Using particle swarm optimization for finding robust optima

Carl-Edward Joseph Dippel
cdippel@liacs.nl

ABSTRACT

This report presents an in-depth look on methods on how to obtain robust optima in optimization problems via the family of particle swarm optimization **(PSO)** algorithms. Several members of this family, namely the canonical PSO, the fully-informed PSO, a multi-swarm PSO and the charged PSO, were tested on their usability for finding robust optima. To assess the usefulness of these algorithms, they were tested against each other (with different modifications) as well as against a current state-of-the-art robust evolutionary algorithms **(EA)**. The results produced show that the PSO algorithms are a promising method, when it comes to finding robust optima.

1. INTRODUCTION

For many optimization problems in science and industry, solutions can be found with various methods, ranging from mathematical programming to meta-heuristics. Using these methods it is possible to successfully find good to excellent approximations of optima. However, when it comes to real-world applications, the produced solutions by these methods are often not of much use, for example because of precision problems of manufacturing processes. Hence, what is often really needed in real-world problems are not just any solutions/optima but robust solutions/optima, which are not sensitive to input fluctuations or noise to a certain degree.

The area of robust optimization is relatively new and fresh at the time of writing. Thus, currently research into this area is at a stage of infancy, where a lot of effort seems to go into the area of evolutionary algorithms for robust optimization. A summary of the current state of research can for example be found in [2, 8].

PSO is relatively new in itself, as this stochastic optimization algorithm (family) has only been found recently in the 1990s. It is an open question whether the PSO paradigm lends itself also well for robust optimization or how it needs to be extended to address such problems. Hence, this paper tries to fill this knowledge gap and tries to show how swarm-based algorithms, in particular PSO algorithms, can be utilized to find robust optima using suitable extensions.

This report will try to answer some fundamental questions: How can PSO algorithms be used for finding robust optima? What changes can be made to help the canonical PSO algorithms in finding robust optima? How well do PSO algorithms perform on different problem types? How much impact can certain (micro-)optimizations (swarm-sizes, special topologies, diversification methods, ...) have? How do the different PSO algorithms fair against each other and against other meta-heuristics?

The paper is structured as follows: Section 2 gives a formal introduction and definition on robustness. Section 3 presents the used PSO algorithms and their extensions are discussed. Section 4 gives an overview of the experiments, which were conducted and makes comparisons between the PSO algorithms and a state-of-the-art evolutionary algorithm for robust optimization. Sections 5 and 6 end with conclusions and an outlook for future research. Section 7 offers a small discussion on elitism and noise.

2. ROBUSTNESS

In the literature [9, 8, 5, 16] there are three (similar) interpretations or viewpoints on what robust optimization is: 1) finding optima given noisy or uncertain objective functions, 2) finding optima that remain stable under fluctuations of the input variables, and 3) finding optima in dynamic environments.

In this paper the focus is set towards finding robust optima on continuous optimization problems with uncertainty or fluctuations in the input variables.

Mathematically this means a function $f(\vec{x}) \rightarrow min/max$ is to be optimized, which experiences fluctuations $\vec{\delta}$ on its input variables $\vec{x}$.

To assess the robustness of a solution for $f(\vec{x})$ there are two common measures which are normally employed [9]:

- the expected fitness
$$f_{exp}(\vec{x}) = \int_{-\infty}^{\infty} f(\vec{x}+\vec{\delta}) \cdot pdf(\vec{\delta}) d\vec{\delta}, \tag{1}$$
which is calculated by taking into account the input variable fluctuations $\vec{\delta}$ and their possibility of occurring via a probability distribution function $pdf(\vec{\delta})$ over the whole input variable space $[-\infty, \infty]^N$, where N is the problem's dimension,
- and the worst-case fitness
$$f_{wc}(\vec{x}) = \sup_{\vec{\delta} \in \eta} f(\vec{x}+\vec{\delta}), \tag{2}$$
which is calculated by finding the supremum of the fluctuating fitness values over an area η, which describes the region of possible variations.

In complex problems these formulas are often not obtainable in a closed form and need to be approximated. For the approximation of the effective fitness f_{eff} normally Monte-Carlo sampling/integration is employed, and for the approximation of the worst-case fitness f_{wc} a min-max approach is normally used. These approximations can be very expensive, depending on the precision required and may not produce similar results upon successive applications, because of their random nature, leading to noisy robustness measures, which are highly dependent on the parameters used in the approximation process.

3. ALGORITHMS

The paper looks at several base PSO algorithms for their usability in finding robust optima: a canonical PSO algorithm **CPSO** [14], a fully-informed PSO algorithm **FIPS** [12], a charged CPSO algorithm **CCPSO** [3, 7] and a multi-swarm PSO algorithm **MPSO** [7]. All the algorithms are very similar to one another more or less and only differ in a few details. By looking at several PSO algorithms it should be possible to see if any algorithm might have an edge in a one way or the other. To make these base algorithms usable for finding robust optima they undergo some key modifications.

3.1 Base algorithms

3.1.1 CPSO

The canonical PSO **(CPSO)** algorithm was developed by Kennedy and Eberhart in 1995 [14, 1]. It is a population based optimization technique, which was inspired by the special behavior of animals in swarms like fish schools and bird flocks. The general idea is that in the PSO algorithm there exists a swarm of particles and each particle i resides at a position $\vec{x}_i$ in the search space. Each position is associated with a fitness given by the fitness function $f(\vec{x}_i)$. The particles move over the search space with a certain velocity $\vec{v}_i$.

The velocity is influenced by the global best position $\vec{g}_i$ and fitness g_{best} a neighbor has found so far and the best position $\vec{p}_i$ and fitness p_{best} a particle has personally found. After a certain amount of time the particles of the swarm converge towards positions it finds optimal (local and/or global).

The degree of influence the personal best solution should have is defined by a coefficient ϕ_1. Likewise the influence of the best global or neighborhood solution is defined by a coefficient ϕ_2. The velocity update function, which drives the CPSO algorithm forward, is defined mathematically as

$$\vec{v}_i := \chi \cdot (\vec{v}_i \tag{3}$$
$$+ \vec{U}(0, \phi_1) \otimes (\vec{p}_i - \vec{x}_i) \tag{4}$$
$$+ \vec{U}(0, \phi_2) \otimes (\vec{g}_i - \vec{x}_i)), \tag{5}$$

where $\vec{U}$ is a uniformly random number sampled out of $[0, \phi_1]$ or $[0, \phi_2]$, and χ is a constriction coefficient, which was introduced in [6] and it helps in preventing velocity explosions. It is defined as:

$$\chi = \frac{2}{\phi + \sqrt{\phi^2 - 4\phi}}, \text{with } \phi = \phi_1 + \phi_2 > 4. \tag{6}$$

In the velocity update function, (3) is called the momentum, which represents the particle's current direction, (4) is called the social component, which is the force of being attracted towards the best solution so far found by the neighbors, (5) is called the cognitive component and refers to the force of being attracted towards previous best known solutions the particle knows about.

To define a neighborhood relationship among the particles, neighborhood topologies are used. There are two main ways on how to define them: **Geographical neighborhood topologies**, which are based on geographical closeness (for example the euclidean distance of the particles) and **Communication neighborhood topologies**, which define a logical communication network (much in the same way it is done in real communication networks). In the report focus is held towards communication neighborhood topologies, because they are computationally less expensive. These topologies hold data on whether a particle is a neighbor of another particle. Hence, they basically represent graphs and can thus be stored in adjacency matrices or lists. Figures 1 and 2 show two topologies, which are commonly used in PSO algorithms: the ring and the all topology as they were called in [12]. The ring has the advantage of having a higher diversity and thus being able to find global optima better and the all topology has shown itself to be well suited for finding local optima and converging fast [7].

Aside from these tried and tested standard topologies, this report also looks at a custom topology. This custom topology creates groups with a maximum of m members out of n particles in total. The members of a group are fully connected. In each group there is a "leader particle" and the leader particles are connected in a ring to exchange information with each other. The motivation of using this topology is to combine the benefits of the ring and all topology. The sub-groups are there for exploitation and supposed to converge towards an interesting area, while the leader particles would be there to help keep diversity up.

Algorithm 1 Canonical PSO (*CPSO*)

```
input: Personal best coefficient $\phi_1$
input: Neighborhood best coefficient $\phi_2$
input: Number of particles to use $n$
input: Fitness function $f$
input: Neighborhood topology $T$
input: Search space lower bounds $\vec{lb}$
input: Search space upper bounds $\vec{ub}$
 1: $X = \{\vec{x_1}, ..., \vec{x_n}\} := InitParticlePositions(\vec{lb}, \vec{ub}) \rightarrow \forall i \in \{1, ..., n\} : \vec{x_i} := \vec{U}(\vec{lb}, \vec{ub})$
 2: $V = \{\vec{v_1}, ..., \vec{v_n}\} := InitParticleVelocities(\vec{lb}, \vec{ub}) \rightarrow \forall i \in \{1, ..., n\} : \vec{v_i} := (\vec{ub} - \vec{lb}) \otimes \vec{U}(0, 1) - \frac{1}{2} \cdot (\vec{ub} - \vec{lb})$
 3: $Y = \{y_1, ..., y_n\} := EvaluateFitness(X, f) \rightarrow \forall i \in \{1, ..., n\} : y_i := f(\vec{x_i})$
 4: $P = \{\vec{p_1}, ..., \vec{p_n}\} := InitPersonalPositions(X) \rightarrow X$
 5: $P^f = \{p_1^f, ..., p_n^f\} := InitPersonalFitness(Y) \rightarrow Y$
 6: $G = \{\vec{g_1}, ..., \vec{g_n}\} := InitGlobalPositions(P, T) \rightarrow P$
 7: $G^f = \{g_1^f, ..., g_n^f\} := InitGlobalFitness(P^f, T) \rightarrow P^f$
 8: while termination condition not met do
 9:     for each particle $i$ of $n$ do
10:         $\vec{v_i} := \chi \cdot (\vec{v_i} + \vec{U}(0, \phi_1) \otimes (\vec{p_i} - \vec{x_i}) + \vec{U}(0, \phi_2) \otimes (\vec{g_i} - \vec{x_i}))$
11:         $\vec{x_i} := \vec{x_i} + \vec{v_i}$
12:     end for
13:     $Y := EvaluateFitness(X, f)$
14:     $P, P^f := UpdatePersonalBest(X, Y) \rightarrow \forall i \in \{1, ..., n\} : \vec{p_i}, p_i^f := \begin{cases} \vec{x_i}, y_i, & \text{if } y_i \text{'better' than } p_i^f \\ \vec{p_i}, p_i^f, & \text{otherwise} \end{cases}$
15:     $G, G^f := UpdateGlobalBest(P, P^f, T) \rightarrow \forall i \in \{1, ..., n\} : \vec{g_i}, g_i^f := best(P_{T_i}, P^f_{T_i})$, where $T_i$ are the neighbors of $i$
16: end while
output: Best solution found (norm. min/max of $G, G^f$)
```

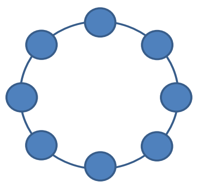


Figure 1: Ring topology

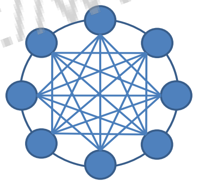


Figure 2: All topology

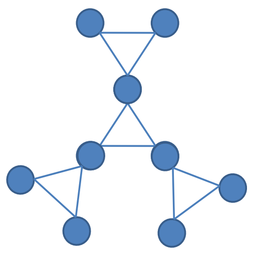


Figure 3: (3,3) m,n-group topology

For further reference this topology shall be called an **m,n-group topology**. An example of a (3,9) m,n-group or 3,9-group topology is shown in Figure 3.

To initialize the PSO algorithms, commonly a randomized approach is used, as is done here. However, it is also possible to initialize differently (for example by including or taking into account problem-specific information).

Algorithm 1 gives a detailed description of the CPSO algorithm.

3.1.2 FIPS

The fully informed PSO algorithm **FIPS** is very similar to the **CPSO** algorithm. It only differs in the velocity update function. In the FIPS algorithm not just the best neighbor solution is taken into account but all of them. Hence, the name fully informed particle swarm [12].

The velocity update function in FIPS is defined as

$$\vec{v_i} := \chi \cdot (\vec{v_i} + \frac{1}{K_i} \sum_{m=1}^{K_i} \vec{U}(0, \phi_1) \otimes (\vec{p}_{nbr_m} - \vec{x_i})), \quad (7)$$

where K_i is the number of neighbors for particle i and $\vec{p}_{nbr_m}$ is the m-th neighbor of particle i.

3.1.3 CCPSO

The charged CPSO algorithm [7] is related to the CPSO algorithm and can even be seen as a further generalization. The two algorithms are almost the same, just that the charged PSO has an addition to its velocity update function.

The velocity update function of this PSO variant is defined as

$$\begin{aligned}\vec{v}_i := \chi \cdot (&\vec{v}_i \\ &+ \vec{U}(0,\phi_1) \otimes (\vec{p}_i - \vec{x}_i) \\ &+ \vec{U}(0,\phi_2) \otimes (\vec{g}_i - \vec{x}_i) \\ &+ \vec{a}_i),\end{aligned} \tag{8}$$

where $\vec{a}_i$ is an acceleration/repelling coefficient. This coefficient is used to calculate a repelling force between particle i and other particles. The motivation of adding repelling forces is to decrease the chances of all/most of the particles (prematurely) converging towards certain optima and to keep diversity of the population up, because a population of higher diversity could be benefiting in exploration processes and could lead to better results, especially in dynamic environments [3]. The value for the repulsion force a_i is calculated as

$$\vec{a}_i = \sum_{l=1, i \neq l}^{n} \vec{a}_{il}, \tag{9}$$

where a_{il} is the repulsion force between particle i and particle l and is calculated as

$$\vec{a}_{il} = \begin{cases} \frac{Q_{il}(\vec{x}_i - \vec{x}_l)}{||\vec{x}_i - \vec{x}_l||^3}, & R_c \leq ||\vec{x}_i - \vec{x}_l|| \leq R_p \\ \frac{Q_{il}(\vec{x}_i - \vec{x}_l)}{R_c^2 ||\vec{x}_i - \vec{x}_l||}, & R_c > ||\vec{x}_i - \vec{x}_l|| \\ 0, & R_p < ||\vec{x}_i - \vec{x}_l|| \end{cases}, \tag{10}$$

where Q_{il} is a charge value between particle i and l, R_c is the core radius (which helps with reducing repulsion force explosions in case the particles become very close) and R_p is the perception limit of each particle.

The charges of the particles can be stored in different ways. In this implementation, the charges are stored in a charge map/matrix. This is done for flexibility, as more general charge relations (i.e., unsymmetric charge relations) can be defined this way, than would be the case if particles were assigned fixed charges.

Three different charged swarm types have been studied in the literature: **Neutral**, where all charges are 0 (CCPSO transforms to CPSO), **Charged**, where charges are not equal 0, **Atomic**, where half of the swarm is charged and the other half is not. Out of these types it was found that the atomic swarm generally performed best in dynamic environments [7, 3]. A reason for this is that half of the (non-charged) particles are there to converge towards an area and the other (charged) half is there to explore and react to changes in the dynamic fitness landscape. The neutral swarm naturally does not have this ability, especially if it is already at an advanced stage of converging towards an optima. The charged swarm on the other hand does a lot of exploration but lacks real convergence behavior. Since in this swarm type all particles have a charge, they try to move away from each other. This in effect creates a high possibility of not being able to properly zoom in to an optima.

3.1.4 MPSO

The multi-swarm PSO (MPSO) [7] algorithm is another PSO modification, which aims to increase diversity and help with reducing the likelihood of the algorithm possibly concentrating too much on one optimum. The MPSO algorithm used in this report uses a main swarm M with n particles $(\vec{x}_i, \vec{v}_i, y_i, \sigma_i), i \in \{1, ..., n\}$, where $\vec{x}_i$ defines the position, $\vec{v}_i$ is the speed and direction, y_i the fitness value and σ_i the standard deviation of the fitness values over an observation window w of the particle i. In the main swarm the particles run on only the momentum and cognitive components of the velocity update function. The main swarm is used to explore the search landscape. From this main swarm sub-swarms S_k are generated. A new sub-swarm is generated, when a particle $(\vec{x}_i, \vec{v}_i, y_i, \sigma_i)$ from the main swarm does not have a change in its fitness, which is measured by the standard deviation σ_i of the fitness values over a defined time frame. Non-change of fitness for a particle i is detected by the standard deviation σ_i going below a threshold ϵ for an observation window w.

When a new sub-swarm S_k is formed from the main-swarm M, the particle which triggered the event $(\vec{x}_i, \vec{v}_i, y_i, \sigma_i) \in M$ and the particle closest to it $(\vec{x}_l, \vec{v}_l, y_l, \sigma_l) \in M$ get merged into the new sub-swarm and are removed from the main swarm. Each sub-swarm has a center $S_k.\vec{c}$ and a radius $S_k.r$. The sub-swarm center is defined to be the position of the particle in the sub-swarm with the best fitness value. The sub-swarm radius is the farthest (euclidean) distance from the center to one of its particles. Algorithm 2 shows the process of sub-swarm creation in detail.

If a particle from the main swarm moves into a sub-swarm's radius, it gets absorbed into it. Other sub-swarms can merge together, when the cores of two swarms breach a defined distance d_{max}.

Each generated sub-swarm acts as an independent swarm entity and runs on a full velocity update function. Hence, a lot of different MPSO variants are possible depending on which full PSO model is used for by the sub-swarms. In the presented implementation the velocity update function of the FIPS algorithm was used.

By using several sub-swarms the MPSO algorithm promises more diversity, than standard PSO algorithms as each sub-swarm could concentrate on different areas of the search space, which in turn could be beneficial for finding better end solutions.

Algorithm 3 gives a detailed explanation of the MPSO algorithm. Figure 4 visualizes the MPSO processes swarm creation (T1), absorption (T2) and merging (T3) in a search space S. The blue or non-marked particles in the picture denote random particles in the search space. The green particles or those marked with an 'M' are particles that are absorbed into a sub-swarm. Red particles or those marked with a letter other than 'M' are sub-swarm particles and their letter defines to which sub-swarm they belong too. Sub-swarm particles with a perforated 'membrane' hold a special position, they are the sub-swarm's centers. Double pointed arrows define distances in the search space.

Algorithm 2 Sub-swarm creation

```
input: Set of all sub-swarms $Q$
input: Main swarm $M$
input: Threshold $\epsilon$
input: Distance measure $Distance(\vec{x}_1, \vec{x}_2) \rightarrow ||\vec{x}_1 - \vec{x}_2||$
input: Fitness function $f$
 1: for $\forall \vec{p}_i = (\vec{x}_i, \vec{v}_i, y_i, \sigma_i) \in M$ do
 2:   if $\sigma_i < \epsilon$ then
 3:     $\vec{p_l} = (\vec{x}_l, \vec{v}_l, y_l, \sigma_l) := ClosestParticle(M, \vec{p}_i) \rightarrow head\{(\vec{x}, \vec{v}, y, \sigma) \in M | \forall(\vec{x}', \_, \_, \_) \in M \setminus \{\vec{p}_i\} : ||\vec{x} - \vec{x}_i|| \leq ||\vec{x}' - \vec{x}_i||\}$
 4:     $S_{|Q|} = \{\vec{p}_i, \vec{p}_l\} := CreateSubSwarm()$
 5:     $S_{|Q|}.r := Distance(\vec{x}_i, \vec{x}_l)$
 6:     $\{\vec{y}_i, \vec{y}_l\} := EvaluateFitness(\{\vec{x}_i, \vec{x}_l\}, f)$
 7:     $S_{|Q|}.\vec{c} := SetCenter(S_{|Q|}) \rightarrow \begin{cases} \vec{x}_i, & \text{if } y_i \text{ 'better'} \\ \vec{x}_l, & \text{otherwise} \end{cases}$
 8:     $M := M \setminus S_{|Q|}$
 9:     $Q := Q \cup \{S_{|Q|}\}$
10:   end if
11: end for
output: Updated sub-swarm set $Q$ and main swarm $M$
```

In the swarm creation process depicted in T1, a particle 'A' has converged locally using the momentum and the cognitive components of the standard CPSO velocity update function. Using particle 'A' and the closest main-swarm particle to it, a new swarm is created. The best particle, which is in this case particle 'A' (arbitrarily chosen), is set as the swarm's center. The radius of the sub-swarm *Ra* is set to the distance between the two particles.

In T2 the particle absorption process is depicted. In this process there exists a sub-swarm in which its elements are marked with a 'B' and several main-swarm particles marked with an 'M'. The particles marked with an 'M' are inside the sub-swarms area, which is defined by its radius *Rb*, and because of this the particles 'M' of the main-swarm get absorbed into the sub-swarm 'B'.

The sub-swarm merging procedure, which is depicted in T3, illustrates two sub-swarms 'C' and 'D'. The distance between them is less than equal d_{max}, so the two sub-swarms merge together.

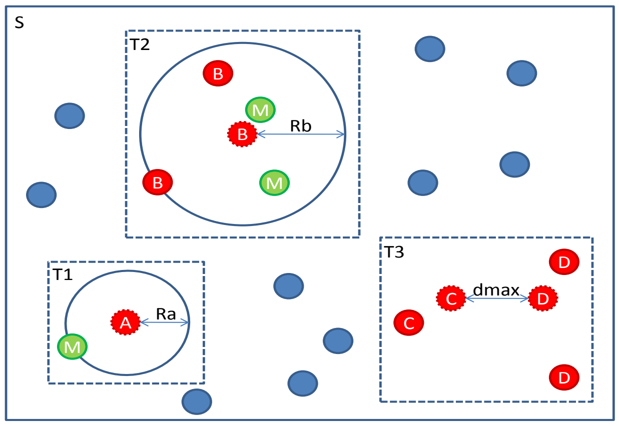


Figure 4: MPSO processes (swarm creation (T1), absorption (T2), merging (T3))

3.2 Modifications for robust optimization

Generally there are different methods that can be applied to help standard meta-heuristics in finding robust optima. These modifications generally approximate the robust fitness landscape by using information provided by the non-robust fitness landscape. The non-robust fitness landscape is computed by a fitness function f, while the robust fitness function landscape is approximated via the "effective fitness" $\hat{f}_{eff}$. In practice this would mean that the meta-heuristics would not use f, but $\hat{f}_{eff}$ instead during optimization runs.

3.2.1 *Raw*

The first and simplest way to approximate the robust fitness landscape is by using the standard algorithms as is. This means that the effective fitness becomes the normal fitness $\hat{f}_{eff} = f$. This method is also called the raw method. However, this method is not optimal. Studies have shown that the standard/raw approach usually only finds the robust optima, when the robust optima are at the same area as the non-robust optima, or when the algorithm had a very lucky run, as for example seen in [9]. These results are not surprising, as the robust fitness approximation quality is dependent on how similar the robust fitness landscape is to the non-robust fitness landscape. Hence, this modification is not further considered in this study.

3.2.2 *Sampling*

A more promising approach is to modify the algorithms to use one of the robustness measures for its effective fitness calculation: the expected fitness f_{exp} or the worst-case fitness f_{wc} [9, 2].

Since it is usually not possible to obtain closed form expressions for these measures for complex problems, they need to be approximated. A popular way to do this is by means of (Monte-Carlo) sampling in an area of interest $I = [\vec{x} - \vec{\sigma_\epsilon}, \vec{x} + \vec{\sigma_\epsilon}]$, where I is dependent on an interest point $\vec{x}$ and limits $\vec{\sigma_\epsilon}$, which define the boundaries of the region of uncertainty. In the area of interest I (input) disturbances $\vec{\delta}$ can occur with a probability defined by a probability distribution function *pdf*.

Algorithm 3 Multi-swarm PSO ($MPSO$)

```
input: Personal best coefficient $\phi_1$
input: Neighborhood best coefficient $\phi_2$
input: Number of particles to use $n$
input: Fitness function $f$
input: Neighborhood topology $T$
input: Max distance $d_{max}$
input: Threshold $\epsilon$
input: Search space lower bounds $\vec{lb}$
input: Search space upper bounds $\vec{ub}$
input: Observation window $w$
input: Distance measure $Distance$
 1: $A := M := MainSwarmInitialization(n, \vec{lb}, \vec{ub}) \rightarrow$ CPSO initialization procedure
 2: $Q := \{\} := SubSwarmInitialization()$
 3: $M := InitializeStandardDeviations(n) \rightarrow \forall i \in \{1, ..., n\} : \sigma_i := \infty$
 4: repeat
 5:    $M := MainSwarmTraining(M) \rightarrow \forall(\vec{x_i}, \vec{v_i}, \_, \_) \in M : \vec{v_i} := \chi \cdot (\vec{v_i} + \vec{U}(0, \phi_1) \otimes (\vec{p_i} - \vec{x_i})), \vec{x_i} := \vec{x_i} + \vec{v_i}$
 6:    for each sub-swarm $S_k \in Q$ do do
 7:       $S_k := SubSwarmTraining(S_k) \rightarrow \forall(\vec{x_i}, \vec{v_i}, \_, \_) \in S_k : \vec{v_i} := \chi \cdot (\vec{v_i} + \frac{1}{K_i} \sum_{m=1}^{K_i} \vec{U}(0, \phi_1) \otimes (\vec{p}_{nbr_m} - \vec{x_i})), \vec{x_i} := \vec{x_i} + \vec{v_i}$
 8:    end for
 9:    $EvaluateFitness(A, f)$
10:    $Q := UpdateSwarmCenters(Q) \rightarrow \forall S_k \in Q : S_k.\vec{c} := head\{\vec{x} | (\vec{x}, \_, y, \_) \in S_k, \forall(\vec{x}', \_, y', \_) \in S_k : y$ better or equal $y'\}$
11:    $Q := UpdateSwarmRadii(Q) \rightarrow \forall S_k \in Q : S_k.r := max\{||S_k.\vec{c} - \vec{x_i}||\ |(\vec{x_i}, \_, \_, \_) \in S_k\}$
12:    $Q := MergeSubSwarms(Q, d_{max}) \rightarrow \forall S_i \in Q : S' := \{S_j | S_j \in Q, ||S_i.\vec{c} - S_j.\vec{c}|| \leq d_{max}\}, S_i := \bigcup_{S'' \in S'} S'', Q := Q \setminus S'$
13:    $Q, M := AbsorbFromMain(Q, M) \rightarrow \forall S_k \in Q : X := \{(\vec{x}, \vec{v}, y, \sigma) \in M | ||\vec{x} - S_k.\vec{c}|| \leq S_k.r\}, S_k := S_k \cup X, M := M \setminus X$
14:    $M := UpdateStatistics(M, w)$
15:    $Q, M := CreateNewSubSwarms(Q, M, \epsilon, Distance, f)$
16: until Stopping condition
output: The best global solution or the best solutions of each swarm
```

For the approximation of the worst-case fitness $\hat{f}_{wc}$ this would mean n samples would be taken out of an area of interest I and the min/max (worst) of the sampled fitness values would be returned:

$$\hat{f}_{wc}(\vec{x}) = max\{f(\vec{x} + \vec{\delta}_i) | i \in \{1, ..., n\}\} \tag{11}$$

or

$$\hat{f}_{wc}(\vec{x}) = min\{f(\vec{x} + \vec{\delta}_i) | i \in \{1, ..., n\}\} \tag{12}$$

with $\vec{\delta}_i \sim pdf(\vec{\delta})$. For the expected fitness approximation $\hat{f}_{exp}$ the mean of the samples would be taken instead:

$$\hat{f}_{exp}(\vec{x}) = \frac{1}{n} \sum_{i=1}^{n} f(\vec{x} + \vec{\delta}_i). \tag{13}$$

This method of finding the expected fitness is in some literature also sometimes mentioned as the so called multi-evaluation model **(MEM)** [15]. For both approximations it is clear that using more sample points will increase the accuracy of the approximation. However, this can be problematic in practice, when evaluating a sample point (running the fitness function) is an expensive operation.

The formulas described here would be used in the fitness evaluation functions/parts $EvaluateFitness(*)$ of the PSO algorithms.

3.2.3 Archive

To reduce the number of fitness function evaluations (FFE), it is possible to use an archive to store and retrieve evaluated points in the search space. The archive in use here is based on the archive in [9] with slight modifications and is a data structure that stores pairs of points and fitness values $(\vec{x}, f)$.

To select and add points to the archive a selection scheme is used, which is described in detail by Algorithm 4. Values can be stored and read from the archive. When values are read from the archive an (LHS[1] [11]) sampling scheme is used. The sampling scheme creates m candidate sampling points out of an area of interest. For each candidate point its closest representative in the archive is computed. Each representative archive point for which the candidate sampling point is the closest as well, is added to a selected archive point set S. All other points are added to a re-sampling candidate point set X.

The set S can be used by a MEM function without a need for calling the main fitness function. On the other hand, all points from the set X are to be sampled/computed with the fitness function and the results are added to the archive. Once the values in X have been sampled, they can be used in the effective fitness calculation. Using the sets S and X, the calculation of the effective fitness value would be computed as

$$\hat{f}_{exp}(\vec{x}) = \frac{1}{|S \cup X|} \sum_{\vec{x}' \in X \cup S} f(\vec{x}'). \tag{14}$$

[1]A detailed explanation of why LHS sampling is preferably used is given in [9]. Alternatively other sampling techniques can be employed, such as uniform or stratified sampling.

Algorithm 4 Selection scheme

```
input: Number of samples to take $m$
input: Input variable disturbance range $\delta$
input: Archive $A$
input: Sampling function $Sampler \rightarrow LHS$
input: Interest point $x$
input: Distance measure $d(\vec{x}_1, \vec{x}_2) \rightarrow ||\vec{x}_1 - \vec{x}_2||$
 1: Initialize selected points from the archive $S := \emptyset$
 2: Initialize candidate points to be sampled $X := \emptyset$
 3: $R = \{r_1, ..., r_m\} := Sampler([\vec{x} - \delta, \vec{x} + \delta], m)$
 4: for $i := \{1, ..., m\}$ do
 5:     $(\vec{x}_a, f_a) := ClosestPoint(A, \vec{r}_i, d) \rightarrow head\{(\vec{x}_j, f_j) | (\vec{x}_j, f_j) \in A, \forall(\vec{x}', f') \in A : d(\vec{x}_j, \vec{r}_i) \leq d(\vec{x}', \vec{r}_i)\}$
 6:     if $(\vec{x}_a, f_a) = ClosestPoint(R, \vec{x}_a, d)$ then
 7:         $S := S \cup \{(\vec{x}_a, f_a)\}$
 8:     else
 9:         $X := X \cup \{\vec{r}_i\}$
10:     end if
11: end for
output: Selected archive points $S$ and candidate points $X$
```

This computation can be further improved for increased accuracy of approximation by using all the available and usable values in the archive A_u. The computation of the effective fitness would change to

$$f_{eff}(\vec{x}) = \frac{\sum_{\vec{x}' \in X \cup S \cup A_u} w(\vec{x}') \cdot f(\vec{x}')}{\sum_{\vec{x}' \in X \cup S \cup A_u} w(\vec{x}')}, \qquad (15)$$

where A_u are all archive points from an area of interest I and $w(x') \sim pdf(\vec{\delta})$ is a weighting function, that weighs the importance of an archive point in the effective fitness calculation according to the problem's input fluctuations. This formula is similar to the one shown in [4].

If the input noise follows a uniform distribution and the weighting function is defined as

$$w(\vec{x}') = \begin{cases} 1, & \text{if } \vec{x}' \in [\vec{x} - \vec{\delta}, \vec{x} + \vec{\delta}] \\ 0, & otherwise \end{cases}, \qquad (16)$$

then the function for calculating the effective fitness simplifies to

$$f_{eff}(\vec{x}) = \frac{1}{|S \cup X \cup A_u|} \sum_{\vec{x}' \in X \cup S \cup A_u} f(\vec{x}'). \qquad (17)$$

Since the data structure used for the archive is a list[2], some modifications to improve handling of the list have been made: The archive has a limited size $A_{max} = 5000$ to keep run time computation for list adds and look-ups at an acceptable level. To prevent the archive from cluttering up and to keep up the performance of its list operations a clean-up procedure is employed, which is shown in Algorithm 5. Entries that have not been used for a long time are removed from the archive. The motivation of doing this is the convergence behavior of the PSO algorithms. Since the PSO algorithms converge towards areas that they find interesting, areas that do not seem interesting will not be visited again with a very high likelihood. Hence, these old and (with high probability) uninteresting points are erased from the archive. To determine which archive entries have not been used for a long time, a lifespan counter for each entry is used. Whenever an entry gets added to the archive or when it gets used in a MEM approach, the entry's lifespan counter is set to a maximum lifespan value ls_{max}. After each robust fitness evaluation the lifespan counter for each entry is decremented. Lastly, entries first added to the archive are removed, if the maximum archive size has been breached.

[2]For the archive implementation other data structures are also possible of course, like trees which could be more efficient for several operations like archive look-ups and writes.

Algorithm 5 Archive clean-up procedure

```
input: Archive $A$
input: Maximum archive size $A_{max}$
input: Archive entry lifespans $LS$
1: for each archive entry $i \in A$ do
2:     if $LS(i) < 0$ then
3:         $A := DeleteEntry(A, i)$
4:     end if
5: end for
6: while $size(A) > A_{max}$ do
7:     $A := RemoveFirstEntry(A)$
8: end while
output: Cleaned-up archive $A$
```

As with the MEM approach, the formulas and the calls to the selection scheme and clean-up procedure would be used in the fitness evaluation functions/parts $EvaluateFitness(*)$ of the PSO algorithms.

3.2.4 Metamodels

To further reduce the number of FFE, it is possible to integrate a metamodel into the algorithms to predict fitness function values, using previously found data. Meta-models have been successfully deployed in evolutionary algorithms, using various techniques such as neural networks, kriging and regression. Metamodels can be used to complement the archiving approach [13]. However, they will not be considered in this study.

4. EXPERIMENTS

To investigate the usability of PSO algorithms in finding robust optima, several tests, which use selected (multi-dimensional) test functions, have been devised and executed.

For each algorithm three main test groups were run. The first test group, called **sampling**, is used to investigate whether the sampling or MEM approach for finding robust optima is generally usable with PSO algorithms. The second group, called **archiving**, is used to investigate two things: One, whether an archive can help in reducing the number of FFE, while retaining search accuracy; and secondly, whether an archive can even be used to increase the accuracy of robust fitness value approximation. The third test group, called **particle behavior**, is there to help get an idea of the behavior of the algorithms by investigating how the particles move on the contours of the test functions.

The results of the test groups are shown and discussed one by one under different modifications and settings. At the end a comparison between the tested algorithms is made to find out which algorithms were the most promising.

The raw and metamodel methods were not handled in the experiments.

4.1 General settings

In the experiments four test functions TP1 to TP4 were used. Each experiment has an evaluation budget of 3000 effective fitness evaluations and is run 30 times to somewhat ensure statistical significance. For the sampling and archiving sections of the experiments, 5-dimensional versions of the test functions were used. For the investigation of the particle behaviors of the PSO algorithms, 2-dimensional versions of the test functions were used instead, because these are easier to draw and comprehend, but still give a clue on how the algorithms operate. For the particle behavior investigation only the pure sampling based algorithms are considered.

The estimation of the robust fitness is done via the approximation of the effective or expected fitness f_{eff}, using $m = 10$ samples for the approximation. The coefficients $\phi_1 = 2.8$ and $\phi_2 = 1.3$ are set so that $\phi_1 + \phi_2 \geq 4$ [14, 12]. To ensure that the algorithms optimize within the bounds of valid search spaces, a cut-off boundary constraint handler $g(\vec{x}, \vec{lb}, \vec{ub}) = max(min(\vec{x}, \vec{ub}), \vec{lb})$ is used, where $\vec{x}$ is the variable to be manipulated, and $\vec{lb}$ and $\vec{ub}$ are lower and upper bounds. Variables like particle positions and velocities are initialized as shown in algorithm 1. In case an archive is being used, the maximum size an archive is allowed to have is $A_{max} = 5000$ samples, where each sample is only allowed to not be used for a maximum of $ls_{max} = 100$.

Presented are plots which show the performance of the algorithms and the costs involved via the relationship of fitness function calls versus effective fitness function calls. Data tables contain standard statistical information like averages, standard deviations, medians and also ranks and rank sums. For the ranking and rank sums a method as used in [10] is employed. For the produced graphs and data tables an a posteriori estimation of the real effective fitness values was used via the MEM approach with 200 samples. Among others, the resulting graphs display fitness value developments, as well as the (approximate) fitness value positions of the real robust optima (via constant lines).

Table 1 summarizes the general settings used.

Test functions	TP1, TP2, TP3, TP4
Problem dimension	5, 2
Runs	30
Effective fitness evaluations	3000
Number of samples m	10
Robust fitness estimation	f_{eff}
Boundary handler	Cut-off
Initialization	random
Personal best coefficient ϕ_1	2.8
Neighbor best coefficient ϕ_2	1.3
Maximum archive size A_{max}	5000
Maximum lifespan ls_{max}	100
Number of particles	30, 100

Table 1: General settings

4.2 Test functions

The test functions used in the tests can be categorized by using the relationship between the (global) non-robust optimum and robust optimum in a fitness landscape as was shown in [13, 9] . Four different categories of robustness for functions have been defined in the literature:

1. **Identical optimum**: non-robust optimum and robust optimum are identical.
2. **Neighborhood optimum**: non-robust optimum and robust optimum are on the same peak (from the perspective of the robust fitness), but their exact position is not the same.
3. **Local-global-flip**: a local optimum is the local robust optimum, while the global optimum is not the global robust optimum.
4. **Max-min-flip**: the global robust maximum/minimum is located at a non-robust minimum/maximum.

Category 1 problems are not really interesting, because for this problem type no specialized algorithm for robust optima is necessary. Hence, four test problems (TP1 to TP4), that cover the categories two to four, have been used in the experiments[3]. They are minimization problems. The test functions TP1 and TP2 are unimodal optimization problems, meaning that they have only one optimum in the search space. The TP3 and TP4 test functions on the other hand are multimodal optimization problems, as they house more than one optimum.

Detailed information about the used test problems is covered in the following page. Graphs and solutions were computed using a high-precision MEM approach with 20000 samples.

[3]Note, that in practice it is often not known what kind of category a problem is in.

TP1

$$f(\vec{x}) = \left(1 - \prod_{i=1}^{N} H(x_i)\right) + \frac{1}{100}\sum_{i=1}^{N} x_i^2 \tag{18}$$

$$H(x_i) = \begin{cases} 0 & x_i < 0 \\ 1 & otherwise \end{cases} \tag{19}$$

Search interval: $\vec{x} \in [-10, 10]^N$
Input noise: $\vec{\delta} \sim \vec{U}(-1, 1)$
Type(s): Category 2
Robust optimum value: ≈ 0.024 (2D), ≈ 0.06 (5D)
Robust optimum location: $(1, 1)$ (2D), $(1, 1, 1, 1, 1)$ (5D)

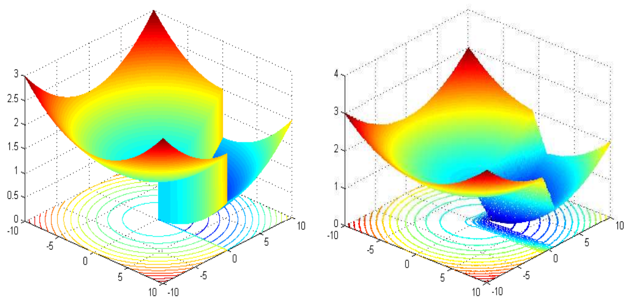

(a) fitness landscape (b) robust fitness landscape

Figure 5: TP1

TP2

$$f(\vec{x}) = \sqrt{||\vec{x}||} + e^{-5||\vec{x}||^2} \tag{20}$$

Search interval: $\vec{x} \in [-4, 4]^N$
Input noise: $\vec{\delta} \sim \vec{U}(-0.5, 0.5)$
Type(s): Category 2 and 4
Robust optimum value: ≈ 1.1 (2D), ≈ 0.96 (5D)
Robust optimum location: $(0, 0)$ (2D), $(0, 0, 0, 0, 0)$ (5D)

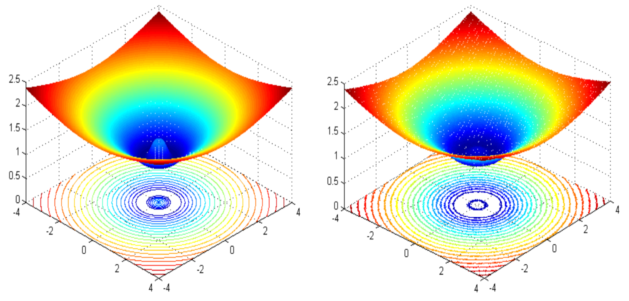

(a) fitness landscape (b) robust fitness landscape

Figure 6: TP2

TP3

$$f(\vec{x}) = \frac{5}{5-\sqrt{5}} - max\{f_0(\vec{x}), f_1(\vec{x}), f_2(\vec{x}), f_3(\vec{x})\} \tag{21}$$

$$f_0(\vec{x}) = \frac{1}{10} \cdot e^{\frac{1}{2}\cdot||\vec{x}||} \tag{22}$$

$$f_1(\vec{x}) = \frac{5}{5-\sqrt{5}} \cdot \left(1 - \sqrt{\frac{||\vec{x}+5||}{5\sqrt{N}}}\right) \tag{23}$$

$$f_2(\vec{x}) = c_1 \cdot \left(1 - \left(\frac{||\vec{x}+5||}{5\sqrt{N}}\right)^4\right) \tag{24}$$

$$f_3(\vec{x}) = c_1 \cdot \left(1 - \left(\frac{||\vec{x}-5||}{5\sqrt{N}}\right)^{d_2}\right) \tag{25}$$

$$c_1 = \frac{625}{624}, c_2 = 1.5975, d_2 = 1.1513 \tag{26}$$

Search interval: $\vec{x} \in [-10, 10]^N$
Input noise: $\vec{\delta} \sim \vec{U}(-1, 1)$
Type(s): Category 3
Robust optimum value: ≈ 0.33 (2D), ≈ 0.34 (5D)
Robust optimum location: $(5, 5)$ (2D), $(5, 5, 5, 5, 5)$ (5D)

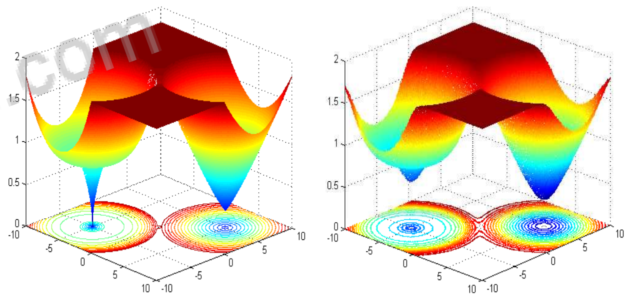

(a) fitness landscape (b) robust fitness landscape

Figure 7: TP3

TP4

$$f(\vec{x}) = c - \frac{1}{N}\sum_{i=1}^{N} f_1(x_i) \tag{27}$$

$$f_1(x_i) = \begin{cases} -(x_i+1)^2 + 1 & -2 \le x_i < 0 \\ c \cdot 2^{-8|x_i-1|} & 0 \le x_i < 2 \end{cases} \tag{28}$$

$$c = 1.3 \tag{29}$$

Search interval: $\vec{x} \in [-2, 2]^N$
Input noise: $\vec{\delta} \sim \vec{U}(0.5, 0.5)$
Type(s): Category 3
Robust optimum value: ≈ 0.37 (2D), ≈ 0.37 (5D)
Robust optimum location: $-(1, 1)$ (2D), $-(1, 1, 1, 1, 1)$ (5D)

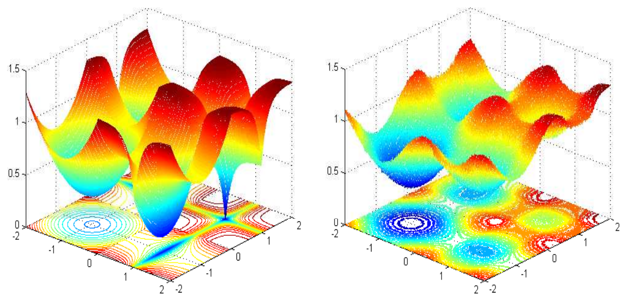

(a) fitness landscape (b) robust fitness landscape

Figure 8: TP4

4.3 CPSO

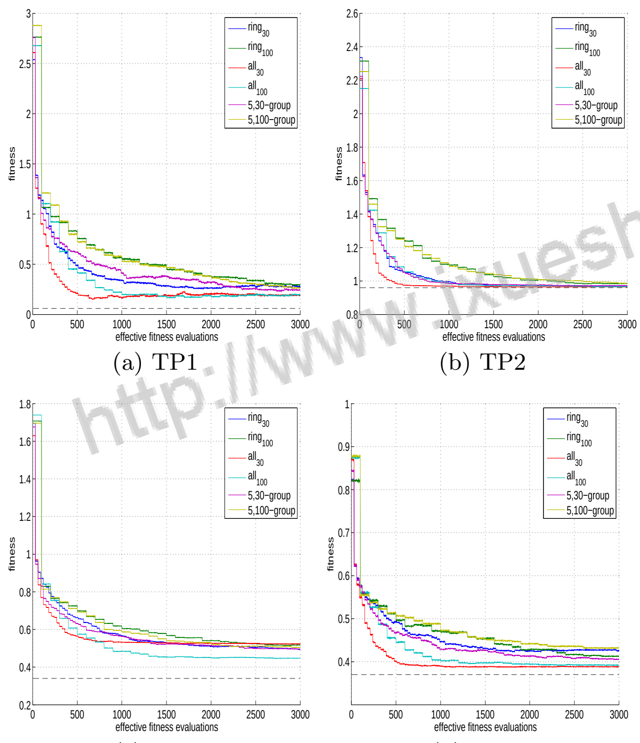
4.3.1 *Sampling*

As Table 2 shows, in this test group six tests were conducted: two tests used the all topology with 30 and 100 particles, another two tests used the ring topology with 30 and 100 particles and finally the last two tests used the 5,n-group topology with $n = 30$ and $n = 100$ particles.

Number of particles c	30, 100
Topologies T	all, ring, m,n-group

Table 2: Experimental settings (CPSO - Sampling)

An overview of the results that have been produced for this algorithm is given by Figure 9, which displays the development of the average fitness on the test functions TP1 to TP4. A more detailed look at the results is given in Figures 26, 28, 30 and 32, which show the performance of the algorithm relative to the number of evaluations, using a time series of standard box-plots. Figure 58 shows boxplots of the end results from the different settings used. Tables 9 and 10 show numerical results in the form of means, standard deviations, medians, rank sums and ranks. The tables and figures may also show data on the archive versions of the algorithms for easy comparison.



(a) TP1 (b) TP2

(c) TP3 (d) TP4

Figure 9: Fitness means (CPSO - Sampling)

From these results it is possible to make some definite statements:

- The algorithm was able to optimize in all four test cases with all of the given settings.
- The all topology generally worked the best from a solution quality standpoint based on median, average and standard deviation. The group topology came in second and the ring topology had the worst performance. This also holds when the test functions are viewed from a unimodal or multimodal standpoint. The difference of quality could be attributed to the amount of information, that is available between the particles and which is defined by the communication topology, which the particles use. In the all topology, all information between the particles is available, while in the group and ring topologies information is capsuled more locally in their sub-groups. In the ring topology the information a particle has is shared between only two neighboring particles, while with the group topology the information is shared among a group of pre-defined size. Since the latter topologies do not have all their information shared, they tend to explore more and this in turn tends to create more diverse solution paths during the test runs, which can be best seen from the standard deviation values.
- There does not seem to be a benefit from a higher number of particles evaluation wise. The data shows, that generally the 100 particle test runs generate end results that are on par with the 30 particle test runs or worse. Looking at the convergence speed, it can be observed that the performance is worse for the 100 particle test runs, than for the 30 particle test runs.
- The box plots show, that the algorithm is indeed able to definitely converge towards the true robust optima on some runs.
- The hardest test problems for the algorithm (in the sense of getting good quality solutions) seem to be the TP1 and TP3 fitness functions, where suboptimal convergence is observed. Very peculiar is that for the TP1 test case, the algorithm showed better results on some settings at the beginning and worsened its results over time. This behavior could be the result of the low number of samples ($m = 10$), that have been used for the approximation of the robust fitness value. The low number of samples could cause the algorithm to make mistakes in generating robust optima approximations, which could cause the algorithm to move towards more unfavorable areas in the search space.
- The low number of samples for the fitness approximation, could be a probable reason why the algorithm's convergence is a little bit off from the actual optima on most test functions.

4.3.2 *Archiving*

For the tests of the CPSO algorithm with archive, three tests were conducted, as shown by Table 3. The same topologies were used but only 30 particles are considered, because the sampling test group showed that a higher number of particles does not seem to give a benefit, when it comes to the number of evaluations. Finally Tables 9 and 10 show some statistical results in the form of means, standard deviations, medians, rank sums and ranks.

Figure 10 gives an overview on the performance of the CPSO algorithm with archiving and Figures 27, 29, 31, 33 and

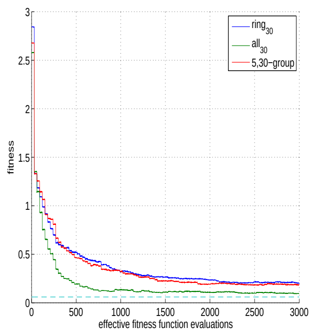
Number of particles c	30
Topologies T	all, ring, (5,30)-group

Table 3: Experimental settings (CPSO - Archive)

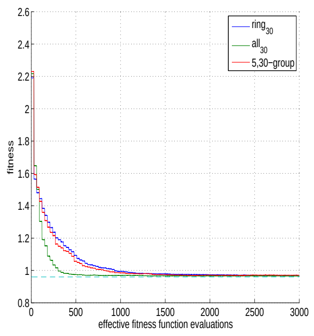
58 show more detailed graphs concerning the algorithm's performance. Figure 11 shows how many fitness function evaluations **(FFE)** were needed to produce the result. The perforated line indicates how many FFE would have been necessary using the standard sampling approach. Tables 9 and 10 present statistical data.

From these results, several things can be concluded:

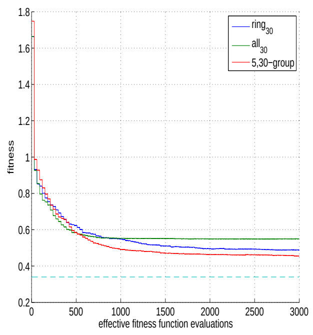
- In general the archive-backed versions of the CPSO algorithm were able to produce better results, than their plain sampling-based counterparts. This also holds true for both unimodal and multimodal test functions.
- The general best topology turned out to be the all topology followed by the group and ring topologies. This holds true for the unimodal test functions as well, but not for the multimodal test functions. The best topology for this problem type turned out to be the group topology. It seems as if this topology type is able to profit greatly from using an archive, as the performance of the group topology with archiving far surpassed the performance of the plain sampling method. Better optima approximations from the archiving approach could explain this phenomenon, as better optima approximations could lead to better group placements.
- The number of fitness function evaluations per effective fitness function call was generally a magnitude less, than what would have been necessary using a standard MEM sampling procedure.
- The all topology seems to make the best use of the archive, being able to use a considerable amount of fitness function evaluations less, than the other two tested topologies. An explanation for this could be, that the all topology converges to an area of the search space faster. This would mean that many particles would be concentrated in one area of the search space, which would then have the effect of (mainly) filling the archive with points of that area. Since the particles have probably stopped exploring the rest of the search space by that time, they could then make full use of the archive without needing to add more values to the archive, which would result in a saving of fitness function calls. An analog explanation can be used to explain why the ring and group topologies use around double as many fitness function calls. Since on these topologies the particles tend to explore more it makes sense, that the archive will have less usable entries. Hence, there is a bigger need of making fitness function calls.



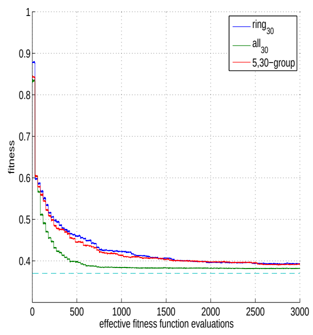
(a) TP1



(b) TP2

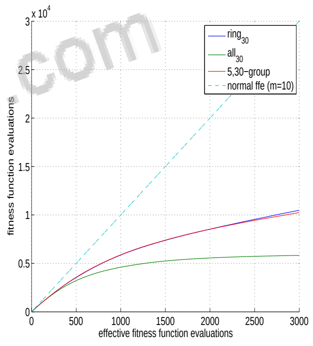


(c) TP3

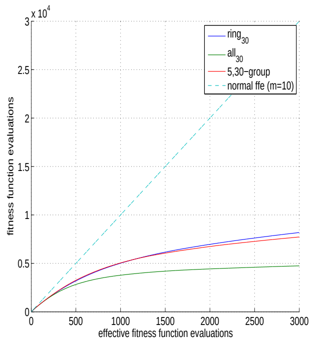


(d) TP4

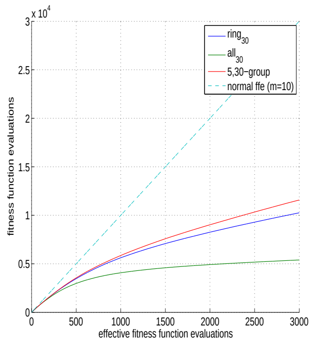
Figure 10: Fitness means (CPSO - Archive)



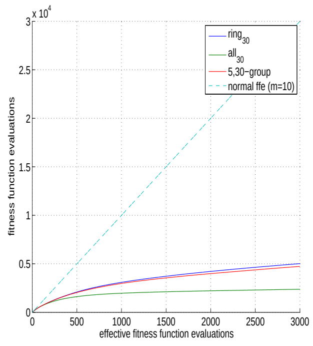
(a) TP1



(b) TP2



(c) TP3

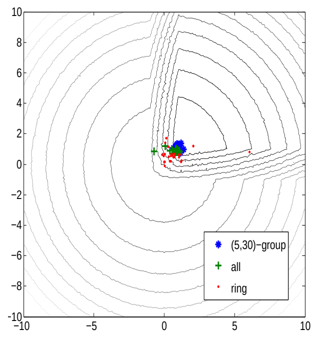


(d) TP4

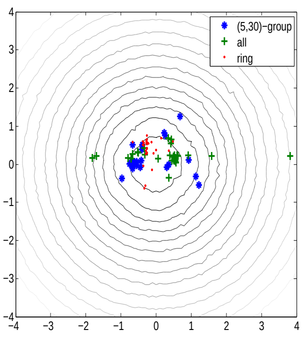
Figure 11: Mean FFE (CPSO - Archive)

4.3.3 *Particle behavior*

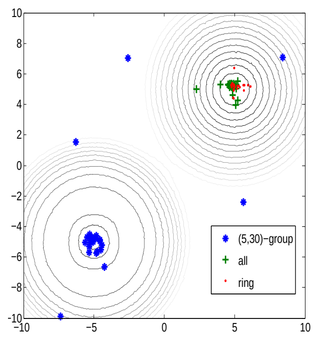
To get a feel of how the CPSO algorithm operates, a look at the particle behavior is made by investigating how the particles have 'converged' at the end of some sample runs for the sampling versions of the algorithm. This is done for all the test functions under dimension 2, using all of the topologies with 30 particles, as shown by Figure 12.



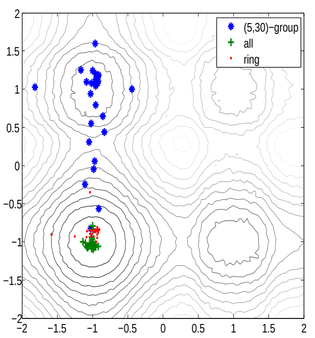
(a) TP1



(b) TP2



(c) TP3



(d) TP4

Figure 12: Particle behaviors (CPSO)

From these graphics, it can be seen that the CPSO algorithm generally converges towards one small area of the search space with some spreading of its particles here and there. The ring topology seems to behave similarly with more spreading or exploration of the particles. The group topology shows the most spreading and seems to explore the most of the tested topologies. TP3 and TP4 show that with the group topology, the algorithm moves towards several different optima, which could be beneficial for finding better robust optima. However, it can also be seen that the group topology seems to mostly concentrate its particles at the wrong peaks. This could be the reason why it on average produced mediocre results for the TP3 test function. It seems as if there might have been an error in the effective fitness approximation, which led to the algorithm moving towards the wrong optima. It seems this problem may be due to the low number of samples used for the approximation of the effective fitness. This idea is backed up by the observation that the archiving version generally generated better approximations and does not seem to usually have the same problem of getting confused and converging towards the wrong optima. Something else that is noticeable on the TP3 test function for the group topology is that several particles were moving outside of the peaks. These particles were mostly leader particles, which were exploring.

4.4 FIPS

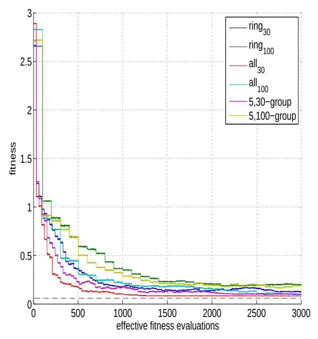
4.4.1 *Sampling*

For the sampling tests of the FIPS algorithm, the same settings as for the CPSO algorithm were used to make a fair comparison between the two. Table 4 summarizes the settings used.

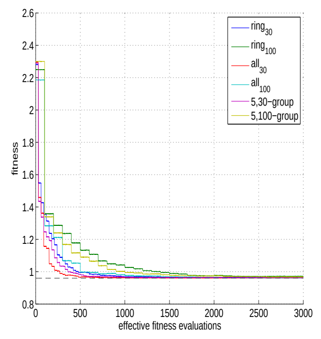
Number of particles c	30, 100
Topologies T	all, ring, m,n-group

Table 4: Experimental settings (FIPS - Sampling)

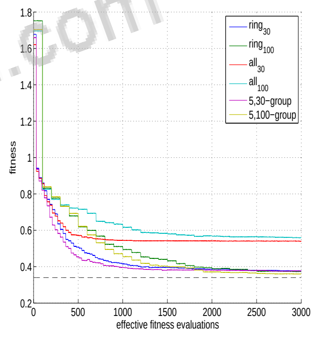
Figure 13 gives an overview on the performance of the FIPS algorithm with archiving, and Figures 34, 36, 38, 40 and 59 show more detailed graphs concerning the algorithm's performance. Tables 11 and 12 present statistical data.



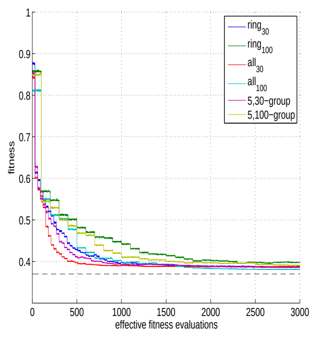
(a) TP1



(b) TP2



(c) TP3



(d) TP4

Figure 13: Fitness means (FIPS - Sampling)

From these results, several things can be concluded:

- Under all used topologies and amounts of particles, FIPS was able to show good performance.
- The best topology overall for this algorithm seems to be the all topology followed by the group and the ring topologies. For unimodal problems the group and the ring topologies also the best, while for multimodal problems it seems that the group topology followed by the ring topology work best. This becomes apparent in the multimodal TP3 test function, where the ring and group topologies greatly outperform the all topology.

- A high number of particles does not seem to generally give the algorithm an edge. Convergence speed seems to slow down generally. However, in some special cases a large number of particles did seem to have a positive impact on performance. This can be observed for example on test function TP1, where the all topology using 100 particles showed the best average result.

4.4.2 *Archiving*

As Table 5 shows, for the archiving tests of the FIPS algorithm, the same settings as for the sampling runs were used to make a fair comparison between the two. The tests with 100 particles have been removed, as the high number count does not seem to help generally.

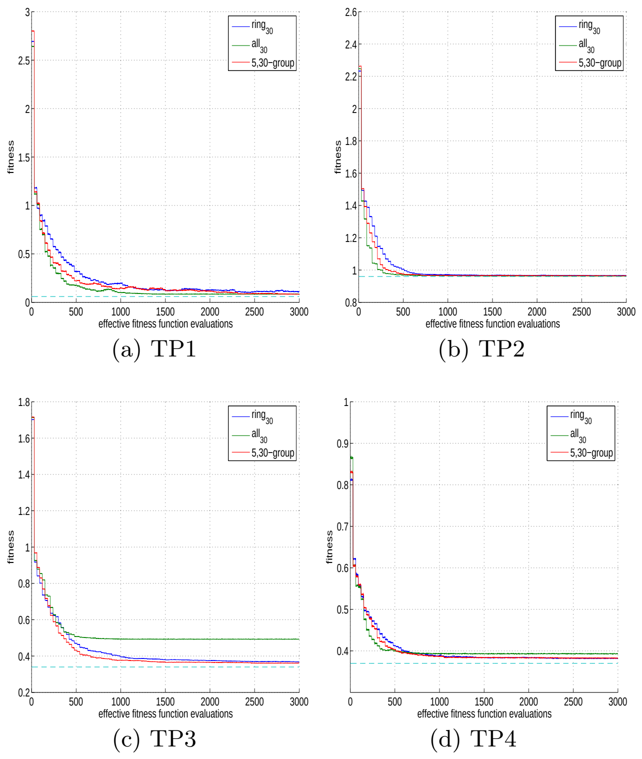
Number of particles c	30
Topologies T	all, ring, m,n-group

Table 5: Experimental settings (FIPS - Archive)

An overview of the algorithm's performance is given in Figure 14. Figures 35, 37, 39, 41 and 59 give detailed boxplots for the performance. Figure 15 shows the related average number of fitness function evaluations in relation to the number of effective fitness function calls. Tables 11 and 12 present statistical data.

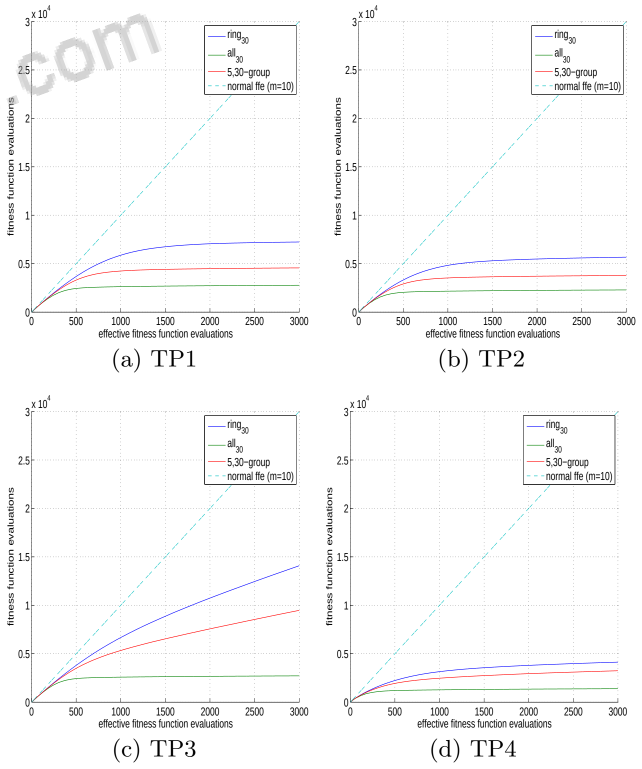
From these results it can be concluded that:

- The archive-based FIPS were generally able to produce good optimization results. For the ring and group topologies, the archive-based versions were able to produce better results than the sampling-based versions. Interestingly, for the all topology the pure sampling-based FIPS ran similarly well compared to its archive-based counterpart.
- The archive was able to significantly reduce the number of fitness function evaluations per effective fitness function call, while retaining search accuracy. Additionally, an increase in accuracy could generally be observed.
- The all topology seems to be using the archive best, resulting in the lowest number of fitness function evaluations among the tested topologies. The ring and group topologies tend to need around double or more fitness function evaluations compared to the all topology. The difference in needed fitness function evaluations is even more severe and apparent in the TP3 test function. However, this sacrifice in fitness function evaluations seems to be able to result in better fitness values in some cases as again seen in the TP3 test results.
- The generally best-performing topology in terms of optimization accuracy is the group topology followed by the all and ring topologies. However, the other topologies also did very well and the differences in performance are not very large. The main reason on why the group topology beat the all topology, seems to be because it was much stronger in the multimodal problems as can be very well seen in the the TP3 test function results.



(a) TP1 (b) TP2
(c) TP3 (d) TP4

Figure 14: Fitness means (FIPS - Archive)

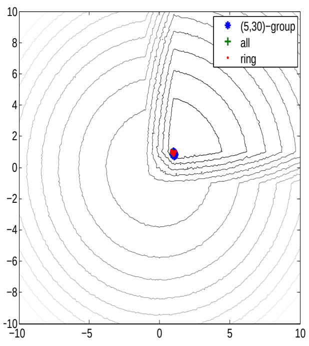


(a) TP1 (b) TP2
(c) TP3 (d) TP4

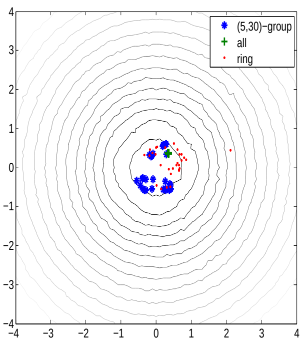
Figure 15: Mean FFE (FIPS - Archive)

4.4.3 *Particle behavior*

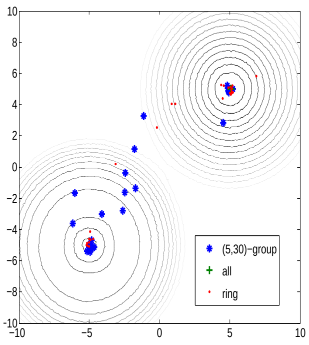
Sample behaviors of the particles under the FIPS algorithm on 2D versions of the test functions are given in Figure 16.



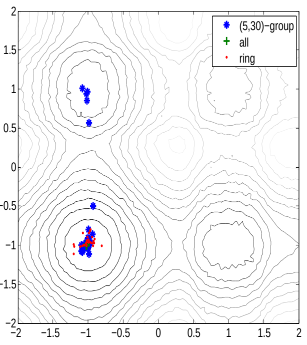
(a) TP1



(b) TP2



(c) TP3



(d) TP4

Figure 16: Particle behaviors (FIPS)

The plots show that the all topology generally concentrates its particles in a small area of the search space. This could be a reason on why the all topology works so well on unimodal problems. The strong focus towards a small search space seems to be beneficial in converging towards high quality optima approximations. The ring and group topologies show more spreading and exploration in their particle behavior. This spreading seems to be the reason why the group and ring topologies are able to function better on multimodal problems, as the particles are more likely to reach different optima and thus the probability of getting stuck in sub-optimal regions gets reduced. This spreading or exploration behavior also explains the massive difference in fitness function evaluations needed between the all and the ring and group topologies when an archive is used. An explanation of this could be that the greater exploratory behavior would mean that the probability of reaching yet unseen areas of the search space would increase. This in turn would increase the likelihood of the archive not having usable entries for the effective fitness approximation, which would lead to more fitness function calls needing to be made.

4.5 CCPSO

4.5.1 *Sampling*

For the sampling tests of the CCPSO algorithm 30 particles and the topologies ring, all and group are used. Also some algorithm specific parameters are defined:

- The core radius R_c is here set to 1 to allow sufficient protection against velocity explosions.
- The perception radius R_p is set to the maximum possible input fluctuation value $max(\vec{\delta})$. The idea of using this value is to spread out the charged particles far enough that they would not get stuck into a local optimum.
- The charge q between each repelling pair of particles is set to 1.
- As charge topology, the atomic charge topology is being used, because it was found that it generally performed best on dynamic problems[7, 3]. Also this setup seems very reasonable, as it would utilize one half of the population for exploration and the other half for exploitation. If instead a fully or randomly charged topology was used, then the algorithm might for example be good in exploration, but not in exploitation. This would have the effect of the algorithm not properly converging towards an optimum.

Table 6 summarizes the settings used.

Number of particles c	30
Topologies T	all, ring, m,n-group
Core radius R_c	1
Perception radius R_p	$max(\vec{\delta})$
Charge q	1
Charge topology Q	atomic

Table 6: Experimental settings (CCPSO)

Figure 17 gives an overview on the performance of the CCPSO algorithm with archiving, and Figures 42, 44, 46, 48 and 60 show more detailed graphs concerning the algorithm's performance. Tables 13 and 14 present statistical data. From these results it can be said that:

- The CCPSO algorithm was able to optimize under all settings used.
- The best general performance was observed from the all topology. This holds true for both the unimodal and multimodal problems. The group and ring topologies worked similarly well and were able to best the all topology on the TP3 test function.

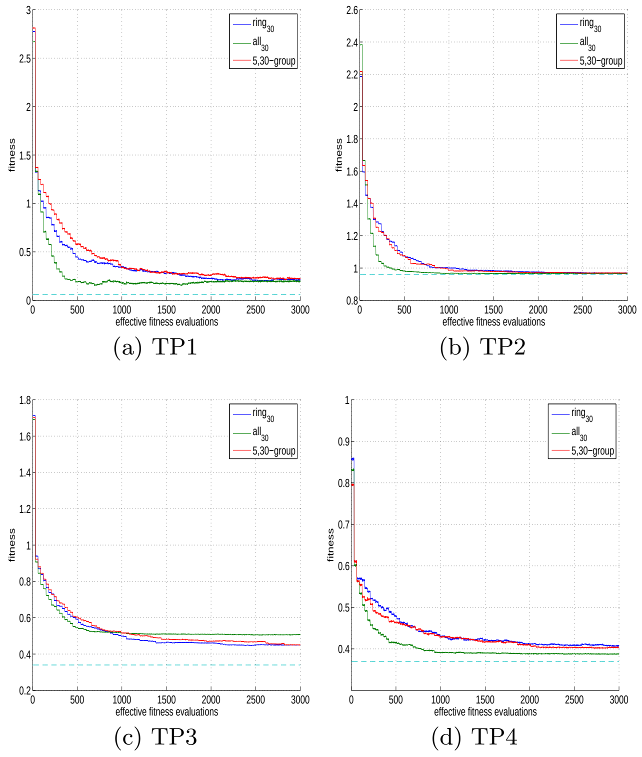


Figure 17: Fitness means (CCPSO - Sampling)

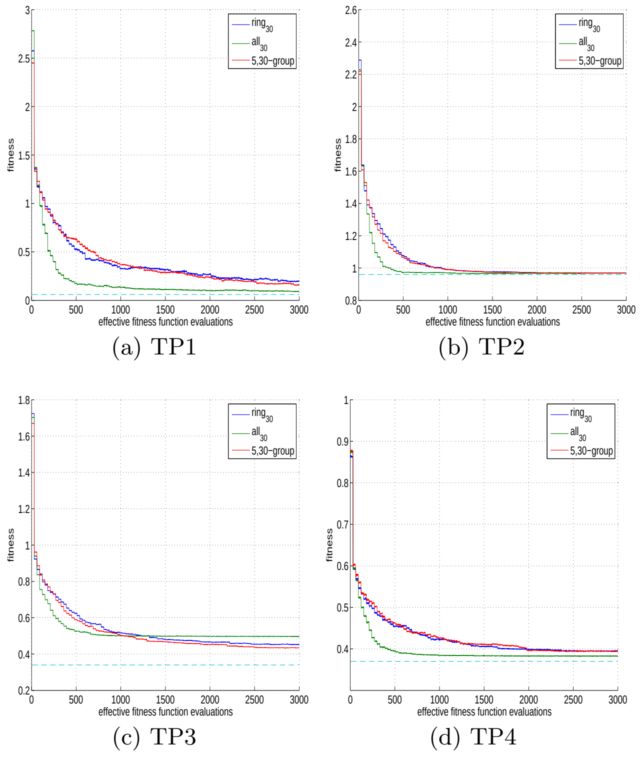


Figure 18: Fitness means (CCPSO - Archive)

4.5.2 Archiving

For the archiving tests, the same settings from the sampling tests are used, which are shown in Table 6.

Figure 18 gives an overview on the performance of the CPSO algorithm with archiving, and Figures 43, 45, 47, 49 and 60 show more detailed graphs concerning the algorithm's performance. From these results it can be said that:

- The archiving versions of the CCPSO algorithm seem to work better than the sampling versions of the algorithm on all settings used.
- A lot less fitness function evaluations were needed for reaching similar sampling results and the all topology seems to have best used the archive, resulting in the least recorded number of fitness function evaluations.
- Similarly to the pure sampling test runs, the all topology generally produced the best results. The group and ring topologies produced similar results and again bested the all topology on the TP3 test function slightly, but fell short on the others. Between the group and ring topologies, the group topology produced slightly better results, but also needed slightly more fitness function evaluations.

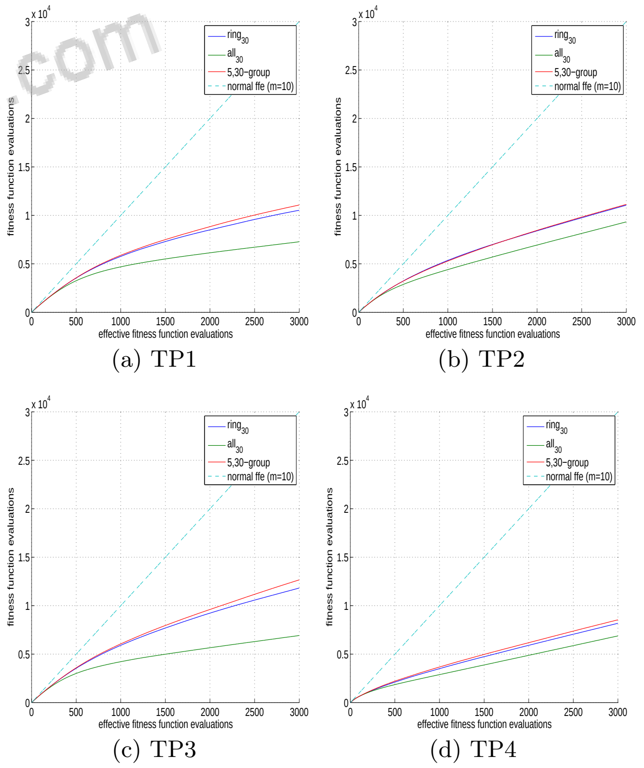
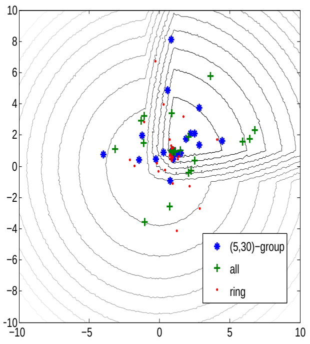


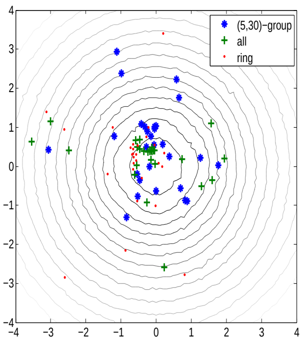
Figure 19: Mean FFE (CCPSO - Archive)

4.5.3 *Particle behavior*

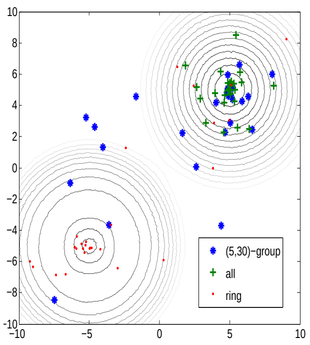
Sample behaviors of the particles under the CCPSO algorithm on 2D versions of the test functions are given in Figure 20.



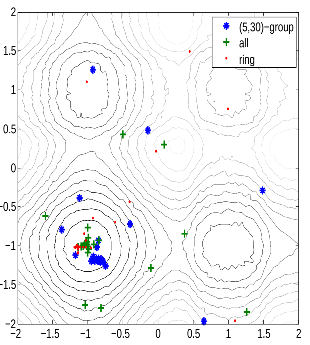
(a) TP1



(b) TP2



(c) TP3



(d) TP4

Figure 20: Particle behaviors (CCPSO)

From these graphs it can be said that the CCPSO algorithm indeed does what was expected of it. Part of its particles seem to be engaged in exploratory behavior and are spread across the search space seemingly randomly, but not too far from each other. The other part of its particle base seems to be more or less clustered at around an optima. Comparing the topologies used, it seems as if the ring and group topologies show the most exploration behavior, while the all topology seems to prefer clustering more.

An interesting observation of the exploring particles is that they seem a little isolated and uncoordinated. This could lead to problems when an exploring particle is on the right track, but there are too few other particles following it. This in turn could lead to the swarm not acknowledging the more suitable area. The consequence of this is, that the algorithm might get confused and might find a less robust optimum, than it could have. Another problem that seems to arise from the lack of coordination in the exploring particles is that they might explore areas, that are completely fruitless and only driven there by the urge of not being near another charged particle. This could lead to lots of potentially lost computational time and bad archive filling, if an archive is in use. The behavior plot of test function TP3 shows this well, where some of the exploring particles seem to be moving far in the "flat lands" of the search space.

4.6 MPSO

4.6.1 *Sampling*

For the sampling tests of the MPSO algorithms, the following settings were used:

- A population size of 30 was used and higher population sizes were not investigated into, because previous tests on other PSO algorithms have generally not shown that they can be beneficial.
- The topologies ring and all were tested and the group-topology was ignored, because the algorithm already provided grouping, so additional grouping via a special topology did not seem very prosperous.
- The merging distance of two different swarms was set to $d_{max} = 0.001$.
- Five percent was set as the sub-swarm creation threshold ϵ for an an observation window $w = 50$.
- For the sub-swarm particle training the FIPS algorithm was used as their full model PSO.

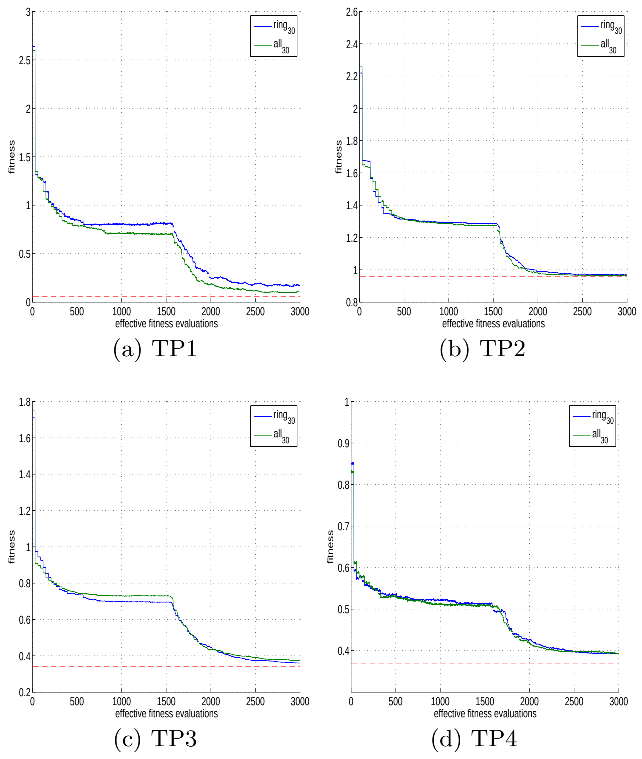
Table 7 summarizes the settings.

Number of particles c	30
Topologies T	all, ring
Max distance d_{max}	0.001
Merging threshold ϵ	0.05
Observation window w	50
Full model PSO	FIPS

Table 7: Experimental settings (MPSO)

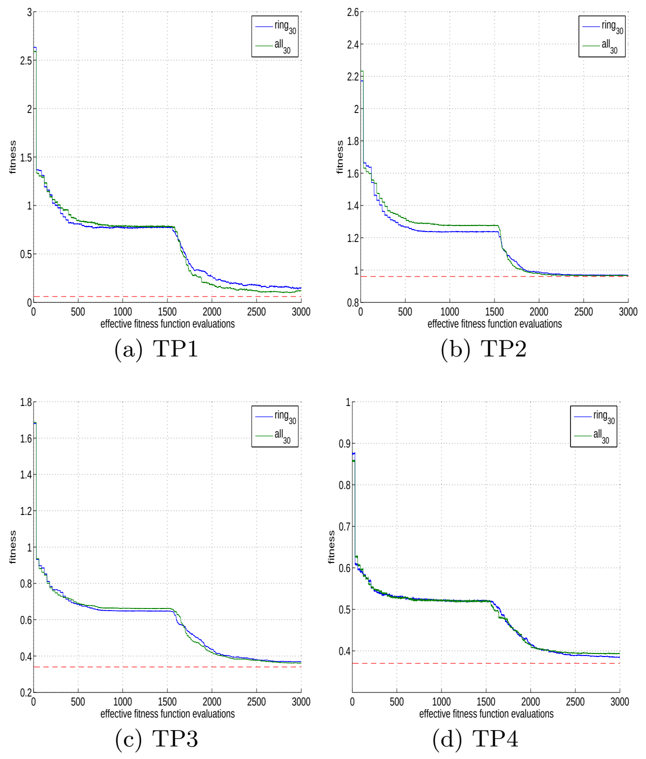
Figure 21 gives an overview on the performance of the algorithm, and Figures 50, 52, 54, 56 and 59 show more detailed graphs concerning the performance. Tables 15 and 16 present statistical data. From these results it can be said that:

- The MPSO algorithm has produced very good results overall under all settings used.
- Overall it seems that the all topology worked best for the algorithm. This could be explained by the groups actually doing local searches and having all the data from the neighbors available is certainly helpful for that, as was shown in [12]. However, the difference in performance between the two topologies does not seem very big.
- Convergence behavior seems to be divided into two phases, which can be clearly seen from the overview graphs. The first part seems to depict the convergence of the main swarm, which only uses the cognitive and momentum components of the standard velocity update function. The second part depicts the convergence behavior of the best sub-swarm, which is available. Since it uses a full PSO model, it is able to generate much better results.



(a) TP1 (b) TP2 (c) TP3 (d) TP4

Figure 21: Fitness means (MPSO - Sampling)



(a) TP1 (b) TP2 (c) TP3 (d) TP4

Figure 22: Fitness means (MPSO - Archive)

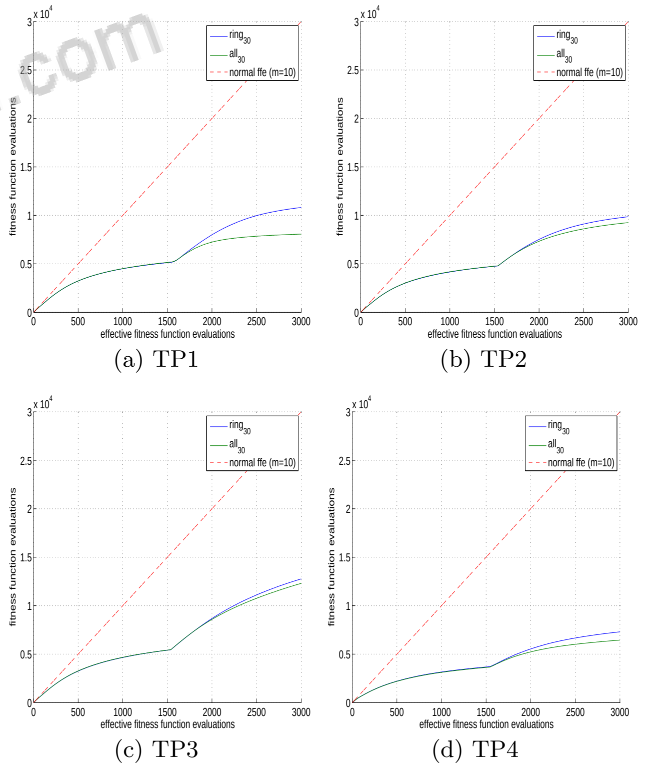
- Very noticeable is that the end point of the first phase and the starting point of the second phase have a very sharp transition and that the point of transition is very stable under the 30 test runs made. Also this point seems to be at a similar position for all test functions. This leads to the assumption that the particles need a finite time to stabilize their fitness values and positions in the search space and this in turn would mean that by using more optimal settings for ϵ and w, the duration of the first phase could be reduced, while retaining overall accuracy. In the first phase the particles are basically stochastic hill climbers.

4.6.2 *Archiving*

For the archiving tests, the same settings from the sampling tests are used, which are shown in Table 7.

Figure 22 gives an overview on the performance of the algorithm, and Figures 51, 53, 55, 57 and 59 show more detailed graphs concerning the performance. Tables 15 and 16 present statistical data. From these results it can be said that:

- The archiving version of the algorithm performed similarly well to the plain sampling version of the algorithm, while using a lot less fitness function evaluations.
- From a solution quality standpoint it seems that here also the all topology worked better in general compared to the ring topology.



(a) TP1 (b) TP2 (c) TP3 (d) TP4

Figure 23: Mean FFE (MPSO - Archive)

- The all topology seems to work best with the archive, using generally the least fitness function evaluations. Although, the amount of fitness function evaluations saved seems highly dependent on the problem, which becomes apparent from looking at the results of TP3 and TP4.

4.6.3 Particle behavior

Sample behaviors of the particles under the MPSO algorithm on 2D versions of the test functions are given in Figure 24.

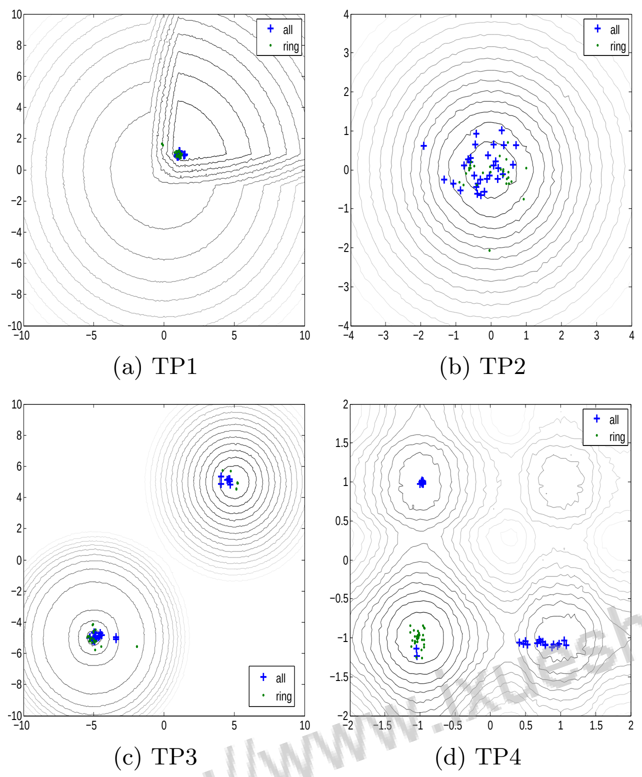


Figure 24: Particle behaviors (MPSO)

The plots show, that generally several swarms were created and they converged independently from one another. This is nicely shown on the TP3 and TP4 test functions, where it could be observed that under both used topologies the algorithm converges towards the best optima.

The particles of the sub-swarm seem to converge more "tightly" and the overall exploration behavior looks orderly, which can be an indicator for good optimization quality, as was previously also noted with the FIPS algorithm.

4.7 Overview and Comparison

For brevity's sake only the two best settings from each archive-based PSO algorithm (MPSO, CCPSO, CPSO, FIPS) were used for comparison with each algorithm using a swarm-size of $c = 30$ particles. The comparison is limited to the the archive-based versions, because they have proven themselves to be generally superior to their plain sampling-based counterparts. For the same reason a swarm-size of 30 particles is preferred versus a swarm-size of 100 particles. The topologies T used by the CCPSO, CPSO and FIPS algorithms for the comparison were the 5,30-group topology and the all topology. The MPSO algorithm used the ring and all topologies instead.

To also get an idea of how well the algorithms from this report compare to other algorithms from the field of robust optimization, they were compared with a state-of-the-art robust evolutionary algorithm at the time of writing. For this purpose the algorithm **Robust CMAES (LHSR)** from the paper [9] was used. Using test functions of dimension $N = 5$, the algorithm ran with an offspring population of $\lambda = 4 + \lfloor 3 \cdot ln(N) \rfloor = 8$ with and a parent population of $\mu = \lfloor \lambda/2 \rfloor = 4$. For approximating the effective fitness, the algorithm used an archive based approach with $m = 10$ samples and an upper limit of 5000 entries for the archive (just like the PSO algorithms). This algorithm was also run 30 times with an effective fitness call budget of 3000 calls (max. 30000 fitness function calls); the same settings which were used for the PSO algorithms.

The results of the comparison in terms of solution quality are presented by Figure 25, which shows the average fitness value development of the algorithms, and Figure 62, which zooms into the fitness at the end of the runs with boxplots. Tables 17 and 18 present statistical data, and Table 8 presents the average cost of each algorithm on the test functions, using the number of fitness function calls needed to generate the results as the relevant measure.

From the solution quality results it can be observed that the best performing algorithm was the FIPS algorithm, followed by the MPSO algorithm, the CCPSO algorithm and the CPSO algorithm. Generally the best topology to use for solution quality was the 5,30-group topology in the case of the CCPSO, CPSO and FIPS algorithms, and the all topology for the case of the MPSO algorithm. The robust CMAES (LHSR) algorithm came in last.

These results show that increased diversification or exploration behavior can indeed be beneficial in finding better solutions from a quality stand-point. It seems that a topology like the 5,30-group, which promotes exploration to a certain degree, can help find better solutions in a multimodal problem setting. On unimodal problems the quality of the solutions were still very good, but overall topologies, which concentrate on exploitation, like the all topology fair a little bit better.

Algorithm	**TP1**	**TP2**	**TP3**	**TP4**	**avg. FFE**	**avg. Saving** $(100 - \frac{\textbf{avg. FFE}}{\textbf{max. FFE}} \cdot 100)$
Rob. CMAES (LHSR)	3492	3437	3451	3357	≈ 3434	$\approx 88.55\%$
MPSO (Archive, ring, 30)	10816	9861	12763	7303	≈ 10185	$\approx 66.05\%$
MPSO (Archive, all, 30)	8068	9254	12304	6451	≈ 9019	$\approx 69.94\%$
CCPSO (Archive, all, 30)	7278	9327	6920	6876	≈ 7600	$\approx 74.7\%$
CCPSO (Archive, group, 30)	11069	11134	12661	8541	≈ 10851	$\approx 63.83\%$
CPSO (Archive, all, 30)	5824	4746	5396	2378	≈ 4686	$\approx 84.38\%$
CPSO (Archive, group, 30)	10245	7725	11578	4729	≈ 8569	$\approx 71.44\%$
FIPS (Archive, all, 30)	2771	2291	2716	1391	≈ 2292	$\approx 92.36\%$
FIPS (Archive, group, 30)	4565	3786	9476	3242	≈ 5267	$\approx 82.44\%$

Table 8: Approx. needed fitness function calls (FFE) and the avg. saving of normally needed FFE

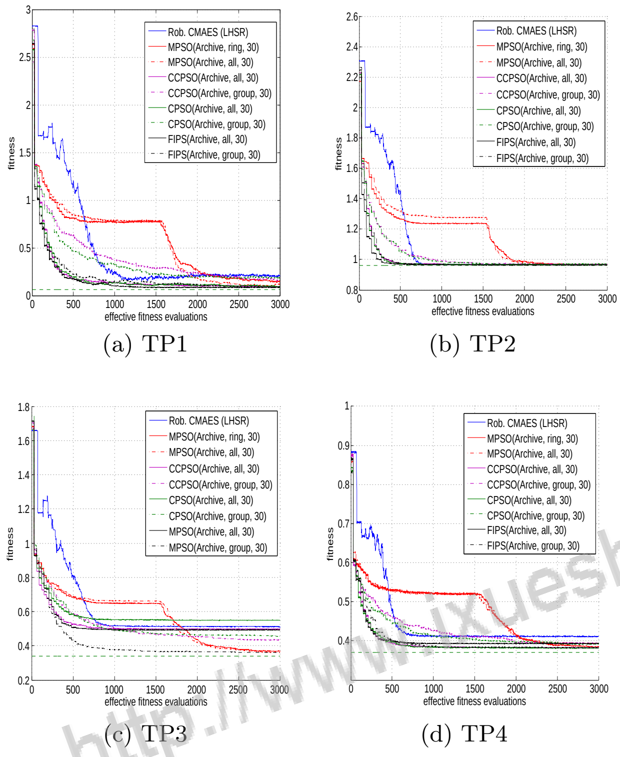


Figure 25: Fitness means (Comparison)

Concerning exploration behavior several more things can be said. The CCPSO algorithm had slightly better performance compared to the CPSO algorithm. That the charged algorithm did not show significant improvements could be attributed towards the settings not being optimal. Another reason for its only slight increase performance could be that the CCPSO algorithm's diversification method is very unorganized and unfocused as shown by the discussion and analysis of the CCPSO algorithm's particle behavior. The MPSO algorithm on the other hand, which focuses on increased exploration behavior through sub-swarming, had a much more focused and organized approach in its diversification method. The result of this is a very competitive algorithm performance wise, which far outdid the CCPSO algorithm. Here again the settings of the algorithm were not optimized, so it could be possible to find better settings for this algorithm still to increase performance.

From the cost results it can be said that the the algorithm, which needed the least number of fitness function evaluations was the FIPS algorithm followed by robust CMAES (LHSR), CPSO, CCPSO and MPSO. A possible explanation on why the CMAES algorithm worked so well in this regard could be the fact, that it used a very low number of individuals compared to the PSO algorithms. In this regard it could be possible to improve the PSO algorithm's performance by further reducing their particle numbers in order to receive a lower number of fitness function calls, while retaining or (better) perhaps improving their solution qualities. Ideally a formula or rule for this would be found. An interesting observation is that it seems that on the long run, a low number of fitness function evaluations does not seem to be a guarantee for generating better solution qualities, as can be seen clearly by the results of the CMAES algorithm, which shows signs of getting stuck, where more evaluations would probably not help in improving the situation. Something else that is interesting about the CMAES algorithm is that it is very consistent on the number of needed fitness function evaluations over all the used test functions, while the PSO algorithms are a little bit more "diverse" in this regard.

From the cost results it can also be seen, that added diversification can have a big impact in the number of fitness function evaluations. This becomes very apparent, when comparing the costs between the ring/group and all topologies, or when comparing PSO algorithms with a strong focus on exploration, like the MPSO and CCPSO algorithms versus PSO algorithms without a strong focus on exploration. However, as shown by the solution quality results, this tradeoff can lead to better end results.

Overall, based on the gathered results it is safe to assume that the PSO algorithms and especially the MPSO and FIPS algorithms were able to compete with the state-of-the-art robust-EA algorithms and offer an alternative method to finding robust optima.

5. CONCLUSION

PSO algorithms are indeed suitable for robust optimization, when using approximations of robustness measures via sampling. Out of the different PSO algorithms that were tested, the generally best performing algorithms were the multi-swarm PSO and the fully informed PSO algorithms, where the FIPS algorithm was the overall best performer.

The charged and non-charged canonical PSO algorithms did not perform as well, but can still be deemed usable. When comparing the charged and the non-charged versions of the canonical PSO algorithms, they did not show very different behaviors, except that the charged version seemed slightly better performance wise. Nevertheless, because the charged version performed slightly better and that adding charge forces to a PSO algorithm is but a minor modification/addition with a small addition in computational complexity, it could be used as an addition for other PSO algorithms to get a little bit more exploration and a slightly better optimization quality.

This report has shown that using an archive can be beneficial in both reducing the number of fitness function evaluations, while retaining optimization accuracy; and also (more or less) increase the quality of results.

From the test results it could also be established that increasing the number of particles in use of the PSO algorithms does not increase performance from either a solution quality standpoint or from a cost standpoint in the form of fitness function calls, when an archive is used in conjunction. Topologies for diversification/exploration purposes were able to produce slightly better results from a solution quality standpoint, while sacrificing savings in fitness function evaluations, when an archive is used. For topologies that concentrate on local search or exploitation like the all topology, the opposite holds true.

Furthermore, this report also offered a way on how to add robustness to existing algorithms; namely in the form of general extensions, that require only small modifications to base algorithms. In this case this approach allowed the use of existing and tried and tested (PSO) algorithms on robust optimization problems without needing to invent completely new algorithms.

6. OUTLOOK

Future research into using PSO algorithms for finding robust optima could go into several areas. It could be fruitful to take a deeper look at charged algorithms and trying to find general settings which can be used for finding robust optima. Another interesting research area (although maybe not that big) could be to take a look at how archives can be more deeply exploited for increasing optimization quality. A possibility could be to make the archive focus more on that, than on focusing on fitness function reduction. What would also be interesting is to take a more in-depth look at the similarities between dynamic and robust optimization problems, as they seem very similar. Concerning multiple-swarms it could be very interesting to try out more sophisticated multi-swarming algorithms. What makes these multi-swarm algorithms additionally very attractive is the fact, that it is possible to not just get the best result of one swarm, but multiple swarms. This increases the chances of finding a good robust optima by rechecking the solutions found (via a high precision MEM method) (although this has not been done here).

Other future research could involve modifying other existing meta-heuristics for robust optimization by using algorithm extensions as presented in this paper. It would also be interesting to see, if using metamodels to complement the archive approach could lead to better results on PSO algorithms.

7. DISCUSSION: ELITISM AND NOISE

In the PSO algorithms shown only the best personal and neighborhood solutions are considered in the velocity update function, which controls the movement of the particles. This behavior of only considering the best solutions can be described as "elitism". Robust problems which are handled with an approximation of a robustness measure are noisy to a degree, depending on the quality of the approximation. This noise could prove deadly for an algorithm (family), that uses elitism. The reason for this is that the PSO algorithm might be led towards a wrong path via a wrongly approximated fitness value. If this wrongly approximated value for some reason is extremely good, it could result in the algorithm getting trapped towards a wrong path without hope of getting back towards a better path. However, this problem has not strongly manifested itself in this study except on some cases, like for the robust canonical PSO on test problem 1, where the algorithm converged to a worse end result than what it initially found.

A possible reason on why elitism has not had a big impact on the experimental results of this study could be that the amount and strength of the noise in the test problems might not have been enough to create significant problems. This idea is backed up by the performance of the archive-based algorithms, which do not show the same bad convergence behavior as the pure sampling-based algorithms and can usually generate much better robustness approximations, because of the added samples provided by the archive. Better approximations would lead to less noise and thus less problems.

Another possible factor could be the choice of topologies. Based on the produced results, it seems that the all topology is more susceptible to a problem with noise and elitism, than the ring or group topologies. A possible explanation on why the latter topologies were not as affected as the all topology, could be that the propagation-delay of information between the particles is not instantaneously, but is bound to the number of particles or groups of particles used. Say for example that one particle would be affected by a very bad robustness approximation with a high fitness value. This value would need some time to reach all the other particles to influence them. In this time the particles have the possibility of finding better robustness approximations, which (if found) could counter the negative effects of the badly approximated value. This "safety net" is of course not available for the all topology, as all particles in this topology receive this value almost immediately.

Several solutions are thinkable to defuse the problems PSOs could have with the combination of noise and elitism. One possible way might be to remove or weaken the elitism aspect of the PSO algorithms. This could for example be done through re-sampling of personal and neighborhood best solutions or by using solution stacks for those variables and using a (randomized) scheme to choose solutions from the stack for the velocity update function.

Another possibility of reducing possible problems concerning noise and elitism might be to to reduce the noise itself. This could for example be achieved by increasing the number of samples used in the fitness functions. However, this might not be desirable, because of the added computational costs involved in this. Possible alternatives could be the use of topologies with propagation-delays in information or the use of an archive, which in later stages (once the archive is filled) should increase the number of samples for the robustness approximations substantially.

References

[1] *Particle swarm optimization*, volume 4, August 2002.

[2] H.-G. Beyer and B. Sendhoff. Robust optimization - a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33-34):3190–3218, July 2007.

[3] T. M. Blackwell. Dynamic search with charged swarms. In *in GECCO 2002*, pages 9–13. Morgan Kaufmann Publishers, 2002.

[4] J. Branke. Creating robust solutions by means of evolutionary algorithms. In *PPSN V: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pages 119–128, London, UK, 1998. Springer-Verlag.

[5] J. Branke. *Evolutionary Optimization in Dynamic Environments.* Kluwer Academic Publishers, Norwell, MA, USA, 2001.

[6] M. Clerc and J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *Evolutionary Computation, IEEE Transactions on*, 6(1):58–73, 2002.

[7] A. P. Engelbrecht. *Computational Intelligence: An Introduction.* Wiley Publishing, 2007.

[8] Y. Jin and J. Branke. Evolutionary optimization in uncertain environments-a survey. *Evolutionary Computation, IEEE Transactions on*, 9(3):303–317, 2005.

[9] M. T. E. Johannes W. Kruisselbrink and T. Bäck. An archive based method for applying evolutionary algorithms to find robust optima. *Liacs Technical Report*, 2009.

[10] W. H. Kruskal and W. A. Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[11] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61, 2000.

[12] R. Mendes, J. Kennedy, and J. Neves. The fully informed particle swarm: Simpler, maybe better. *IEEE Transactions on Evolutionary Computation*, 8:204–210, 2004.

[13] I. Paenke, J. Branke, and Y. Jin. Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation. *Evolutionary Computation, IEEE Transactions on*, 10(4):405–420, 2006.

[14] R. Poli, J. Kennedy, and T. Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1(1):33–57, June 2007.

[15] S. Tsutsui and A. Ghosh. Effects of adding perturbations to phenotypic parameters in genetic algorithms for searching robust solutions. pages 351–365, 2003.

[16] S. Yang, Y.-S. Ong, and Y. Jin, editors. *Evolutionary Computation in Dynamic and Uncertain Environments*, volume 51 of *Studies in Computational Intelligence.* Springer, 2007.

Test function	Topology	#particles	Rob. approx.	Average	Std. dev.	Median	Rank sum	Rank
TP1	ring	30	MEM	0.2822	0.1468	0.2624	5316	8
	ring	100	MEM	0.2988	0.1684	0.2239	5351	9
	all	30	MEM	0.1892	0.0916	0.1625	3733	4
	all	100	MEM	0.2021	0.0939	0.1961	4062	5
	5,30-group	30	MEM	0.2428	0.1448	0.1998	4564	6
	5,100-group	100	MEM	0.2681	0.1275	0.2536	5171	7
	ring	30	Archive	0.2052	0.1420	0.1574	3706	3
	all	30	Archive	0.0964	0.0342	0.0854	1113	1
	5,30-group	30	Archive	0.1865	0.1001	0.1530	3569	2
TP2	ring	30	MEM	0.9713	0.0085	0.9686	4326	7
	ring	100	MEM	0.9852	0.0286	0.9747	5905	8
	all	30	MEM	0.9641	0.0042	0.9635	2257	1
	all	100	MEM	0.9659	0.0054	0.9648	2834	3
	5,30-group	30	MEM	0.9724	0.0108	0.9695	4318	6
	5,100-group	100	MEM	0.9872	0.0206	0.9846	6330	9
	ring	30	Archive	0.9712	0.0103	0.9688	4113	5
	all	30	Archive	0.9657	0.0067	0.9645	2709	2
	5,30-group	30	Archive	0.9700	0.0094	0.9684	3793	4
TP3	ring	30	MEM	0.4940	0.1339	0.4511	4211	6
	ring	100	MEM	0.5156	0.1376	0.4580	4790	9
	all	30	MEM	0.5233	0.1317	0.6078	4180	5
	all	100	MEM	0.4466	0.1319	0.3580	3066	1
	5,30-group	30	MEM	0.4971	0.1299	0.4644	4310	7
	5,100-group	100	MEM	0.5027	0.1340	0.4338	4630	8
	ring	30	Archive	0.4897	0.1370	0.3844	4127	3
	all	30	Archive	0.5497	0.1145	0.6083	4133	4
	5,30-group	30	Archive	0.4538	0.1314	0.3737	3138	2
TP4	ring	30	MEM	0.4244	0.0378	0.4107	5786	8
	ring	100	MEM	0.4124	0.0205	0.4045	5647	7
	all	30	MEM	0.3886	0.0082	0.3869	2785	3
	all	100	MEM	0.3925	0.0108	0.3893	3420	4
	5,30-group	30	MEM	0.4060	0.0221	0.4003	5097	6
	5,100-group	100	MEM	0.4321	0.0343	0.4261	6459	9
	ring	30	Archive	0.3936	0.0111	0.3904	3600	5
	all	30	Archive	0.3812	0.0039	0.3803	1131	1
	5,30-group	30	Archive	0.3920	0.0241	0.3840	2660	2

Table 9: CPSO statistical data

Topology	#particles	Rob. approx.	Summed Rank (SR)	Unimodal SR	Multimodal SR
ring	30	MEM	29	15	14
ring	100	MEM	33	17	16
all	30	MEM	13	5	8
all	100	MEM	13	8	5
5,30-group	30	MEM	25	12	13
5,100-group	100	MEM	33	16	17
ring	30	Archive	16	8	8
all	30	Archive	8	3	5
5,30-group	30	Archive	10	6	4

Table 10: CPSO ranks summed up over all test functions

Test function	Topology	#particles	Rob. approx.	Average	Std. dev.	Median	Rank sum	Rank
TP1	ring	30	MEM	0.1319	0.0538	0.1204	5199	7
	ring	100	MEM	0.2034	0.1224	0.1441	6683	9
	all	30	MEM	0.0833	0.0057	0.0831	2842	3
	all	100	MEM	0.0950	0.0362	0.0815	3184	4
	5,30-group	30	MEM	0.1003	0.0257	0.0908	4021	6
	5,100-group	100	MEM	0.1872	0.1139	0.1483	6195	8
	ring	30	Archive	0.1103	0.0800	0.0886	3450	5
	all	30	Archive	0.0848	0.0184	0.0800	2486	1
	5,30-group	30	Archive	0.0860	0.0184	0.0776	2525	2
TP2	ring	30	MEM	0.9642	0.0054	0.9642	3868	4
	ring	100	MEM	0.9737	0.0105	0.9699	6196	9
	all	30	MEM	0.9598	0.0048	0.9609	2097	1
	all	100	MEM	0.9635	0.0040	0.9640	3584	3
	5,30-group	30	MEM	0.9628	0.0040	0.9620	3288	2
	5,100-group	100	MEM	0.9683	0.0082	0.9667	4938	8
	ring	30	Archive	0.9663	0.0057	0.9658	4629	7
	all	30	Archive	0.9644	0.0044	0.9646	3894	5
	5,30-group	30	Archive	0.9651	0.0059	0.9643	4091	6
TP3	ring	30	MEM	0.3785	0.0726	0.3556	3713	3
	ring	100	MEM	0.3726	0.0164	0.3721	4714	7
	all	30	MEM	0.5414	0.1223	0.6072	5405	8
	all	100	MEM	0.5588	0.1090	0.6066	5899	9
	5,30-group	30	MEM	0.3777	0.0681	0.3577	3782	5
	5,100-group	100	MEM	0.3595	0.0114	0.3559	3757	4
	ring	30	Archive	0.3684	0.0568	0.3488	2894	2
	all	30	Archive	0.4930	0.1337	0.5991	4178	6
	5,30-group	30	Archive	0.3621	0.0538	0.3468	2243	1
TP4	ring	30	MEM	0.3876	0.0070	0.3860	5183	7
	ring	100	MEM	0.3969	0.0125	0.3937	6377	9
	all	30	MEM	0.3879	0.0259	0.3804	3200	4
	all	100	MEM	0.3811	0.0033	0.3806	2934	1
	5,30-group	30	MEM	0.3857	0.0185	0.3812	3670	6
	5,100-group	100	MEM	0.3906	0.0075	0.3909	5787	8
	ring	30	Archive	0.3811	0.0040	0.3803	2948	2
	all	30	Archive	0.3935	0.0339	0.3811	3387	5
	5,30-group	30	Archive	0.3819	0.0053	0.3807	3099	3

Table 11: FIPS statistical data

Topology	#particles	Rob. approx.	Summed Rank (SR)	Unimodal SR	Multimodal SR
ring	30	MEM	21	11	10
ring	100	MEM	34	18	16
all	30	MEM	16	4	12
all	100	MEM	17	7	10
5,30-group	30	MEM	19	8	11
5,100-group	100	MEM	28	16	12
ring	30	Archive	16	12	4
all	30	Archive	17	6	11
5,30-group	30	Archive	12	8	4

Table 12: FIPS ranks summed up over all test functions

Test function	Topology	#particles	Rob. approx.	Average	Std. dev.	Median	Rank sum	Rank
TP1	ring	30	MEM	0.2177	0.1773	0.1793	3182	4
	all	30	MEM	0.2044	0.1013	0.1849	3265	5
	5,30-group	30	MEM	0.2223	0.1438	0.1860	3409	6
	ring	30	Archive	0.1956	0.1678	0.1352	2879	3
	all	30	Archive	0.0915	0.0276	0.0817	925	1
	5,30-group	30	Archive	0.1597	0.0776	0.1416	2630	2
TP2	ring	30	MEM	0.9704	0.0075	0.9689	3159	6
	all	30	MEM	0.9659	0.0059	0.9647	2206	1
	5,30-group	30	MEM	0.9693	0.0111	0.9672	2684	3
	ring	30	Archive	0.9698	0.0070	0.9680	3082	5
	all	30	Archive	0.9675	0.0103	0.9665	2421	2
	5,30-group	30	Archive	0.9680	0.0055	0.9674	2738	4
TP3	ring	30	MEM	0.4504	0.1267	0.3708	2908	4
	all	30	MEM	0.5077	0.1362	0.6156	3226	6
	5,30-group	30	MEM	0.4508	0.1167	0.3827	2911	5
	ring	30	Archive	0.4521	0.1255	0.3609	2463	2
	all	30	Archive	0.4980	0.1351	0.6052	2542	3
	5,30-group	30	Archive	0.4351	0.1145	0.3606	2240	1
TP4	ring	30	MEM	0.4081	0.0259	0.3964	3766	6
	all	30	MEM	0.3870	0.0069	0.3844	1939	2
	5,30-group	30	MEM	0.4034	0.0183	0.4007	3752	5
	ring	30	Archive	0.3956	0.0247	0.3905	2687	3
	all	30	Archive	0.3834	0.0047	0.3827	1262	1
	5,30-group	30	Archive	0.3933	0.0106	0.3933	2884	4

Table 13: CCPSO statistical data

Topology	#particles	Rob. approx.	Summed Rank (SR)	Unimodal SR	Multimodal SR
ring	30	MEM	20	10	10
all	30	MEM	14	6	8
5,30-group	30	MEM	19	9	10
ring	30	Archive	13	8	5
all	30	Archive	7	3	4
5,30-group	30	Archive	11	6	5

Table 14: CCPSO ranks summed up over all test functions

Test function	Topology	#particles	Rob. approx.	Average	Std. dev.	Median	Rank sum	Rank
TP1	ring	30	MEM	0.1685	0.0888	0.1409	2457	4
	all	30	MEM	0.1151	0.0977	0.0889	1104	1
	ring	30	Archive	0.1453	0.0575	0.1260	2198	3
	all	30	Archive	0.1189	0.0548	0.0990	1501	2
TP2	ring	30	MEM	0.9674	0.0091	0.9653	1882	3
	all	30	MEM	0.9654	0.0051	0.9652	1708	2
	ring	30	Archive	0.9678	0.0055	0.9678	2108	4
	all	30	Archive	0.9648	0.0047	0.9643	1562	1
TP3	ring	30	MEM	0.3603	0.0195	0.3552	1964	3
	all	30	MEM	0.3746	0.0559	0.3550	2076	4
	ring	30	Archive	0.3688	0.0534	0.3519	1803	2
	all	30	Archive	0.3604	0.0366	0.3481	1417	1
TP4	ring	30	MEM	0.3925	0.0116	0.3891	2326	4
	all	30	MEM	0.3945	0.0317	0.3828	1616	2
	ring	30	Archive	0.3861	0.0051	0.3851	1815	3
	all	30	Archive	0.3929	0.0290	0.3823	1503	1

Table 15: MPSO statistical data

Topology	#particles	Rob. approx.	Summed Rank (SR)	Unimodal SR	Multimodal SR
ring	30	MEM	14	7	7
all	30	MEM	9	3	6
ring	30	Archive	12	7	5
all	30	Archive	5	3	2

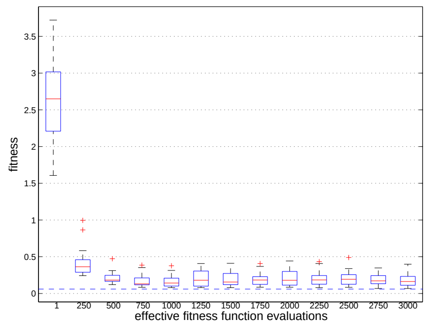
Table 16: MPSO ranks summed up over all test functions

Test function	Algorithm	Average	Std. dev.	Median	Rank sum	Rank
	Rob. CMAES (LHSR)	0.2059	0.1614	0.1841	6016	9
	MPSO (all, 30, archive)	0.1189	0.0548	0.0990	3952	5
	MPSO (ring, 30, archive)	0.1453	0.0575	0.1260	5266	6
	CCPSO (group, 30, archive)	0.1597	0.0776	0.1416	5326	7
TP1	CCPSO (all, 30, archive)	0.0915	0.0276	0.0817	2515	3
	FIPS (group, 30, archive)	0.0860	0.0184	0.0776	2981	2
	FIPS (all, 30, archive)	0.0848	0.0184	0.0800	2088	1
	CPSO (group, 30, archive)	0.1865	0.1001	0.1530	5734	8
	CPSO (all, 30, archive)	0.0964	0.0342	0.0854	2707	4
	Rob. CMAES (LHSR)	0.9682	0.0054	0.9682	4720	8
	MPSO (all, 30, archive)	0.9648	0.0047	0.9643	3277	2
	MPSO (ring, 30, archive)	0.9678	0.0055	0.9678	4583	6
	CCPSO (group, 30, archive)	0.9680	0.0055	0.9674	4629	7
TP2	CCPSO (all, 30, archive)	0.9675	0.0103	0.9665	4117	5
	FIPS (group, 30, archive)	0.9651	0.0059	0.9643	3930	3
	FIPS (all, 30, archive)	0.9644	0.0044	0.9646	3085	1
	CPSO (group, 30, archive)	0.9700	0.0094	0.9684	4756	9
	CPSO (all, 30, archive)	0.9657	0.0067	0.9645	3488	4
	Rob. CMAES (LHSR)	0.5104	0.1294	0.6050	4356	6
	MPSO (all, 30, archive)	0.3604	0.0366	0.3481	2784	2
	MPSO (ring, 30, archive)	0.3688	0.0534	0.3519	3318	3
	CCPSO (group, 30, archive)	0.4351	0.1145	0.3606	4186	5
TP3	CCPSO (all, 30, archive)	0.4980	0.1351	0.6052	4754	8
	FIPS (group, 30, archive)	0.3621	0.0538	0.3468	3111	1
	FIPS (all, 30, archive)	0.4930	0.1337	0.5991	3966	4
	CPSO (group, 30, archive)	0.4538	0.1314	0.3737	4563	7
	CPSO (all, 30, archive)	0.5497	0.1145	0.6083	5547	9
	Rob. CMAES (LHSR)	0.4110	0.0453	0.3838	4627	6
	MPSO (all, 30, archive)	0.3929	0.0290	0.3823	3979	5
	MPSO (ring, 30, archive)	0.3861	0.0051	0.3851	4887	8
	CCPSO (group, 30, archive)	0.3933	0.0106	0.3933	5866	9
TP4	CCPSO (all, 30, archive)	0.3834	0.0047	0.3827	3797	4
	FIPS (group, 30, archive)	0.3819	0.0053	0.3807	2832	2
	FIPS (all, 30, archive)	0.3935	0.0339	0.3811	3145	3
	CPSO (group, 30, archive)	0.3920	0.0241	0.3840	4657	7
	CPSO (all, 30, archive)	0.3812	0.0039	0.3803	2795	1

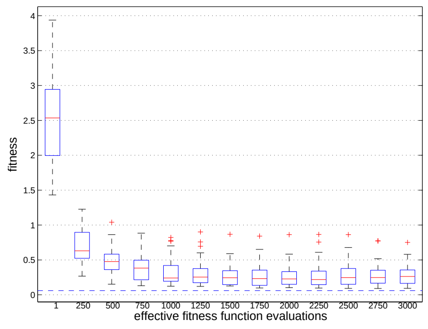
Table 17: Comparison statistical data

Algorithm	Summed Rank (SR)	Unimodal SR	Multimodal SR
Rob. CMAES (LHSR)	29	17	12
MPSO (all, 30, archive)	14	7	7
MPSO (ring, 30, archive)	23	12	11
CCPSO (group, 30, archive)	28	14	14
CCPSO (all, 30, archive)	20	8	12
FIPS (group, 30, archive)	8	5	3
FIPS (all, 30, archive)	9	2	7
CPSO (group, 30, archive)	31	17	14
CPSO (all, 30, archive)	18	8	10

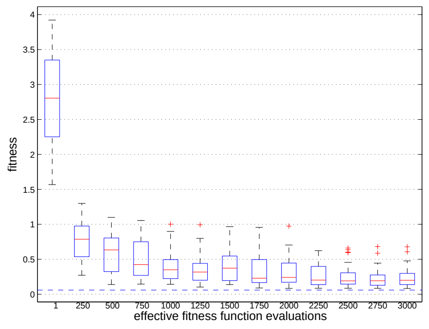
Table 18: Comparison ranks summed up over all test functions



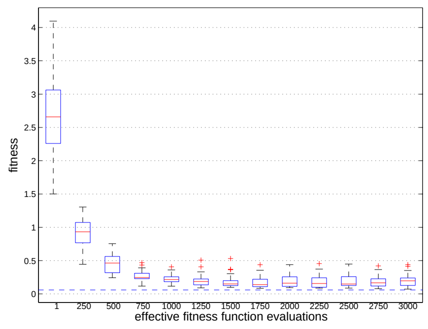
(a) All topology & 30 particles



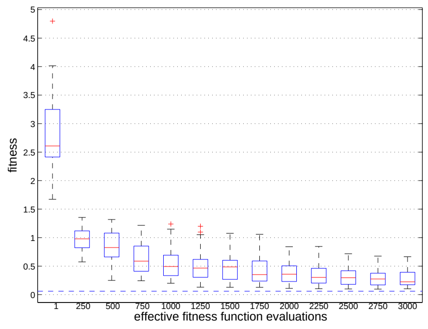
(b) Ring topology & 30 particles



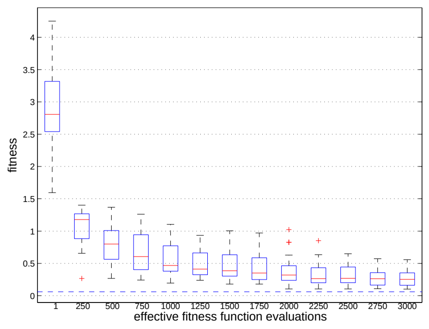
(c) 5,30-Group topology & 30 particles



(d) All topology & 100 particles

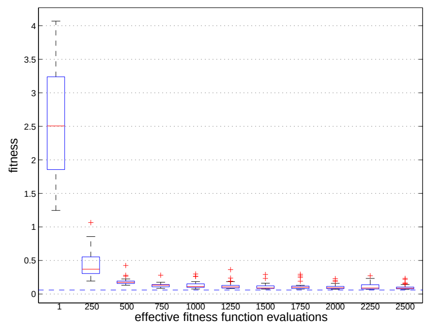


(e) Ring topology & 100 particles

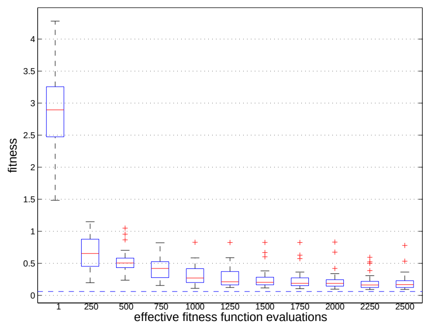


(f) 5,30-Group topology & 100 particles

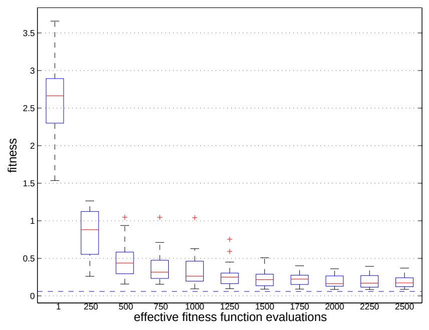
Figure 26: TP1 Results (CPSO - Sampling)



(a) All topology & 30 particles

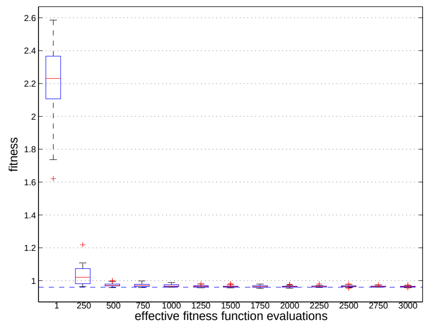


(b) Ring topology & 30 particles

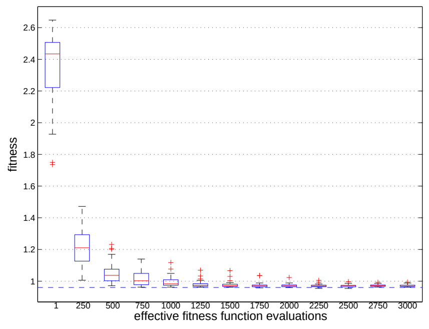


(c) 5,30-Group topology & 30 particles

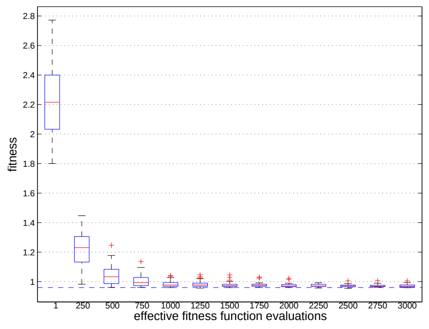
Figure 27: TP1 Results (CPSO - Archive)



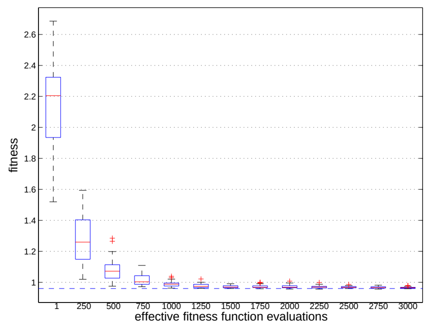
(a) All topology & 30 particles



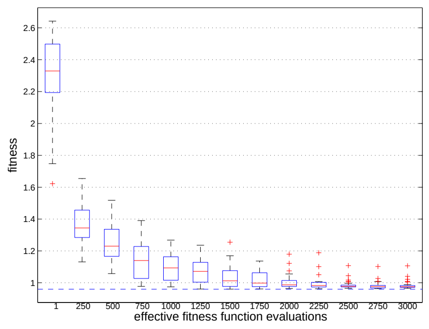
(b) Ring topology & 30 particles



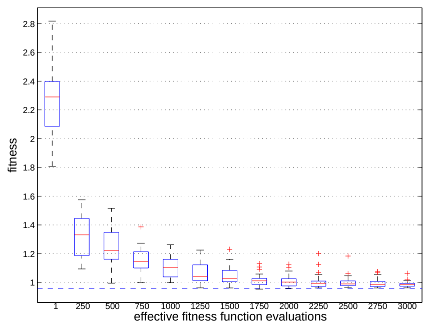
(c) 5,30-Group topology & 30 particles



(d) All topology & 100 particles

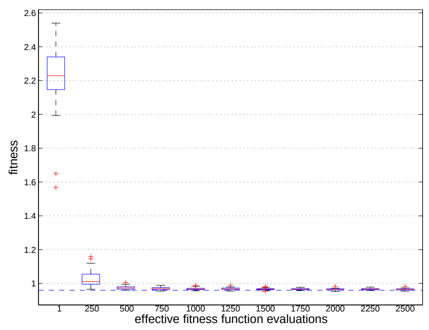


(e) Ring topology & 100 particles

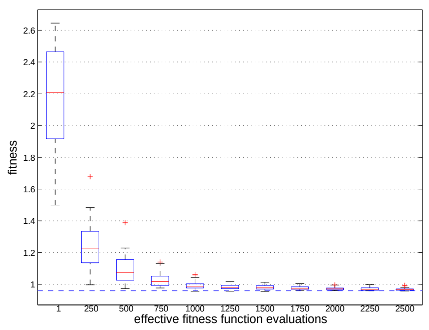


(f) 5,30-Group topology & 100 particles

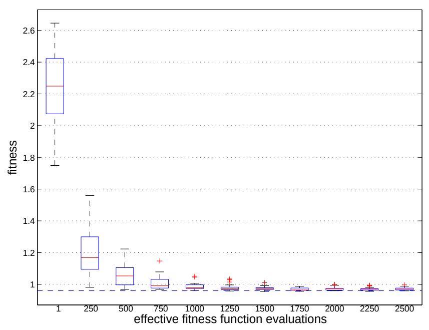
Figure 28: TP2 Results (CPSO - Sampling)



(a) All topology & 30 particles

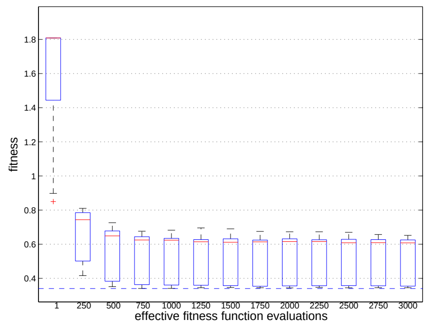


(b) Ring topology & 30 particles

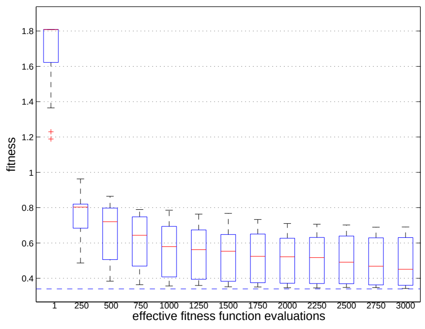


(c) 5,30-Group topology & 30 particles

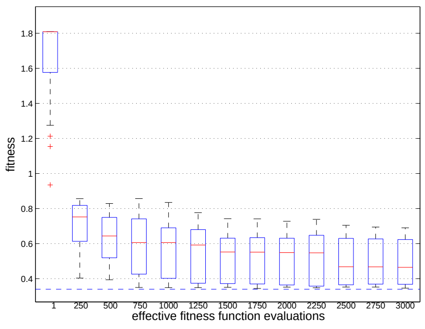
Figure 29: TP2 Results (CPSO - Archive)



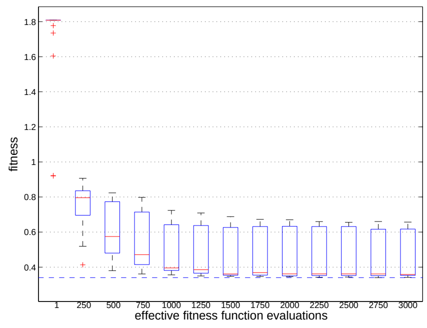
(a) All topology & 30 particles



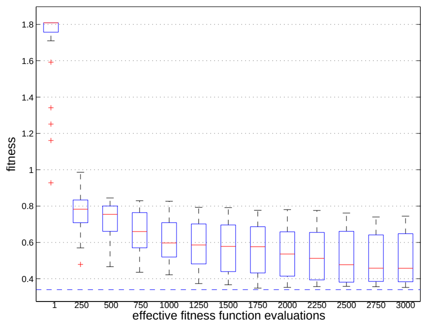
(b) Ring topology & 30 particles



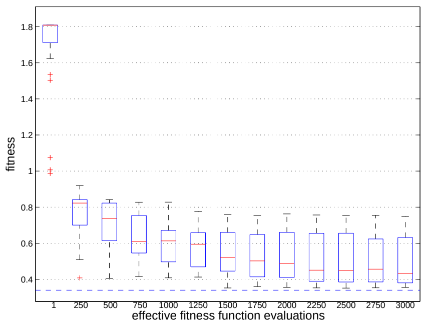
(c) 5,30-Group topology & 30 particles



(d) All topology & 100 particles

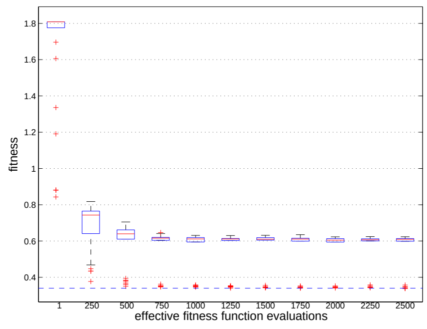


(e) Ring topology & 100 particles

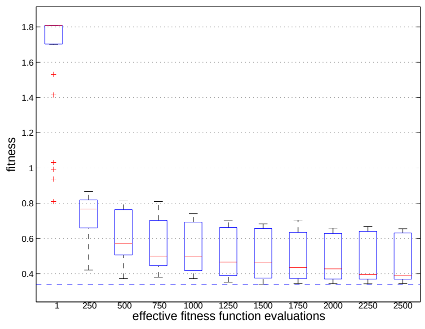


(f) 5,30-Group topology & 100 particles

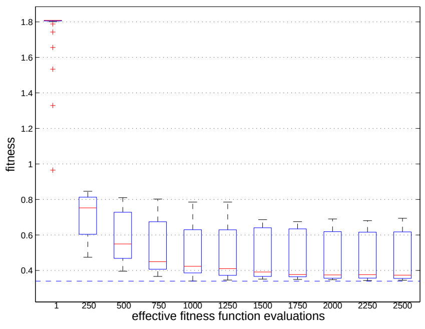
Figure 30: TP3 Results (CPSO - Sampling)



(a) All topology & 30 particles

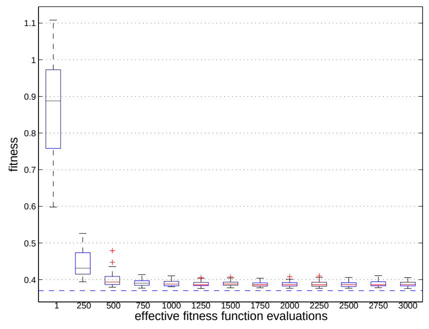


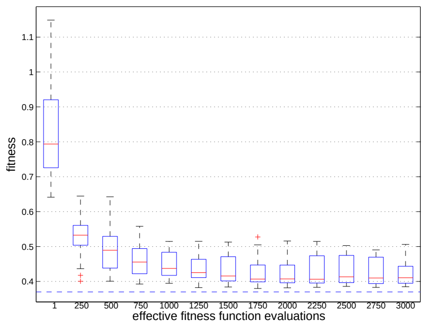
(b) Ring topology & 30 particles

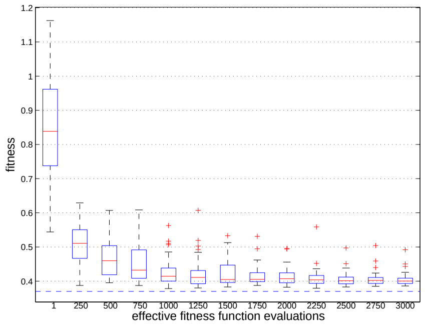


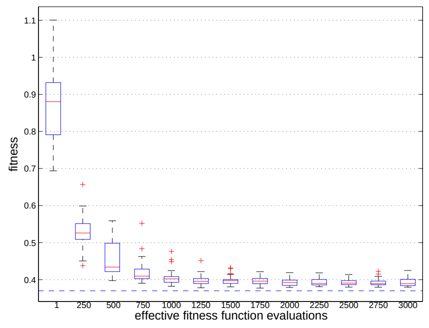
(c) 5,30-Group topology & 30 particles

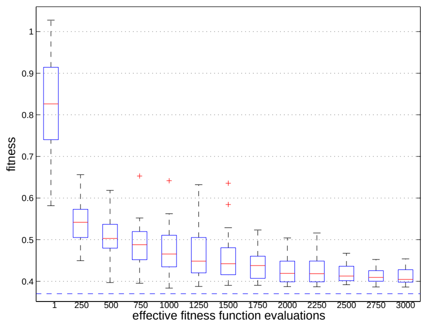
Figure 31: TP3 Results (CPSO - Archive)


(a) All topology & 30 particles


(b) Ring topology & 30 particles


(c) 5,30-Group topology & 30 particles


(d) All topology & 100 particles


(e) Ring topology & 100 particles

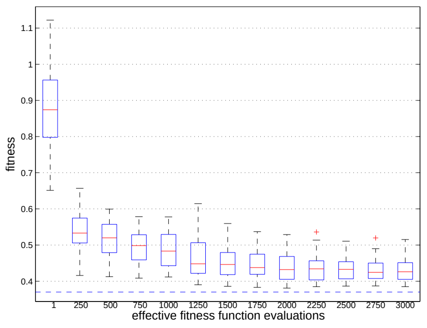
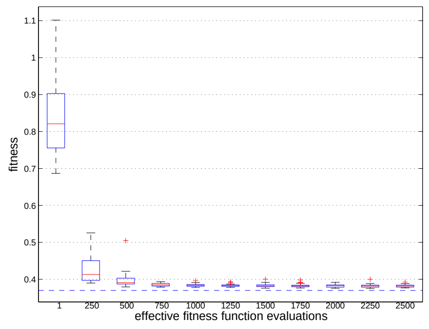
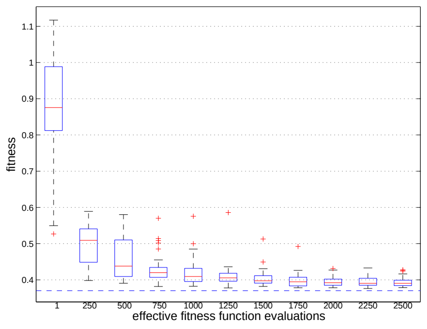

(f) 5,30-Group topology & 100 particles

Figure 32: TP4 Results (CPSO - Sampling)


(a) All topology & 30 particles


(b) Ring topology & 30 particles

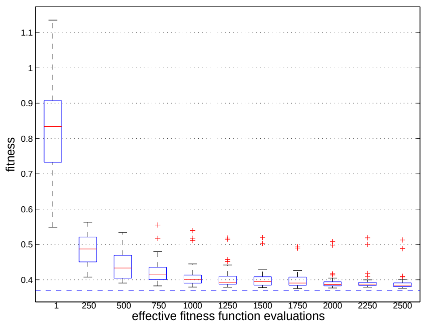
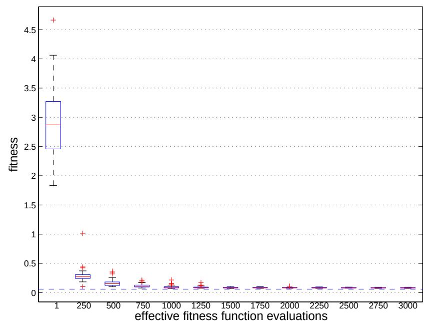
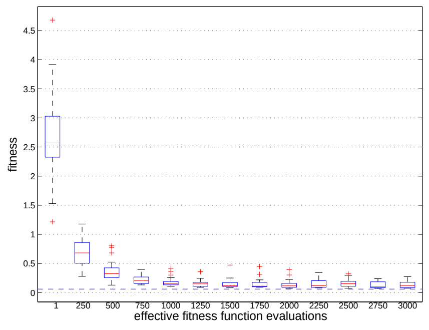
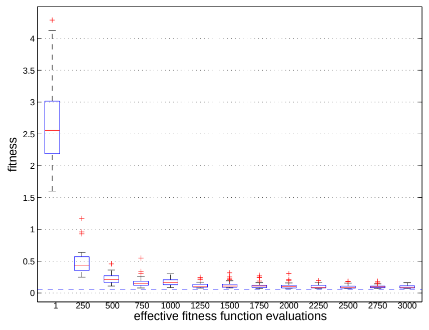
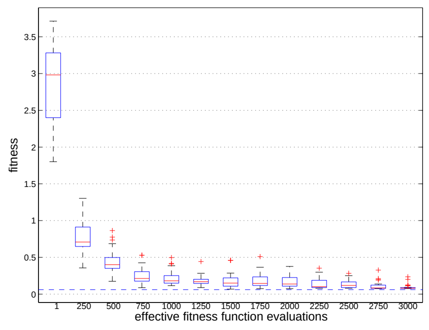

(c) 5,30-Group topology & 30 particles

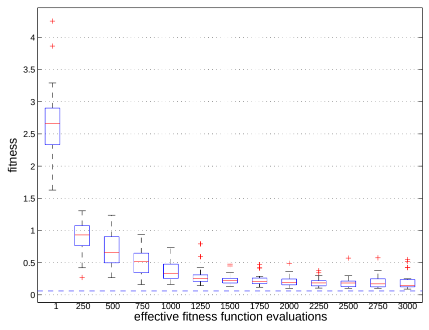
Figure 33: TP4 Results (CPSO - Archive)


(a) All topology & 30 particles


(b) Ring topology & 30 particles


(c) 5,30-Group topology & 30 particles


(d) All topology & 100 particles


(e) Ring topology & 100 particles

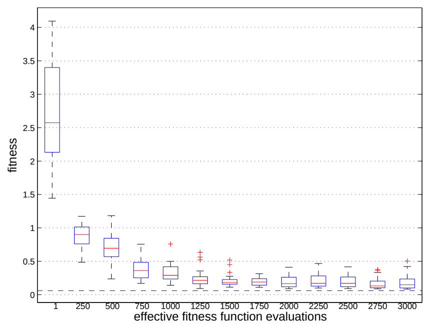
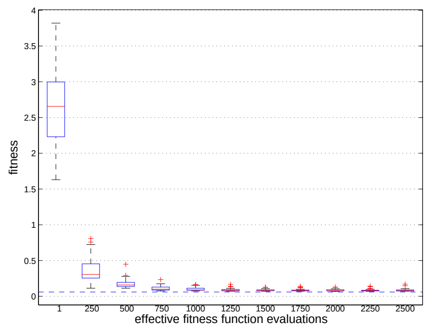
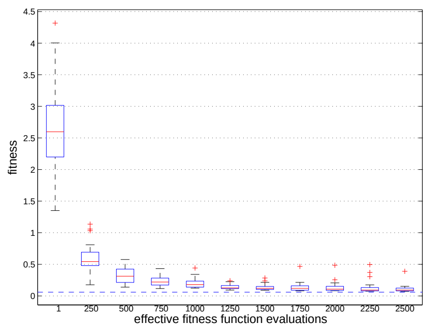

(f) 5,30-Group topology & 100 particles

Figure 34: TP1 Results (FIPS - Sampling)


(a) All topology & 30 particles


(b) Ring topology & 30 particles

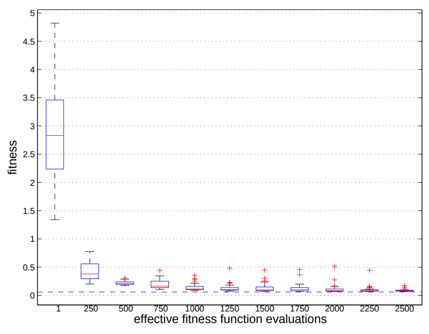
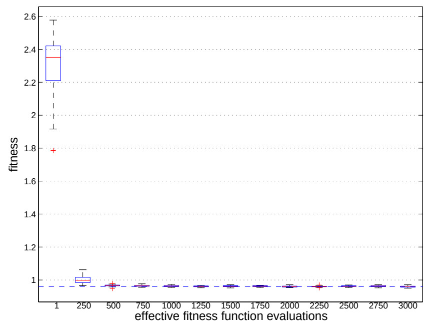
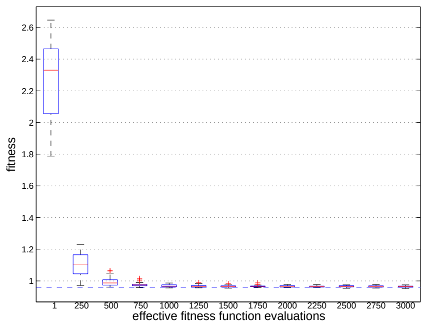

(c) 5,30-Group topology & 30 particles

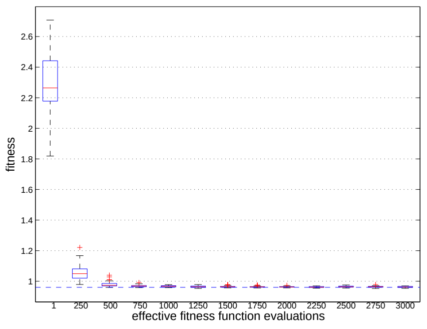
Figure 35: TP1 Results (FIPS - Archive)



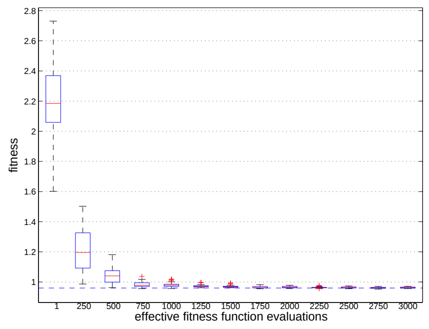
(a) All topology & 30 particles



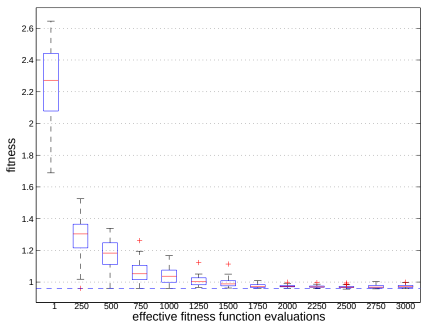
(b) Ring topology & 30 particles



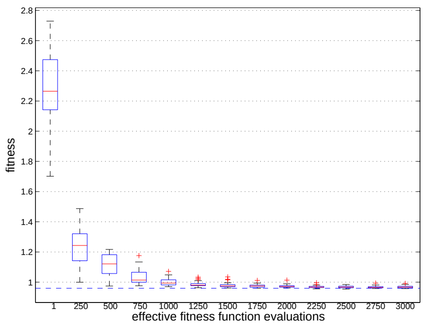
(c) 5,30-Group topology & 30 particles



(d) All topology & 100 particles

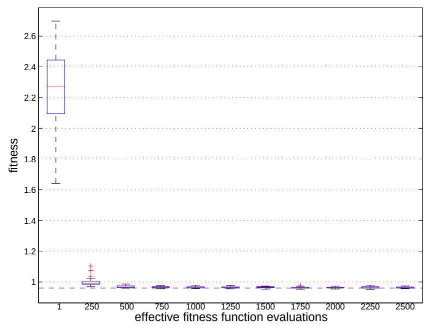


(e) Ring topology & 100 particles

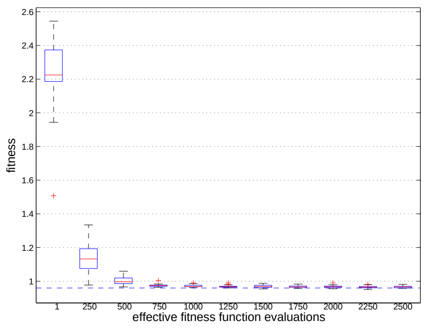


(f) 5,30-Group topology & 100 particles

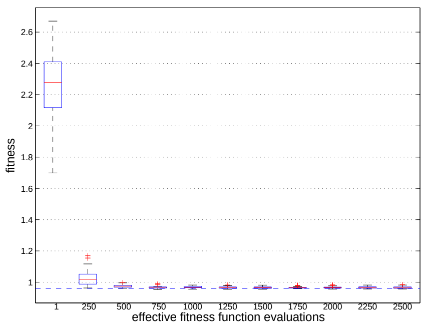
Figure 36: TP2 Results (FIPS - Sampling)



(a) All topology & 30 particles

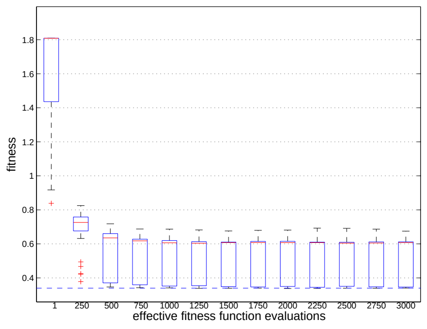


(b) Ring topology & 30 particles

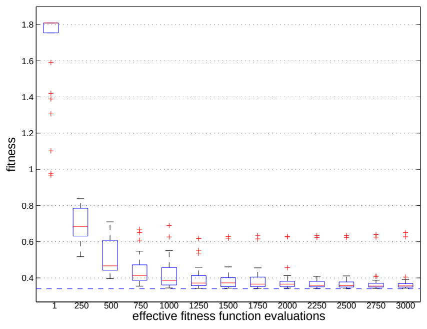


(c) 5,30-Group topology & 30 particles

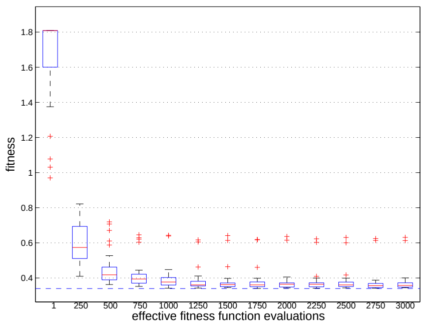
Figure 37: TP2 Results (FIPS - Archive)



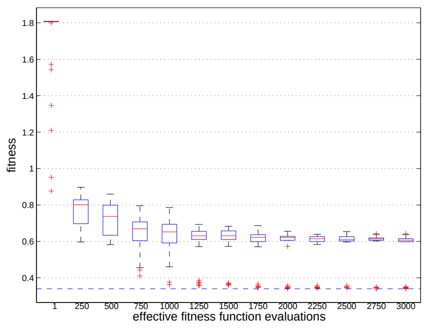
(a) All topology & 30 particles



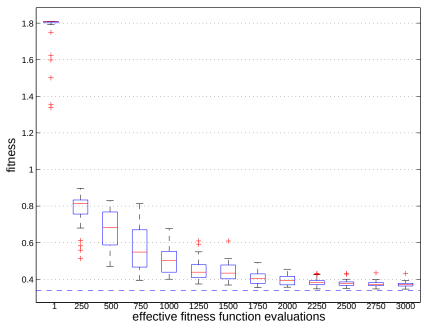
(b) Ring topology & 30 particles



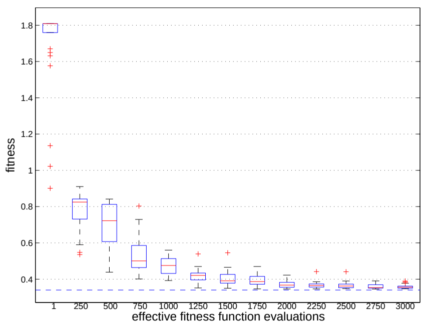
(c) 5,30-Group topology & 30 particles



(d) All topology & 100 particles

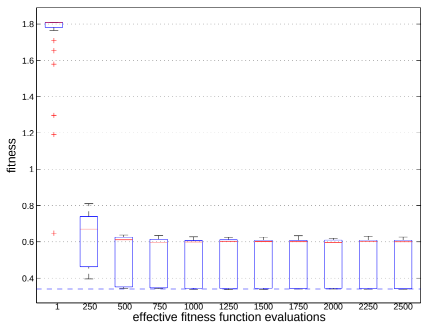


(e) Ring topology & 100 particles

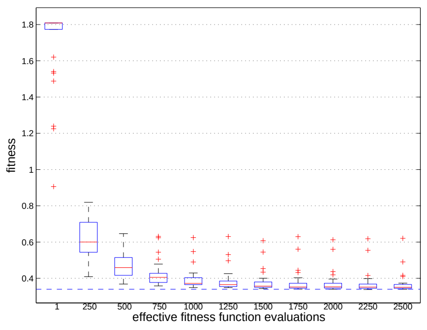


(f) 5,30-Group topology & 100 particles

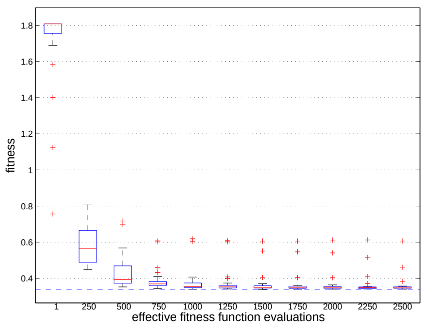
Figure 38: TP3 Results (FIPS - Sampling)



(a) All topology & 30 particles

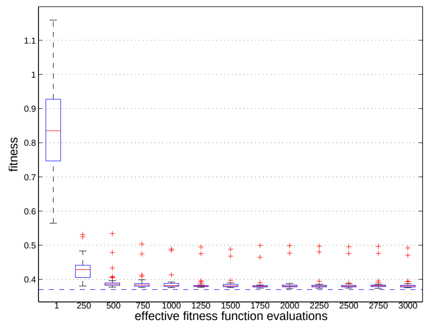


(b) Ring topology & 30 particles

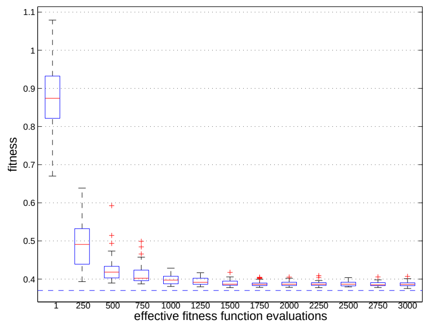


(c) 5,30-Group topology & 30 particles

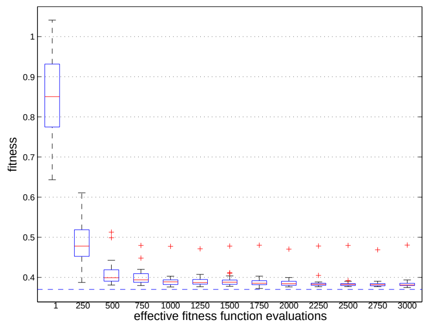
Figure 39: TP3 Results (FIPS - Archive)



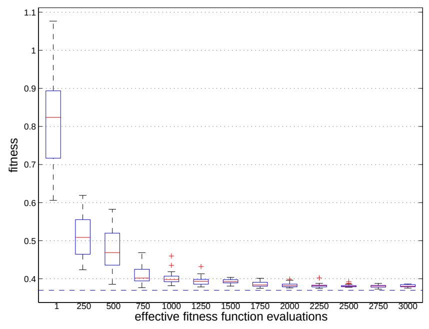
(a) All topology & 30 particles



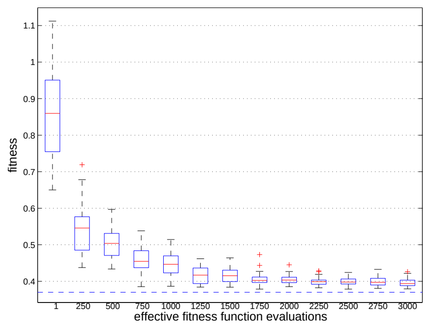
(b) Ring topology & 30 particles



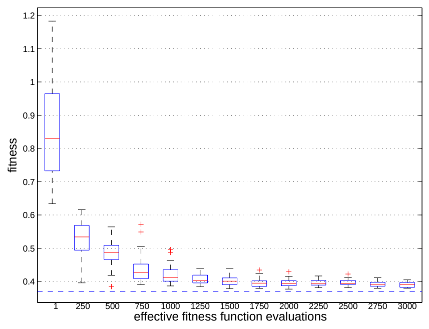
(c) 5,30-Group topology & 30 particles



(d) All topology & 100 particles

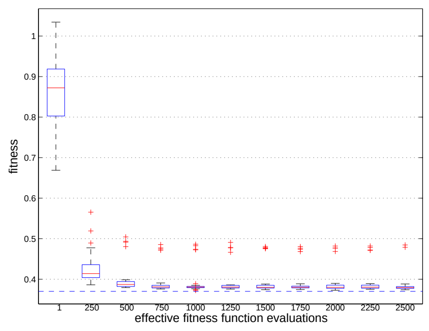


(e) Ring topology & 100 particles

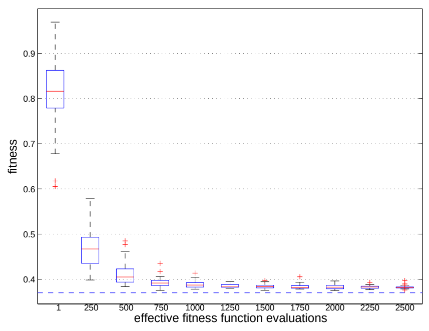


(f) 5,30-Group topology & 100 particles

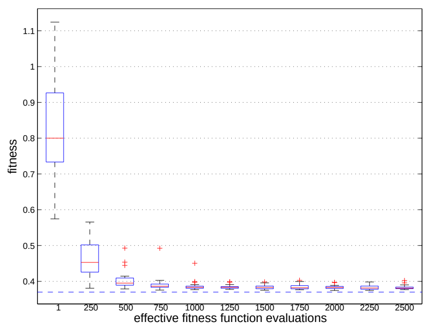
Figure 40: TP4 Results (FIPS - Sampling)



(a) All topology & 30 particles

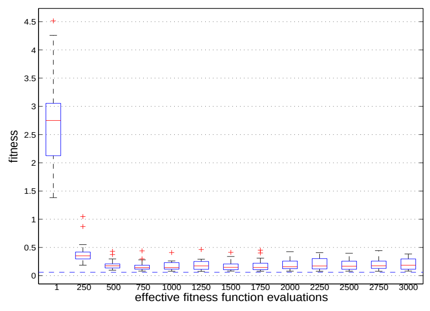


(b) Ring topology & 30 particles

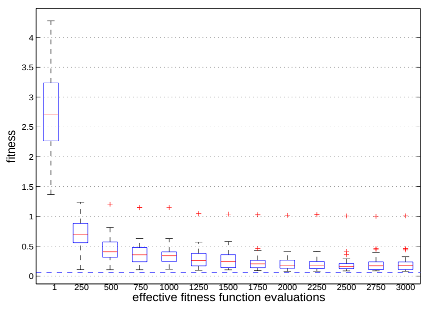


(c) 5,30-Group topology & 30 particles

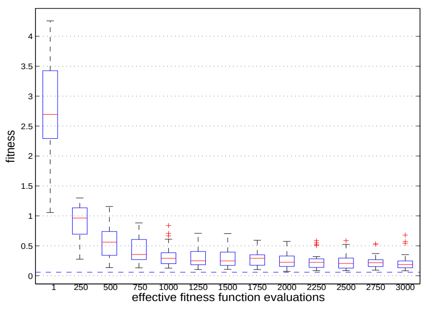
Figure 41: TP4 Results (FIPS - Archive)



(a) All topology & 30 particles

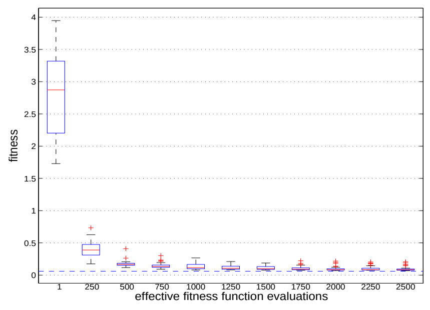


(b) Ring topology & 30 particles

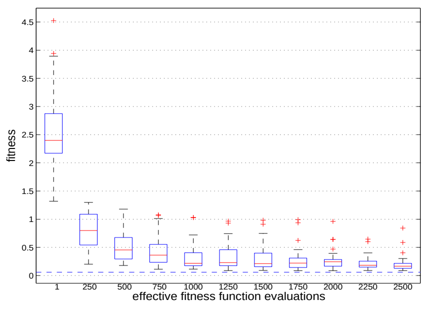


(c) 5,30-Group topology & 30 particles

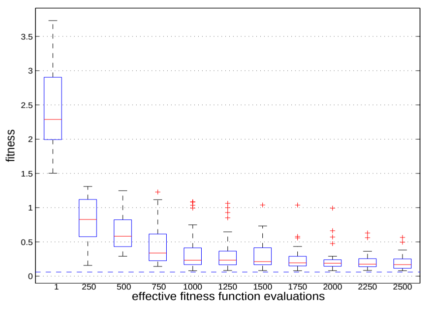
Figure 42: TP1 Results (CCPSO - Sampling)



(a) All topology & 30 particles

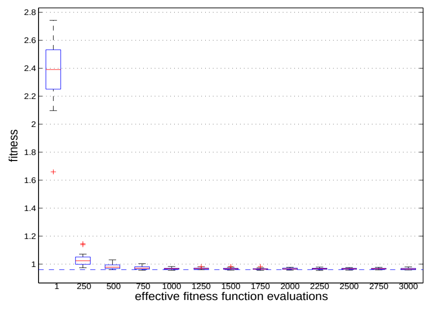


(b) Ring topology & 30 particles

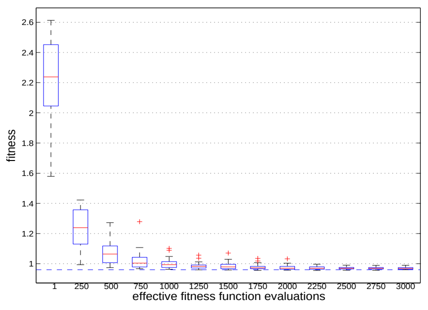


(c) 5,30-Group topology & 30 particles

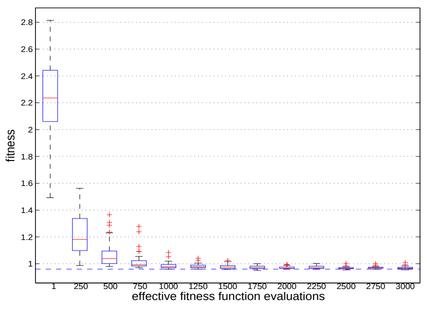
Figure 43: TP1 Results (CCPSO - Archive)



(a) All topology & 30 particles

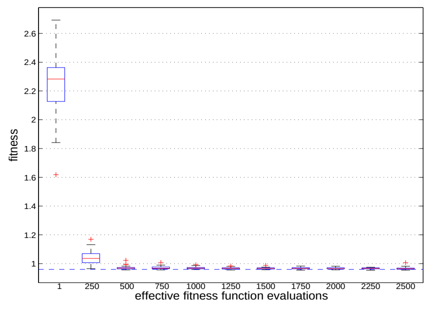


(b) Ring topology & 30 particles

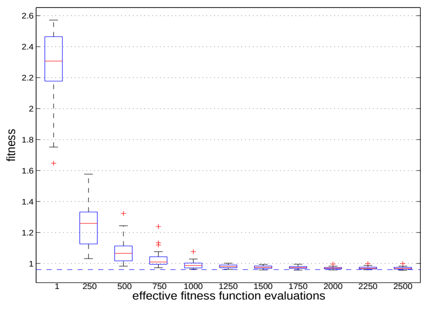


(c) 5,30-Group topology & 30 particles

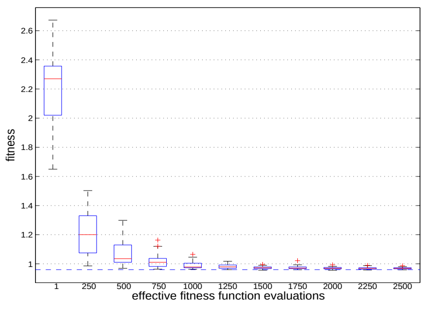
Figure 44: TP2 Results (CCPSO - Sampling)



(a) All topology & 30 particles

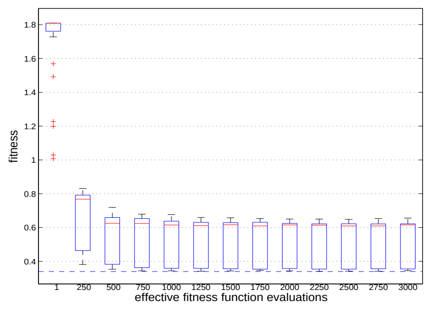


(b) Ring topology & 30 particles

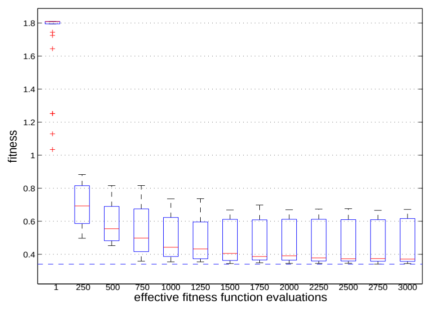


(c) 5,30-Group topology & 30 particles

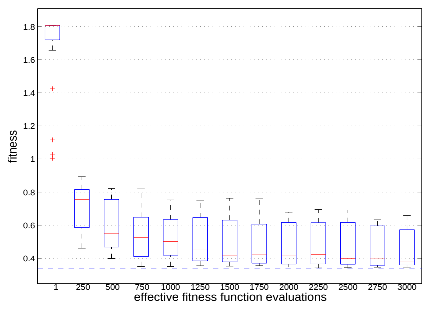
Figure 45: TP2 Results (CCPSO - Archive)



(a) All topology & 30 particles

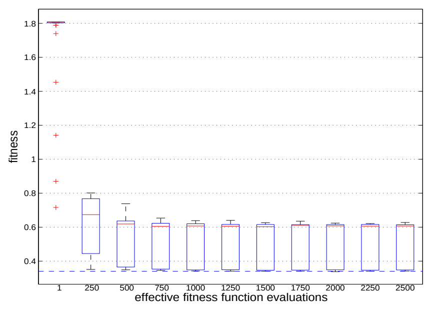


(b) Ring topology & 30 particles

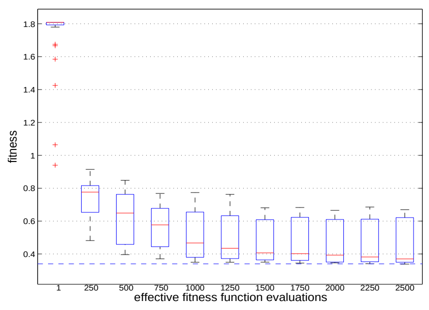


(c) 5,30-Group topology & 30 particles

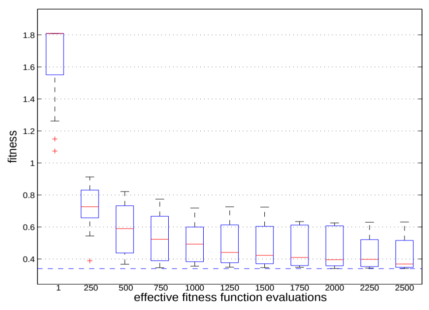
Figure 46: TP3 Results (CCPSO - Sampling)



(a) All topology & 30 particles

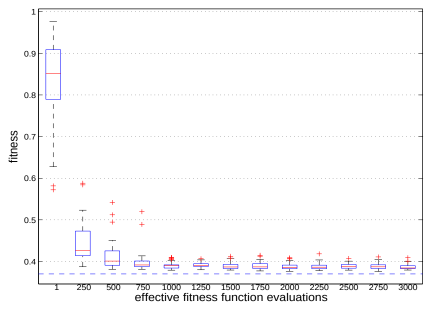


(b) Ring topology & 30 particles

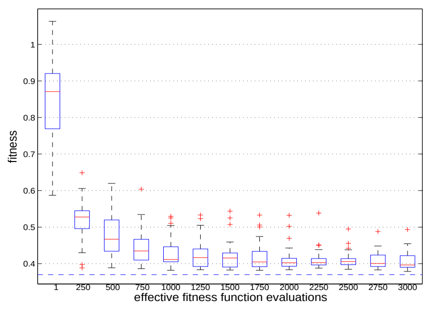


(c) 5,30-Group topology & 30 particles

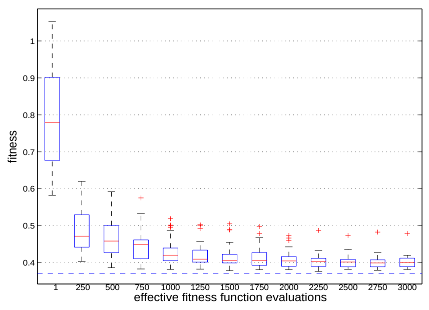
Figure 47: TP3 Results (CCPSO - Archive)



(a) All topology & 30 particles

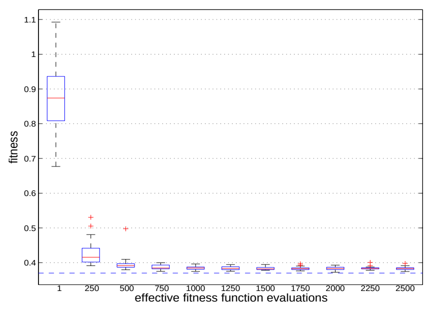


(b) Ring topology & 30 particles

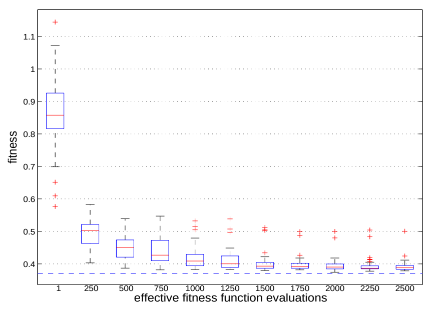


(c) 5,30-Group topology & 30 particles

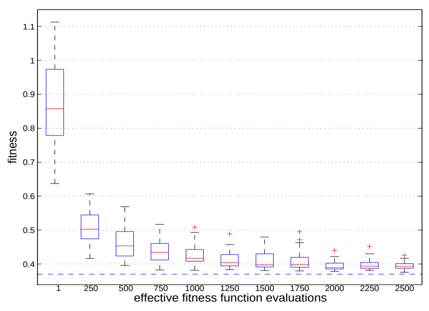
Figure 48: TP4 Results (CCPSO - Sampling)



(a) All topology & 30 particles

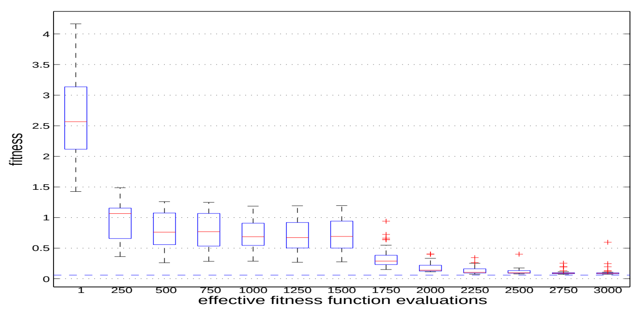


(b) Ring topology & 30 particles

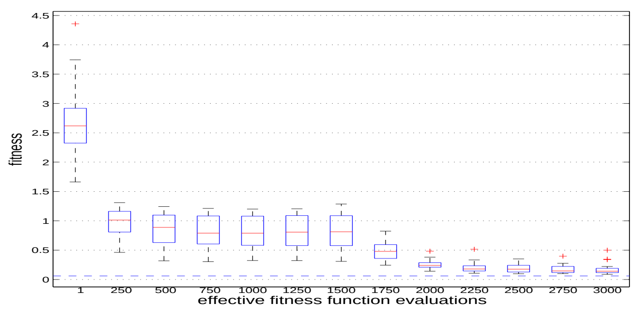


(c) 5,30-Group topology & 30 particles

Figure 49: TP4 Results (CCPSO - Archive)

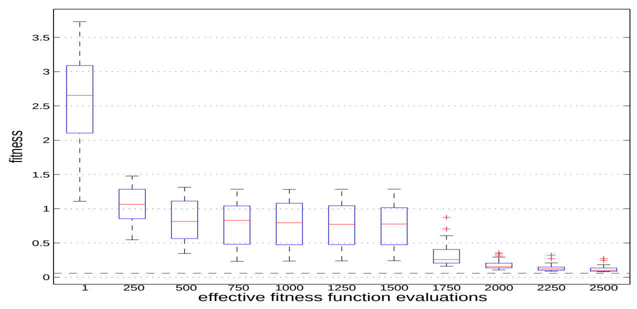


(a) All topology & 30 particles

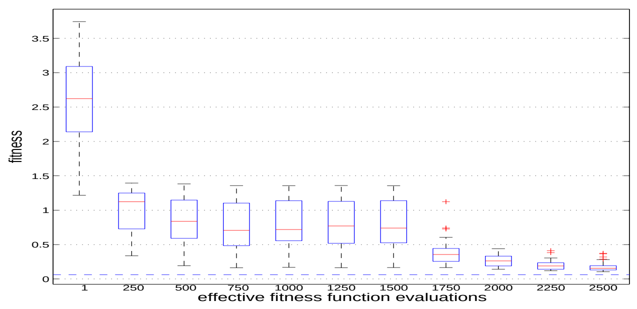


(b) Ring topology & 30 particles

Figure 50: TP1 Results (MPSO - Sampling)

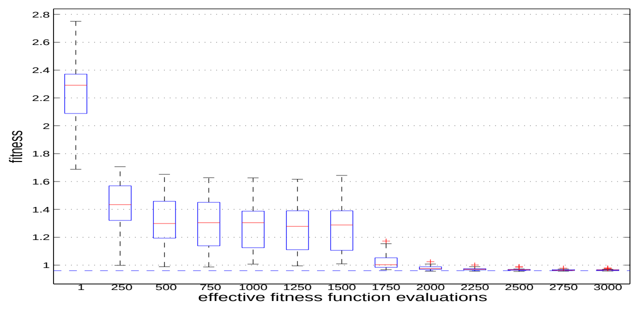


(a) All topology & 30 particles

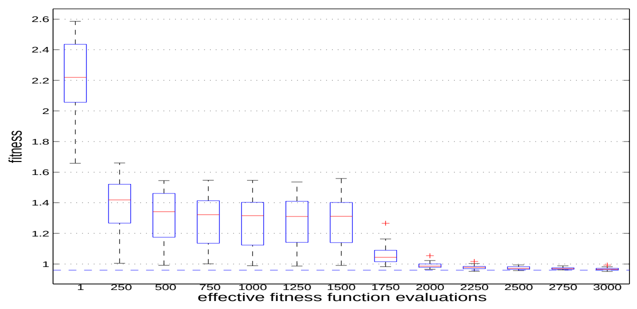


(b) Ring topology & 30 particles

Figure 51: TP1 Results (MPSO - Archive)

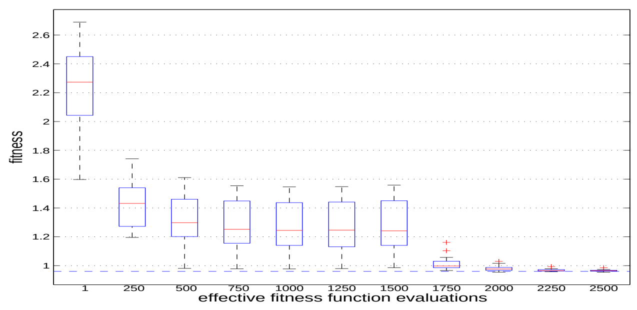


(a) All topology & 30 particles

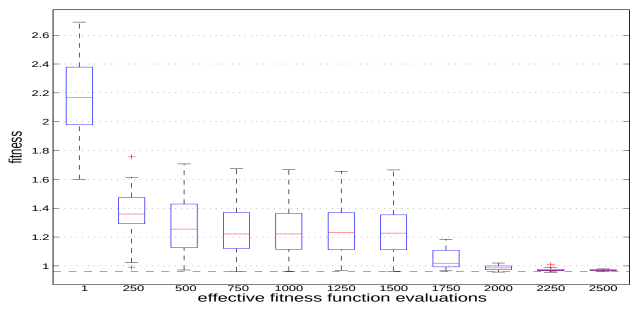


(b) Ring topology & 30 particles

Figure 52: TP2 Results (MPSO - Sampling)

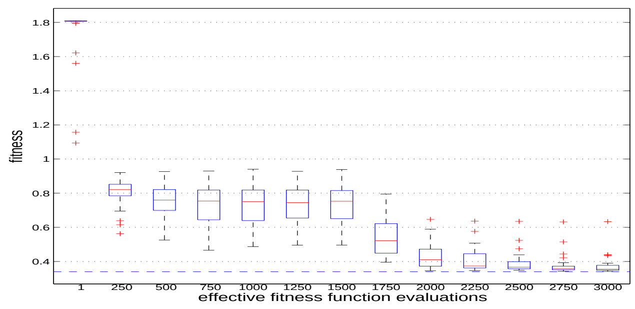


(a) All topology & 30 particles

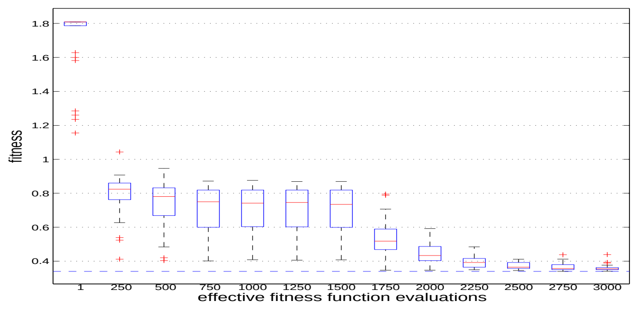


(b) Ring topology & 30 particles

Figure 53: TP2 Results (MPSO - Archive)

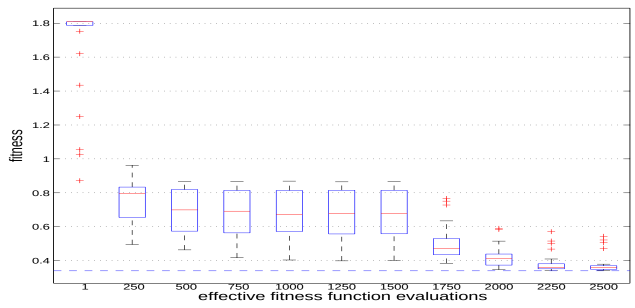


(a) All topology & 30 particles

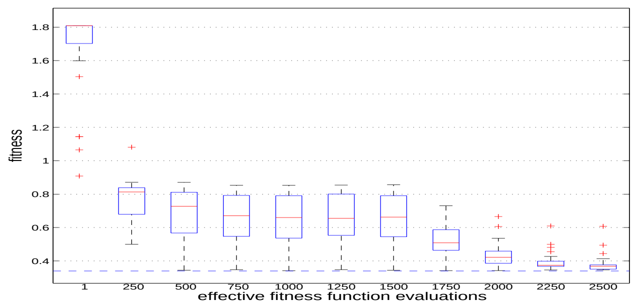


(b) Ring topology & 30 particles

Figure 54: TP3 Results (MPSO - Sampling)

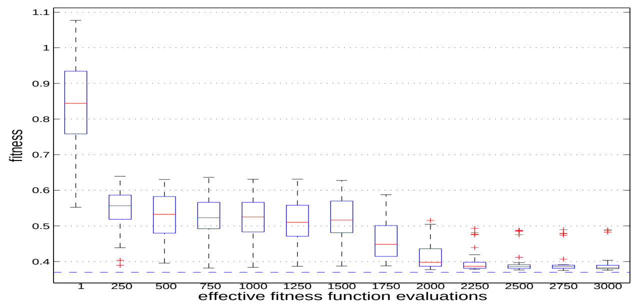


(a) All topology & 30 particles

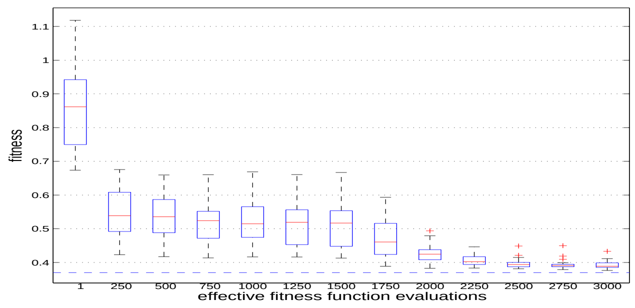


(b) Ring topology & 30 particles

Figure 55: TP3 Results (MPSO - Archive)

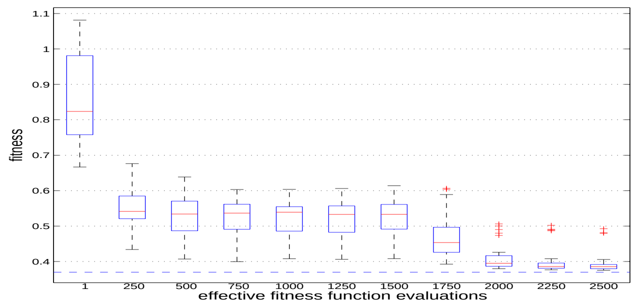


(a) All topology & 30 particles

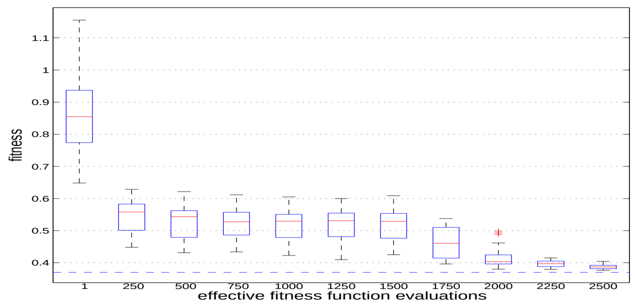


(b) Ring topology & 30 particles

Figure 56: TP4 Results (MPSO - Sampling)

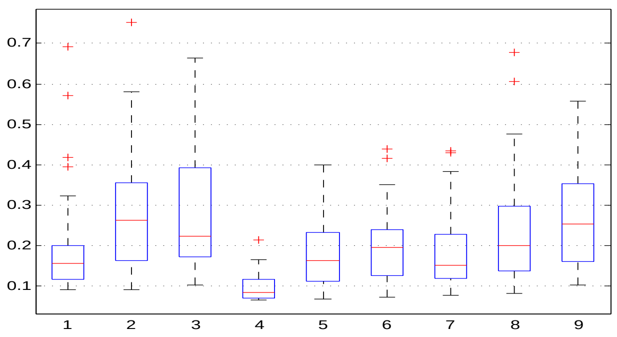


(a) All topology & 30 particles

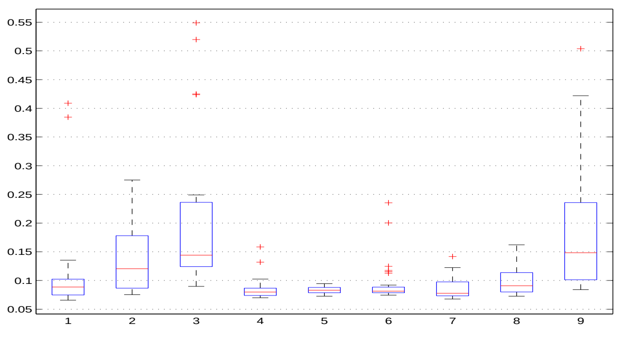


(b) Ring topology & 30 particles

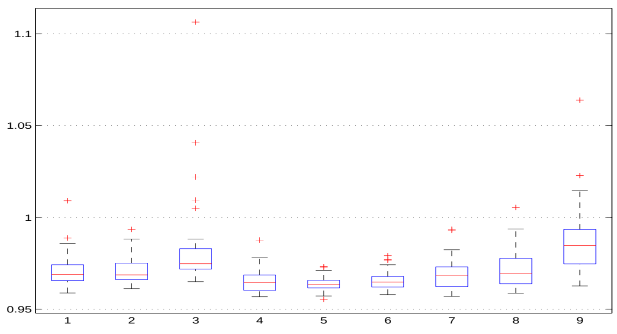
Figure 57: TP4 Results (MPSO - Archive)



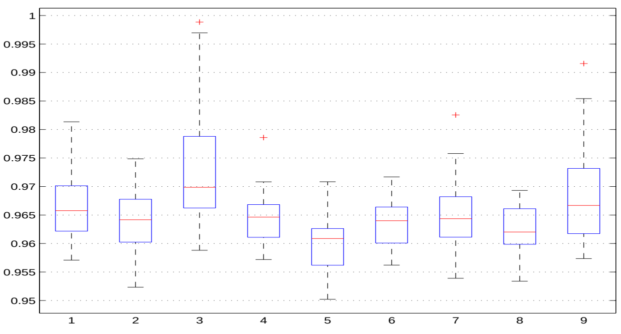
(a) TP1



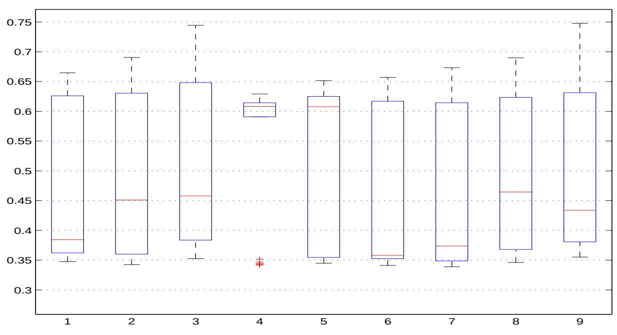
(a) TP1



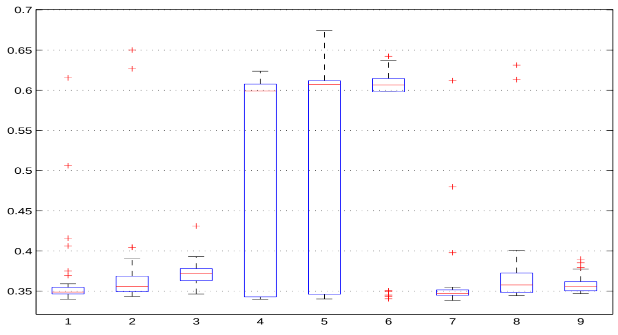
(b) TP2



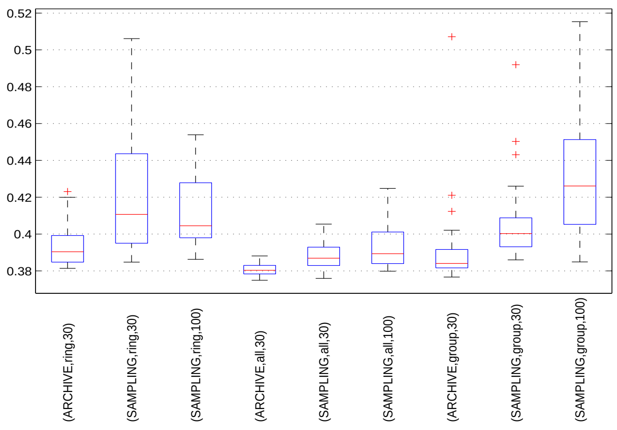
(b) TP2



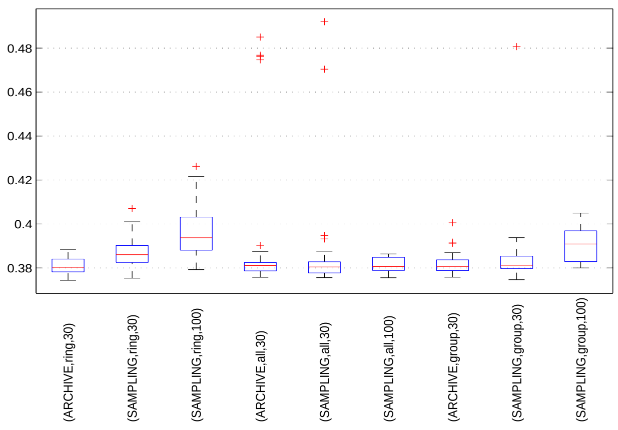
(c) TP3



(c) TP3



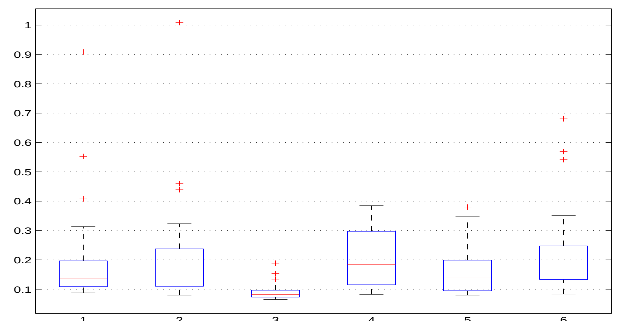
(d) TP4

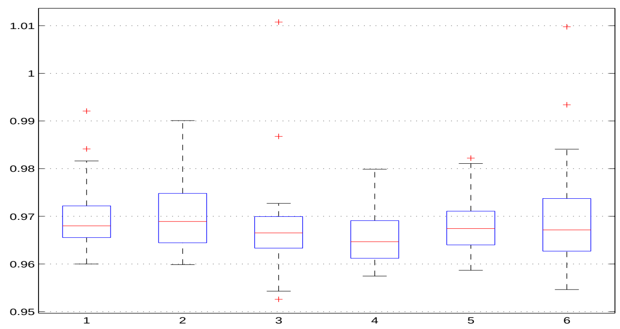


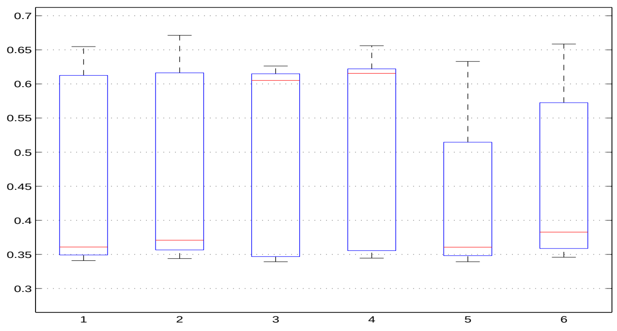
(d) TP4

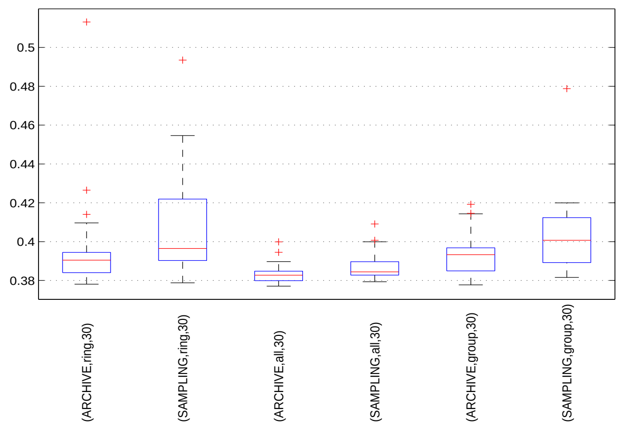
Figure 58: CPSO boxplot end results

Figure 59: FIPS boxplot end results


(a) TP1

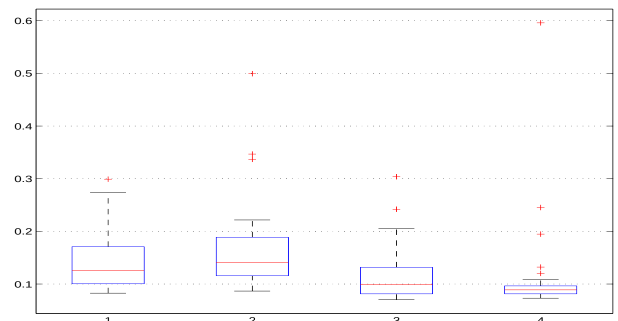

(b) TP2

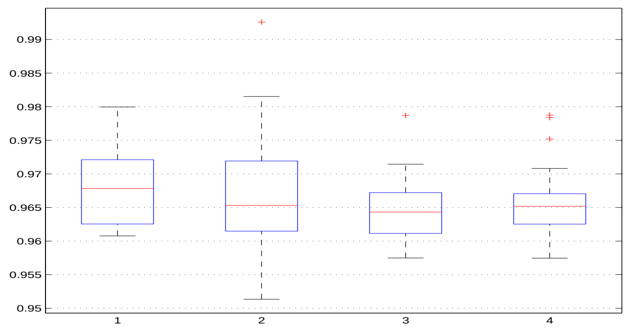

(c) TP3

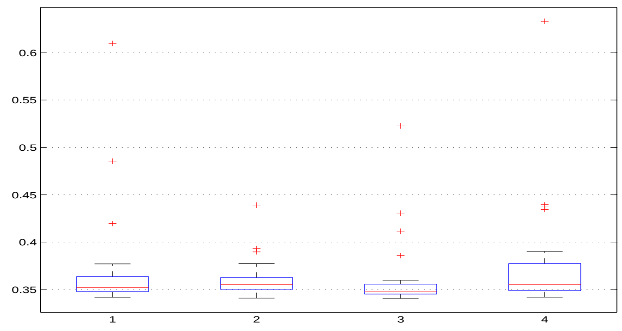


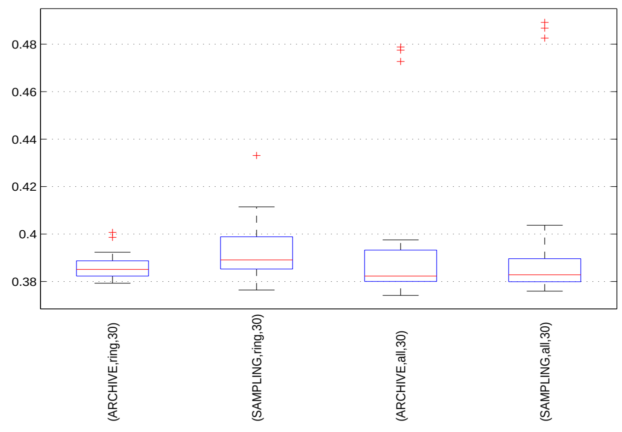
(d) TP4

Figure 60: CCPSO boxplot end results


(a) TP1

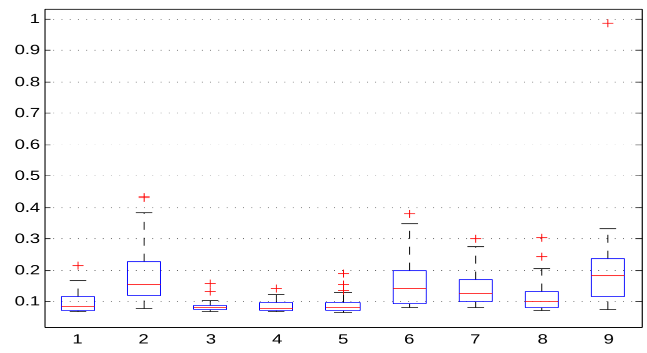

(b) TP2


(c) TP3

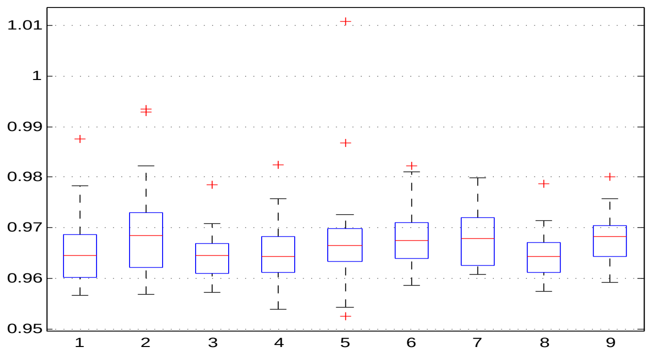


(d) TP4

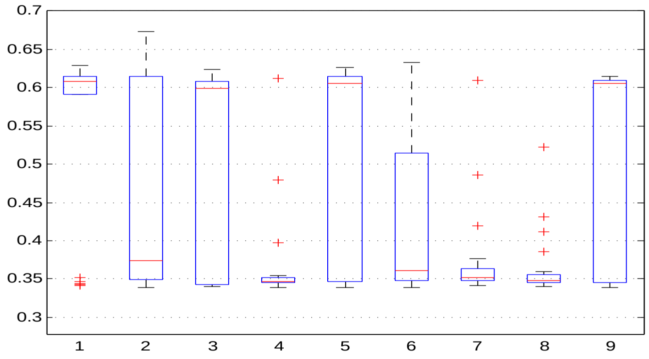
Figure 61: MPSO boxplot end results



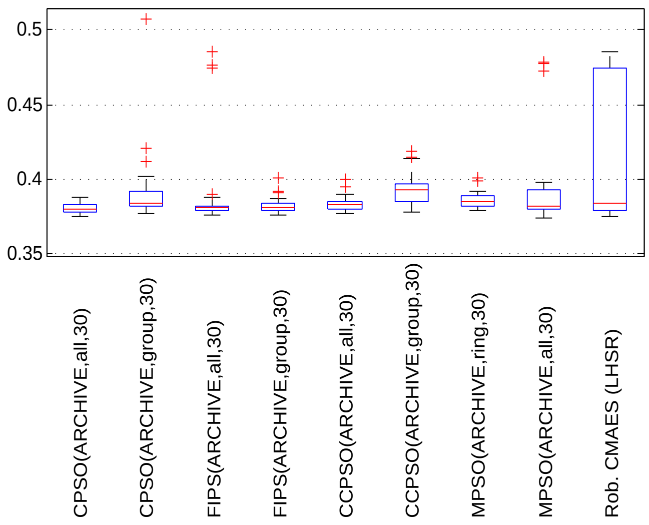
(a) TP1



(b) TP2



(c) TP3



(d) TP4

Figure 62: Comparison boxplot end results
