z_{n_K}^{[K]}\Big)^\top,$$

$i = 1, 2, \ldots, N_j$; $j = 1, 2, ..., K$.

The most obvious choice of contribution of each subswarm is its overall best position, i.e.,

$$z^{[k]} = p_g^{[k]} = \left(p_{g1}^{[k]}, p_{g2}^{[k]}, \ldots, p_{g,n_k}^{[k]}\right)^\top,$$

which results in a buffer that contains the overall best positions of all subswarms:

$$P = \Big(\underbrace{p_{g1}^{[1]}, \ldots, p_{g,n_1}^{[1]}}_{p_g^{[1]} \text{ of } \mathbb{S}_1}, \underbrace{p_{g1}^{[2]}, \ldots, p_{g,n_2}^{[2]}}_{p_g^{[2]} \text{ of } \mathbb{S}_2}, \ldots, \underbrace{p_{g1}^{[K]}, \ldots, p_{g,n_K}^{[K]}}_{p_g^{[K]} \text{ of } \mathbb{S}_K} \Big)^\top. \quad (6)$$

In this case, by definition, the buffer constitutes the best position ever attained by the algorithm, i.e., it is the best obtained approximation of the global minimizer of $f(x)$.

Table 1: Pseudocode of the proposed COMPSO approach.

Input:	K (number of swarms), N_i (swarm sizes), n_i (swarm dimensions), $i = 1, 2, \ldots, K$, P (buffer vector), $D_{\min}$ (diversity threshold), f (objective function)
Step 1.	**Initialize** all subswarms randomly in their search spaces (subspaces of the original one).
Step 2.	**Initialize** buffer vector P using a randomly selected particle from each subswarm.
Step 3.	**Do** $(k = 1, 2, \ldots, K)$
Step 4.	**Do** $(i = 1, 2, \ldots, N_k)$
Step 5.	**Update** the particle $x_i^{[k]}$ using Eqs. (1) and (2).
Step 6.	**Evaluate** $x_i^{[k]}$ using Eq. (5) and the buffer P.
Step 7.	**Update** the best position $p_i^{[k]}$ using Eq. (3).
Step 8.	**If** $\left(f(x_i^{[k]}) < f(P)\right)$ **Then**
Step 9.	**Copy** $x_i^{[k]}$ in the proper position of buffer P.
Step 10.	**End If**
Step 11.	**End Do**
Step 12.	**Compute** standard deviations, d_j, $j = 1, 2, \ldots, n_k$, for all coordinate directions.
Step 13.	**If** $\left(\min_j\{d_j\} < D_{\min}\right)$ **Then**
Step 14.	**Re–initialize** the k–th subswarm randomly retaining its best positions.
Step 15.	**End If**
Step 16.	**End Do**
Step 17.	**Print** buffer P and $f(P)$.

Different choices of buffer result in different properties of the algorithm. For example, instead of the overall best position, a randomly selected particle from each subswarm can be considered as its contribution resulting in a cooperative scheme with slower convergence but higher diversity.

The proposed COMPSO approach combines Micro–PSO with the cooperative approaches described above. More specifically, the problem at hand is broken down in low–dimensional subcomponents and Micro–PSO is applied on each one. In order to render it capable of tackling the assigned problems using only a few particles, the size of each subswarm was selected to be higher than the dimension of the corresponding problem subcomponent.

Since Micro–PSO subswarms shall have a very small size, the dimension of subcomponents shall be selected such that the ratio, $D_i = n_i/N_i$, $i = 1, 2, \ldots, K$, lies as close to zero as possible for each subswarm. In practice, a ratio around 0.5 provides satisfactory results. For example, for subswarm sizes, $N_i = 5$, a partitioning of the original problem to subcomponents of dimension, $n_i = 3$, $i = 1, 2, \ldots, K$, is expected to be an effective configuration with 5 particles operating on 3 coordinate directions of the objective function.

Moreover, in order to avoid search stagnation, a threshold, $D_{\min}$, was set for each subswarm and the standard deviation per coordinate direction of its particles were computed at each iteration. If the smallest standard deviation falls under this threshold, then the subswarm is restarted randomly, retaining its best positions. The COMPSO algorithm is reported in pseudocode in Table 1. In the next section, experimental results on high–dimensional instances of widely used test problems are reported.

4. EXPERIMENTAL RESULTS

COMPSO was applied on the following widely used test problems:

Test Problem 1 (TP1 - Sphere) [21]. This n–dimensional problem is defined as:

$$f(x) = \sum_{i=1}^{n} x_i^2, \tag{7}$$

and it has a global minimizer, $x^* = (0, 0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 2 (TP2 - Generalized Rosenbrock) [22]. This n–dimensional problem is defined as:

$$f(x) = \sum_{i=1}^{n-1} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right), \tag{8}$$

and it has a global minimizer, $x^* = (1, 1, \ldots, 1)^\top$, with $f(x^*) = 0$.

Test Problem 3 (TP3 - Rastrigin) [21]. This n–dimensional problem is defined as:

$$f(x) = 10n + \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i)\right), \tag{9}$$

and it has a global minimizer, $x^* = (0, 0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 4 (TP4 - Griewank) [21]. This n–dimensional problem is defined as:

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \tag{10}$$

and it has a global minimizer, $x^* = (0, 0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Test Problem 5 (TP5 - Ackley) [1]. This n–dimensional

Table 2: Dimension and range for each test problem.

Problem	Dimension (n)	Range
TP1	150, 300, 600, 900, 1200	$[-100, 100]^n$
TP2	150, 300, 600, 900, 1200	$[-30, 30]^n$
TP3	150, 300, 600, 900, 1200	$[-5.12, 5.12]^n$
TP4	150, 300, 600, 900, 1200	$[-600, 600]^n$
TP5	150, 300, 600, 900, 1200	$[-20, 30]^n$

Table 3: The total number of particles and number of subswarms per dimension.

Problem dimension	Total number of particles	Number of subswarms	Particles per subswarm
150	250	50	5
300	500	100	5
600	1000	200	5
900	1500	300	5
1200	2000	400	5

problem is defined as:

$$f(x) = 20 + \exp(1) - 20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right), \quad (11)$$

and it has a global minimizer, $x^* = (0, 0, \ldots, 0)^\top$, with $f(x^*) = 0$.

Each test problem was considered for five different dimensions, namely, $n = 150, 300, 600, 900$, and 1200. The corresponding n–dimensional search spaces are reported in Table 2. COMPSO divides each problem in K, 3–dimensional subcomponents and applies a Micro–PSO subswarm of 5 particles on each. Hence, using the notation of the previous section, we have:

$$n_k = 3, \qquad N_k = 5,$$

for all $k = 1, 2, \ldots, K$, with $K = n/3$. The number of subswarms as well as the total number of particles per dimension are reported in Table 3.

Each subswarm was restarted as soon as its minimum standard deviation per coordinate direction fell under the threshold, $D_{\min} = 10^{-5}$. A maximum number of 1000 iterations was allowed to each subswarm for every test problem and dimension. Regarding their parameters, the local variant with ring neighborhood topology of radius, $r = 1$, was used for all subswarms due to its nice exploration properties, along with the default parameters, $\chi = 0.729$, $c_1 = c_2 = 2.05$ [2]. All subswarm parameters are summarized in Table 4. The parameters were set to reasonable values for each problem case, without any further fine–tuning that could enhance the performance of the proposed approach.

For each test problem, 30 independent experiments were performed. At each experiment, COMPSO was allowed to perform 1000 iterations and the best solution it ever detected was recorded along with its function value. The obtained function values were then analyzed statistically, in terms of their mean, standard deviation, minimum and maximum. For comparison purposes, experiments were conducted also for the standard local PSO variant with the same parameters as COMPSO. In order to make fair comparisons, the swarm size of PSO was set equal to the total number of particles employed by COMPSO, which is reported in Table 3, and the same number of iterations was allowed. Statistics on the performance of PSO were also derived and compared with that of COMPSO.

Table 4: Subswarm parameters.

Parameter	Value
N_k (size)	5
n_k (dimension)	3
$D_{\min}$ (restart threshold)	10^{-5}
$t_{\max}$ (maximum iterations)	10^3
χ (constriction coefficient)	0.729
c_1, c_2	2.05
Neighborhood topology	Ring
r (neighborhood radius)	1

Moreover, statistical hypothesis tests were conducted to ensure significance of the results. Therefore, for each problem, COMPSO was compared against PSO using the non-parametric Wilcoxon rank–sum test [4] with the null hypothesis that the two samples of final solution values in 30 experiments come from identical continuous distributions with equal medians, against the alternative that they do not have equal medians. The decision for acceptance or rejection of the null hypothesis, as well as the corresponding p–value, were recorded for each test problem.

All results and statistical tests are reported in Tables 5–14. In all cases, COMPSO clearly outperforms PSO. The solution values achieved by COMPSO are several orders of magnitude better than those of PSO. Also, COMPSO appears to be less affected when dimension of the problem increases significantly. For example, in TP1, increasing dimension from 150 to 1200 results in a difference of one order of magnitude in the performance of COMPSO (from 10^{-9} to 10^{-8}). At the same time, the performance of PSO changes by three orders of magnitude (from 10^2 to 10^5). Similar observations can be made for the rest problems, while, in TP4 and TP5, COMPSO's performance is marginally affected by dimension increase.

The apparent, naked–eye superiority of COMPSO was verified also by the statistical hypothesis tests. In all cases, the obtained p–values of the Wilcoxon tests were very small (of order 10^{-11}), thereby rejecting the null hypothesis of samples with equal medians.

The reported results support the claim that COMPSO can solve high–dimensional problems efficiently. Its performance was retained in high levels regardless of the increase in problem dimension, while in some cases it was even left unaffected. Although subswarms of equal size and dimension were used in the reported experiments, this can change arbitrarily if required. Moreover, fine–tuning of parameters can further enhance the performance of COMPSO, while its straightforward parallelization capability renders it a very promising approach in high–dimensional problems.

Table 5: Results for the 150–dimensional problems. PSO uses a single swarm of 250 particles, while COMPSO uses 50 subswarms of 5 particles each.

Prob.	Stats.	PSO	COMPSO
TP1	Mean	$6.68847230e+02$	$1.55261791e-09$
	StD	$6.90638180e+01$	$3.79786997e-10$
	Min	$5.14616100e+02$	$9.85937300e-10$
	Max	$8.02410100e+02$	$2.70222600e-09$
TP2	Mean	$3.11629990e+05$	$1.71112142e+02$
	StD	$4.74332076e+04$	$4.62286507e+01$
	Min	$2.24682100e+05$	$7.35445600e+01$
	Max	$3.88668200e+05$	$2.99587000e+02$
TP3	Mean	$6.94741740e+02$	$4.69250047e+01$
	StD	$6.09857058e+01$	$7.36663120e+00$
	Min	$5.49309700e+02$	$3.21413300e+01$
	Max	$8.11608800e+02$	$6.41216700e+01$
TP4	Mean	$7.10879660e+00$	$4.29663675e-02$
	StD	$7.11968399e-01$	$5.84414428e-02$
	Min	$5.61826600e+00$	$5.88289100e-11$
	Max	$8.36398000e+00$	$2.70933600e-01$
TP5	Mean	$5.12332983e+00$	$1.22642674e-05$
	StD	$2.35771525e-01$	$1.27414193e-06$
	Min	$4.64959000e+00$	$9.82757100e-06$
	Max	$5.50357100e+00$	$1.48024800e-05$

Table 6: Wilcoxon rank–sum tests for the 150–dimensional problems.

Prob.	p–value	Decision
TP1	$3.01985936e-11$	Reject
TP2	$3.01985936e-11$	Reject
TP3	$3.01985936e-11$	Reject
TP4	$2.95983137e-11$	Reject
TP5	$3.01985936e-11$	Reject

Table 7: Results for the 300–dimensional problems. PSO uses a single swarm of 500 particles, while COMPSO uses 100 subswarms of 5 particles each.

Prob.	Stats.	PSO	COMPSO
TP 1	Mean	$1.29715330e+04$	$4.46924800e-09$
	StD	$8.90546008e+02$	$7.09153474e-10$
	Min	$1.13007800e+04$	$3.56440200e-09$
	Max	$1.48469900e+04$	$6.06290100e-09$
TP2	Mean	$9.01191120e+06$	$3.42852143e+02$
	StD	$1.11074795e+06$	$4.57457011e+01$
	Min	$6.65948200e+06$	$2.80865000e+02$
	Max	$1.06025700e+07$	$4.39362000e+02$
TP3	Mean	$1.99022420e+03$	$1.00995904e+02$
	StD	$8.12381020e+01$	$1.06166877e+01$
	Min	$1.82741800e+03$	$7.36630800e+01$
	Max	$2.15461500e+03$	$1.13846800e+02$
TP4	Mean	$1.18179580e+02$	$4.08703599e-02$
	StD	$5.72494497e+00$	$8.50423385e-02$
	Min	$1.04005700e+02$	$3.18277600e-11$
	Max	$1.29898800e+02$	$4.36516000e-01$
TP5	Mean	$9.21636153e+00$	$1.44741703e-05$
	StD	$1.56635755e-01$	$1.08774777e-06$
	Min	$8.83649700e+00$	$1.23276500e-05$
	Max	$9.63042100e+00$	$1.63996100e-05$

Table 8: Wilcoxon rank–sum tests for the 300–dimensional problems.

Prob.	p–value	Decision
TP1	$3.01985936e-11$	Reject
TP2	$3.01985936e-11$	Reject
TP3	$3.01985936e-11$	Reject
TP4	$3.01040737e-11$	Reject
TP5	$3.01985936e-11$	Reject

5. CONCLUSIONS

COMPSO, an approach that combines Cooperative PSO schemes with Micro–PSO for solving high–dimensional problems was introduced. The proposed approach was applied on five widely used test problems for dimensions ranging from 150 up to 1200, and it was compared with the corresponding local PSO variant. COMPSO was shown to be very efficient, with its performance being marginally affected or completely unaffected by increases in problem dimension.

Drawbacks identified for Cooperative PSO methods remain to be investigated for COMPSO, especially in cases with highly–correlated coordinate directions. Nevertheless, COMPSO was shown to be both efficient and robust and worths further investigation under different subswarm configurations and cooperative schemes. Applications on real world problems also constitute a very interesting direction for future research.

6. ACKNOWLEDGMENTS

The work of K.E. Parsopoulos was supported by the State Scholarship Foundation of Greece (I.K.Y.).

7. REFERENCES

[1] D. H. Ackley. *A Connectionist Machine for Genetic Hillclimbing*. Kluwer Academic Publishers, Boston, 1987.

[2] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.*, 6(1):58–73, 2002.

[3] H. G. Cobb. Is the genetic algorithm a cooperative learner? In *Foundations of Genetic Algorithms 2*, pages 277–296. Morgan Kaufmann, 1992.

[4] W. J. Conover. *Practical Nonparametric Statistics*. Wiley, 1998.

Table 9: Results for the 600–dimensional problems. PSO uses a single swarm of 1000 particles, while COMPSO uses 200 subswarms of 5 particles each.

Prob.	Stats.	PSO	COMPSO
TP1	Mean	$8.42272790e+04$	$1.24702323e-08$
	StD	$2.39026442e+03$	$1.22528183e-09$
	Min	$7.88212500e+04$	$9.84027000e-09$
	Max	$8.81832900e+04$	$1.49186500e-08$
TP2	Mean	$7.20676663e+07$	$7.12490387e+02$
	StD	$5.45561950e+06$	$8.22158761e+01$
	Min	$5.93519200e+07$	$5.74899300e+02$
	Max	$8.45940800e+07$	$9.68409600e+02$
TP3	Mean	$5.09383190e+03$	$2.09947920e+02$
	StD	$1.14060563e+02$	$1.24609857e+01$
	Min	$4.78738800e+03$	$1.86601700e+02$
	Max	$5.26620300e+03$	$2.33560500e+02$
TP4	Mean	$7.60530330e+02$	$6.71039370e-02$
	StD	$2.90882655e+01$	$1.76657773e-01$
	Min	$7.08135800e+02$	$5.75698300e-11$
	Max	$8.25788500e+02$	$8.82017800e-01$
TP5	Mean	$1.29157233e+01$	$1.56977400e-05$
	StD	$1.48305745e-01$	$5.75971741e-07$
	Min	$1.25799400e+01$	$1.45849400e-05$
	Max	$1.31694600e+01$	$1.66819600e-05$

Table 10: Wilcoxon rank–sum tests for the 600–dimensional problems.

Prob.	p–value	Decision
TP1	$3.01985936e-11$	Reject
TP2	$3.01985936e-11$	Reject
TP3	$3.01985936e-11$	Reject
TP4	$2.98408520e-11$	Reject
TP5	$3.01985936e-11$	Reject

Table 11: Results for the 900–dimensional problems. PSO uses a single swarm of 1500 particles, while COMPSO uses 300 subswarms of 5 particles each.

Prob.	Stats.	PSO	COMPSO
TP1	Mean	$2.00505733e+05$	$2.10178767e-08$
	StD	$5.35649956e+03$	$1.66499862e-09$
	Min	$1.86727100e+05$	$1.78246200e-08$
	Max	$2.09774600e+05$	$2.42341400e-08$
TP2	Mean	$1.93007190e+08$	$1.09891939e+03$
	StD	$1.08314967e+07$	$8.01280591e+01$
	Min	$1.70273400e+08$	$9.79215100e+02$
	Max	$2.08754000e+08$	$1.29033400e+03$
TP3	Mean	$8.45629317e+03$	$3.21360810e+02$
	StD	$1.15925376e+02$	$1.66377811e+01$
	Min	$8.16662800e+03$	$2.96558300e+02$
	Max	$8.67311100e+03$	$3.68271200e+02$
TP4	Mean	$1.79961443e+03$	$2.65183506e-02$
	StD	$6.44995545e+01$	$5.04872530e-02$
	Min	$1.65445100e+03$	$7.50487000e-11$
	Max	$1.92371600e+03$	$2.56953900e-01$
TP5	Mean	$1.45878167e+01$	$1.73350403e-05$
	StD	$7.02359333e-02$	$5.55510464e-07$
	Min	$1.43691700e+01$	$1.62123400e-05$
	Max	$1.47042600e+01$	$1.84951300e-05$

Table 12: Wilcoxon rank–sum tests for the 900–dimensional problems.

Prob.	p–value	Decision
TP1	$3.01985936e-11$	Reject
TP2	$3.01985936e-11$	Reject
TP3	$3.01985936e-11$	Reject
TP4	$3.01796680e-11$	Reject
TP5	$3.01985936e-11$	Reject

[5] R. C. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings Sixth Symposium on Micro Machine and Human Science*, pages 39–43, Piscataway, NJ, 1995. IEEE Service Center.

[6] R. C. Eberhart, P. Simpson, and R. Dobbins. *Computational Intelligence PC Tools*. Academic Press, 1996.

[7] M. El-Abd. *Cooperative Models of Particle Swarm Optimizers*. PhD thesis, Dept. Elect. Comput. Eng., Univ. Waterloo, Waterloo, Ontario, Canada, 2008.

[8] T. Huang and A. S. Mohan. Micro–particle swarm optimizer for solving high dimensional optimization problems (μPSO for high dimensional optimization problems. *Applied Mathematics and Computation*, 181(2):1148–1154, 2006.

[9] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE Int. Conf. Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[10] J. Kennedy and R. C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[11] M. Köppen, K. Franke, and R. Vicente-Garcia. Tiny GAs for image processing applications. *IEEE Computational Intelligence Magazine*, 1(2):17–26, 2006.

[12] M. M. Millonas. Swarms, phase transitions, and collective intelligence. In M. Palaniswami, Y. Attikiouzel, R. Marks, D. Fogel, and T. Fukuda, editors, *Computational Intelligence: A Dynamic System Perspective*, pages 137–151. IEEE Press, Piscataway, NJ, 1994.

[13] Y. Ong, A. Keane, and P. Nair. Surrogate–assisted coevolutionary search. In *Proc. 9th International*

Table 13: Results for the 1200–dimensional problems. PSO uses a single swarm of 2000 particles, while COMPSO uses 400 subswarms of 5 particles each.

Prob.	Stats.	PSO	COMPSO
TP1	Mean	$3.46446790e+05$	$3.09769483e-08$
	StD	$1.25211959e+04$	$1.97360280e-09$
	Min	$3.14708400e+05$	$2.71700400e-08$
	Max	$3.63112000e+05$	$3.72747800e-08$
TP2	Mean	$3.57986637e+08$	$1.46841193e+03$
	StD	$1.50817090e+07$	$8.16410749e+01$
	Min	$3.25739700e+08$	$1.31141300e+03$
	Max	$3.89343700e+08$	$1.68527900e+03$
TP3	Mean	$1.17734573e+04$	$4.36089850e+02$
	StD	$1.12762515e+02$	$2.08992042e+01$
	Min	$1.15154600e+04$	$3.90303400e+02$
	Max	$1.19282800e+04$	$4.75047100e+02$
TP4	Mean	$3.10432017e+03$	$7.18691892e-02$
	StD	$9.03348378e+01$	$2.04131021e-01$
	Min	$2.90395200e+03$	$8.89046600e-11$
	Max	$3.27936300e+03$	$1.02541900e+00$
TP5	Mean	$1.55481950e+01$	$1.85364203e-05$
	StD	$9.03550123e-02$	$6.21871510e-07$
	Min	$1.53380300e+01$	$1.72929800e-05$
	Max	$1.57267600e+01$	$1.99353700e-05$

Table 14: Wilcoxon rank–sum tests for the 1200–dimensional problems.

Prob.	p–value	Decision
TP1	$3.01985936e-11$	Reject
TP2	$3.01985936e-11$	Reject
TP3	$3.01985936e-11$	Reject
TP4	$3.00474914e-11$	Reject
TP5	$3.01985936e-11$	Reject

Conference of Neural Information Processing, pages 1140–1145, 2002.

[14] K. E. Parsopoulos and M. N. Vrahatis. Initializing the particle swarm optimizer using the nonlinear simplex method. In A. Grmela and N.E. Mastorakis, editors, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 216–221. WSEAS Press, 2002.

[15] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. *Natural Computing*, 1(2–3):235–306, 2002.

[16] K. E. Parsopoulos and M. N. Vrahatis. On the computation of all global minimizers through particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):211–224, 2004.

[17] M. A. Potter and K. De Jong. A cooperative coevolutionary approach to function optimization. In Y. Davidor and H.-P. Schwefel, editors, *Proc. 3rd Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer–Verlag, 1994.

[18] M. A. Potter and K. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.

[19] R. E. Smith, S. Forrest, and A. S. Perelson. Searching for diverse, cooperative populations with genetic algorithms. *Evolutionary Computation*, 1(2):127–149, 1993.

[20] D. Sofge, K. De Jong, and A. Schultz. A blended population approach to cooperative coevoultion for decomposition of complex problems. In *Proc. 2002 IEEE Congress on Evolutionary Computation (CEC'02)*, pages 413–418, 2002.

[21] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization*, 11:341–359, 1997.

[22] I. C. Trelea. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 85:317–325, 2003.

[23] F. Van den Bergh and A. P. Engelbrecht. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239, 2004.