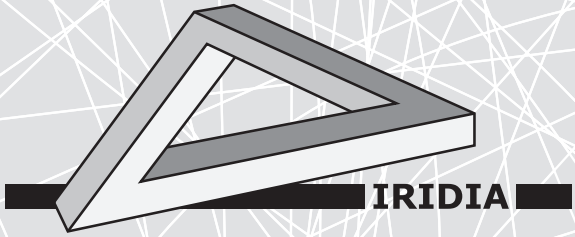


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Incremental Social Learning in Particle Swarms

Marco A. Montes de Oca, Thomas Stützle,
Ken Van den Enden, and Marco Dorigo

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2009-002

January 2009

IRIDIA – Technical Report Series
ISSN 1781-3794

Published by:
IRIDIA, *Institut de Recherches Interdisciplinaires et de Développements en Intelligence Artificielle*
Université Libre de Bruxelles
Av F. D. Roosevelt 50, CP 194/6
1050 Bruxelles, Belgium

Technical report number TR/IRIDIA/2009-002

Revision history:
TR/IRIDIA/2009-002.001 January 2009

Incremental Social Learning in Particle Swarms

Marco A. Montes de Oca[a] mmontes@ulb.ac.be
Thomas Stützle[a] stuetzle@ulb.ac.be
Ken Van den Enden[b] kvdenden@vub.ac.be
Marco Dorigo[a] mdorigo@ulb.ac.be

[a]IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
[b]Vrije Universiteit Brussel

January 2009

Abstract

We have recently proposed two algorithms for continuous optimization that combine aspects of social learning and swarm intelligence. One of these algorithms, called IPSO, is a particle swarm optimizer with a growing population size in which new particles are initialized using a rule that biases their initial random position towards the best so far solution. The other algorithm, called IPSOLS, is an extension of IPSO which allows particles to improve their previous best position by local search.

In this paper, we improve the efficiency of IPSOLS by using the local search procedure only when it is expected to be profitable. We assess its performance by comparing it with fixed and growing population size particle swarm optimization algorithms with and without local search as well as with a random restart local search algorithm. Moreover, we measure the contribution of the algorithms' core components to their overall performance on a family of problems with different fitness distance correlations.

The results of the conducted experiments and of the analysis of the initialization rule show that an incremental growth of the population size in combination with a local search procedure produces a competitive algorithm that is capable of finding good solutions very rapidly without compromising its global search capabilities on problems with positive fitness distance correlation.

1 Introduction

In particle swarm optimization (PSO) algorithms [22, 23, 15, 7, 35, 13], the size of the population is a parameter that directly affects the tradeoff between solution quality and speed. During the first stages of the optimization process, small populations usually make faster improvements than large populations. However, if given enough time, large populations usually return solutions of better quality [40, 28]. This observation motivated us to study a PSO algorithm with a growing population size (hereafter referred to as Incremental PSO or simply IPSO) [27]. In IPSO, every time a new particle is added to the swarm,

its position is not initialized at random, but rather it is initialized using a rule that induces a bias toward the best particle of the swarm. With the proper population growth rate, IPSO's solution quality development over time behavior follows closely that of PSO algorithms with different constant population sizes. IPSO's results can be seen as if multiple PSO algorithms with different constant population sizes were run simultaneously, thus obtaining a better overall tradeoff between speed and solution quality.

In [30], we extended IPSO by making particles call Powell's direction set method [36] as a local search procedure. The local search procedure provides rapid convergence to good solutions (usually local optima) while the PSO component enables a comprehensive exploration of the search space (thus making it possible to escape from low-quality local optima). This extended algorithm (hereafter referred to as IPSOLS) outperforms IPSO and constant population size PSO algorithms with and without local search.

In this paper, the following contributions are presented:

- We modify IPSOLS as defined in [30] in order to improve its efficiency (Section 2). In contrast with the previous version, the modified IPSOLS now invokes the local search procedure only when it is expected to be profitable. The performance of the algorithm is compared with fixed and growing population size PSO algorithms with and without local search as well as with a random restart local search algorithm (Section 3).

- The probability density function induced by the initialization rule used to position the newly added particles in the search space is derived. We complement this analytical result with Monte Carlo simulations of some scenarios that may be encountered in practice (Section 4).

- We measure the contribution of IPSO and IPSOLS core components to their overall performance using a family of problems with different fitness distance correlations based on the Rastrigin function (Section 4). This last analysis shows that the benefits of using an incremental approach in combination with a local search are significant in problems with a fitness distance correlation in the range (0,1).

Although the algorithms mentioned above stand by themselves, their design can be better understood if the framework on which they are based is presented as well (Section 2). IPSO and IPSOLS are based on a multiagent learning framework that we call *incremental social learning* [27]. This framework, which tackles the interference problem caused by the coexistence of multiple simultaneously adapting agents, is inspired by theoretical and empirical studies of social learning in animals [17].

2 Background

The basic PSO algorithm and the incremental social learning framework are described in Sections 2.1 and 2.2 respectively. The two algorithms that are the basis of this paper are described in Section 2.3.

2.1 Particle Swarm Optimization

In PSO parlance, a *swarm* is a group of *particles* that move in the search space $\Theta \subseteq \mathbb{R}^n$ of an optimization problem $f : \Theta \rightarrow \mathbb{R}$ with the goal of finding an optimal solution.[1] A particle's position represents a candidate solution to the problem under consideration. At each iteration of the basic PSO algorithm, every particle is attracted toward its own previous best position and toward the best position found by the particles in its neighborhood. A particle's neighborhood is a subset of the swarm usually defined before the algorithm is run. At the swarm level, neighborhood relations define a population topology which can be seen as a graph $G = \{V, E\}$, where each vertex in V corresponds to a particle in the swarm and each edge in E establishes a neighbor relation between a pair of particles. The equations that govern the movement of a particle i along the j-th dimension of the problem's search space are the following:

$$v_{i,j}^{t+1} = \chi \left[v_{i,j}^{t} + \varphi_1 U_1 \left(p_{i,j}^{t} - x_{i,j}^{t} \right) + \varphi_2 U_2 \left(l_{i,j}^{t} - x_{i,j}^{t} \right) \right] , \tag{1}$$

and

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1}, \tag{2}$$

where $v_{i,j}^{t}$ and $x_{i,j}^{t}$ are the particle's velocity and position at time step t respectively, $p_{i,j}^{t}$ is the particle's best position so far, $l_{i,j}^{t}$ is the best position found by the particle's neighbors, φ_1 and φ_2 are two parameters (called *acceleration coefficients*), U_1 and U_2 are two uniformly distributed random numbers in the range $[0, 1)$, and χ is a constriction factor, proposed by Clerc and Kennedy [8], that guarantees convergence in the search space (not necessarily to a local or global optimum) if the following conditions are met: $\chi = 2k / \left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|$, where $k \in [0, 1]$, and $\varphi = \varphi_1 + \varphi_2 > 4$.

2.2 Incremental Social Learning

In a multiagent environment, learning the state-action mappings of all the participating agents at the same time is a difficult task. The difficulty stems from the exponential growth of the space of combined state-action mappings of all the participating agents as a function of the number of agents [19]. If agents are equipped with their own learning mechanism, then the difficulty arises from the interference caused by the coexistence of multiple simultaneously adapting agents whose rewards depend on the group's performance [33]. In [27], an incremental population approach, called *incremental social learning* (ISL), was proposed as a way to reduce the adverse effects that result from this interference without modifying the agents' default learning mechanism.

ISL is based on the notion that individuals that acquire knowledge from others do not incur the costs of acquiring it individually [24]. In a growing population this means that new individuals that learn from more experienced individuals effectively take a "shortcut" to adaptive information, which they would have to learn by themselves by trial and error otherwise. In fact, in nature, this mechanism allows newborn animals to learn many skills very rapidly from the adult individuals that surround them, increasing in this way their chances of survival [17]. ISL makes naive individuals save time (and in some systems, energy) that they can use to perform their tasks or to learn new things.

[1] Without loss of generality, in this paper we focus on the minimization case.

ISL's algorithmic structure is outlined in Algorithm 1 below.

Algorithm 1 Incremental social learning.

```
/* Initialization */
t ← 0
Initialize environment E^t
Initialize primogenial population of agents X^t

/* Main loop */
while Stopping criteria not met do
  if Agent addition schedule or criterion is not met then
    X^{t+1} ← ilearn(X^t, E^t) /* Individual or default learning mechanism */
  else
    Create new agent a_new
    slearn(a_new, X^t) /* Social learning mechanism */
    X^{t+1} ← X^t ∪ {a_new}
  end if
  E^{t+1} ← update(E^t) /* Update environment */
  t ← t + 1
end while
```

The environment and the primogenial population of agents are initialized before the main loop begins. If, according to an agent addition schedule, no agents are to be added, the agents in the current population learn either individually or using another default learning mechanism. The agent addition schedule controls the rate at which agents are added to the main group and it also creates time delays that allow the agents in the main population to learn from the interaction with the environment and with other agents. Before becoming part of the main population, new agents learn socially from a subset of the already experienced agents. In this way, much of the effort spent by these agents in learning by themselves can be saved for the new agents. In Algorithm 1 above, the environment is updated explicitly in order to note the fact that the environment may be dynamic (although it need not be). In a real implementation, the environment can change at any time and not necessarily at the end of a training round.

The actual implementations of the individual (or default) and social learning mechanisms are independent of the incremental social learning framework outlined above. Both generic or application-specific mechanisms may be used.

2.3 Incremental Social Learning-based PSO algorithms

The framework described above has been instantiated in the context of population-based optimization algorithms [27, 30], where "learning" is understood as the search mechanism whereby new candidate solutions are tried and, eventually, better solutions are found. The resulting algorithms have a growing population size but differ in the default learning mechanism employed. In one of them, called IPSO, particles explore the search space only through the application of the traditional PSO rules [27]. In the other algorithm, called IPSOLS, particles move using the traditional PSO rules and by invoking a local search procedure [30]. The local search procedure employed in IPSOLS is the well-known

Powell's direction set method [36] using Brent's technique [4] as the auxiliary line minimization algorithm. The decision to use Powell's method as the local search procedure was based on performance considerations. IPSOLS calls repeatedly the local search procedure and thus an efficient one is needed. In this sense, Powell's method quadratic convergence can be very advantageous if the objective function is locally quadratic [38].

According to the ISL framework, every time a new agent is added to the main population, it should learn socially from a subset of the more experienced agents. In the case of IPSO and IPSOLS, this means that every time a new particle is added, it should be initialized using information from particles that are already part of the main population. This mechanism is implemented as an initialization rule that moves the new particle from its initial random position in the search space to one that is closer to the position of a particle that serves as a "model" to imitate. The particle that is used as a model in IPSO and IPSOLS is the current best one of the swarm, that is, the new particle is moved closer to the best-so-far solution. The rule, as applied to a new particle's j-th dimension, is the following:

$$x'_{new,j} = x_{new,j} + U \cdot (p_{model,j} - x_{new,j}), \tag{3}$$

where $x'_{new,j}$ is the new particle's updated position, $x_{new,j}$ is the new particle's original random position, $p_{model,j}$ is the model's position and U is a uniformly distributed random number in the range $[0, 1)$. Once the rule is applied for each dimension, the new particle's previous best position is initialized to the point $\boldsymbol{x}'_{new}$. The best particle's previous best position is used as attractor because it stores the best-so-far solution. The random number U is the same for all dimensions in order to ensure that the new particle's updated previous best position will lie somewhere along the direct attraction vector $\boldsymbol{p_{model} - x_{new}}$. Using independent random numbers for each dimension would reduce the strength of the bias induced by the initialization rule because the resulting attraction vector would be rotated and scaled with respect to the direct attraction vector.

Originally, a particle in IPSOLS invoked the local search procedure once every iteration in an attempt to improve its previous best position. In the case a solution is not in a local optimum, the local search procedure can improve the solution further in subsequent calls. However, if the solution is already located in a local optimum, this strategy can result in a waste of function evaluations (particularly in high dimensions). In this paper, we modify IPSOLS so that the local search procedure is called only when it is expected to be beneficial; that is, the local search procedure is invoked only when a particle's previous best position is not already in a local optimum. This is achieved by letting the local search procedure return a value indicating whether it finished because of a very small difference between two solutions, or because the maximum number of iterations was reached. In the first case, it is assumed that the local search has converged to a local optimum and the particle does not invoke the procedure again because no further improvements are expected in that situation. In the second case, the particle may call the local search procedure again because further improvements can still be achieved. The two parameters of the local search procedure that control these exit criteria are the tolerance and the maximum number of iterations respectively. IPSO and the new IPSOLS algorithms are sketched in Algorithm 2. The differences between these two algorithms are indicated with boldface comments.

Algorithm 2 ISL-based PSO algorithms

Input: Objective function $f : \Theta \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$, the initialization domain $\Theta' \subseteq \Theta$, the agent addition criterion A, the local search procedure parameters (tolerance, maximum number of iterations, step size) and the parameters used by the PSO rules (φ_1, φ_2, and χ).
Output: The best found solution $\boldsymbol{sol} \in \Theta$

/* Initialization */
$t \leftarrow 0$
$k \leftarrow 1$ /* Population size */
Initialize position vector $\boldsymbol{x}_k^t$ to random values within Θ'
Initialize velocity vector $\boldsymbol{v}_k^t$ to zero
$\boldsymbol{p}_k^t \leftarrow \boldsymbol{x}_k^t$
$e_k \leftarrow$ **true** /* If $e_k =$ **true**, a local search should be invoked for particle k (**only in IPSOLS**) */

/* Main Loop */
repeat
  /* Individual learning (**only in IPSOLS**) */
  for $i = 1$ to k **do**
    if $e_k =$ **true then**
      $e_k \leftarrow$ localsearch$(f, \boldsymbol{p}_k^t)$ /* Returns **true** if exited without converging, else returns **false** */
    end if
  end for

  /* Horizontal social learning */
  for $i = 1$ to k **do**
    Generate $\boldsymbol{x}_k^{t+1}$ using Eqs. 1 and 2
    if $f(\boldsymbol{x}_k^{t+1}) < f(\boldsymbol{p}_k^t)$ **then**
      $\boldsymbol{p}_k^{t+1} \leftarrow \boldsymbol{x}_k^{t+1}$
      $e_k \leftarrow$ **true** /* **only in IPSOLS** */
    end if
  end for

  /* Population growth and vertical social learning */
  if Agent addition criterion A is met **then**
    Initialize vector $\boldsymbol{p}_{k+1}^{t+1}$ using Eq. 3 for each component
    Initialize velocity vector $\boldsymbol{v}_{k+1}^{t+1}$ to zero
    $\boldsymbol{x}_{k+1}^{t+1} \leftarrow \boldsymbol{p}_{k+1}^{t+1}$
    $e_{k+1} \leftarrow$ **true** /* **only in IPSOLS** */
    $k \leftarrow k + 1$
  end if
  $t \leftarrow t + 1$
  $\boldsymbol{sol} \leftarrow \underset{i \in \{1,\ldots,k\}}{\text{argmin}} \; f(\boldsymbol{p}_i^t)$
until Stopping criterion is satisfied

In the main loop, there are two common structures in both IPSO and IPSOLS and one that is exclusive to IPSOLS. The common structures are the "horizontal social learning" and the "population growth and vertical social learning" parts. They are labeled in this way in order to distinguish between the two patterns of social learning simulated in these algorithms. Horizontal social learning occurs between individuals of a same generation while vertical social learning occurs between individuals of different generations [5]. The local search part of IPSOLS is labeled as "individual learning" because in this part solution improvement can occur in the absence of any social influence.

3 Experimental Evaluation: Setup and Results

In this section, we describe the experimental setup used to assess the performance of the algorithms described in the previous section as well as the results obtained. Preliminary results were published in [27, 30].

3.1 Setup

To assess the operation of IPSO and IPSOLS, their performance was compared to that of the following algorithms:

1. A traditional particle swarm optimization algorithm with constant population size (labeled PSO). This algorithm was included in the evaluation because it allows us to measure the contribution of the incremental population component that IPSO has. Three population sizes of 10, 100 and 1000 particles were used.

2. A hybrid particle swarm optimization algorithm with local search (labeled PSOLS). It is a constant population size particle swarm algorithm in which the particles' previous best positions undergo an improvement phase (via Powell's method) before the velocity update rule is applied. The local search is only applied when a particle's previous best position is not located in a local optimum, just as is done in the new version of IPSOLS (described in the previous section). Three population sizes of 5, 10, and 20 particles were used. Larger populations would have prevented us from observing any of the effects that are due to the interaction of the PSO and local search components. The reason is that given the number of function evaluations required by each invocation of the local search procedure and the maximum number of function evaluations allocated for each experiment (see below), a large population would essentially behave as a random restart local search (a method that was included in the comparison). This algorithm was included in the evaluation because, by comparing its performance to that of IPSOLS, we can measure the contribution of the incremental population component in combination with a local search procedure.

3. A random restart local search algorithm (labeled RLS). Every time the local search procedure (Powell's method) converges, it is restarted from a newly generated random solution. The best solution found so far is considered to be the output of the algorithm. This algorithm was considered because, by comparing its performance with that of IPSOLS, we can

measure the contribution of the PSO search mechanism in the case of a repeated application of a local search from different starting solutions.

In [27], IPSO proved to work well with a fast agent addition schedule. Accordingly, we used a fast particle addition schedule in which a new particle was added every iteration (this schedule was also used in IPSOLS). The maximum size of the swarm was set to 1000 particles. In all experiments, the best particle of the swarm served as a model to imitate.

All particle swarm-based algorithms (PSO, IPSO, PSOLS and IPSOLS) were run with two population topologies: a fully connected topology, in which each particle is a neighbor to all others including itself, and the so-called ring topology, in which each particle has two neighbors apart from itself. In the incremental algorithms, the new particle is randomly placed within the topological structure. The parameter settings used for the PSO rules were: $\varphi_1 = \varphi_2 = 2.05$ and $\chi = 0.7298$.

Powell's method also has a number of parameters to be defined. The tolerance, which is used to stop the procedure once a very small difference between two solutions is detected, was set to 0.01. In case such a difference is not found, the procedure is stopped after a maximum number of iterations is reached. In our experiments, this parameter was set to 10. Other settings were explored but no significant differences in the results were found. Finally, the step size (used for the line minimization procedure) was set to 20% of the length of the corresponding search range. This value was found to yield good results in [30].

For the first set of experiments, we used ten commonly used benchmark functions. Their mathematical definition can be found at this paper's supplementary information webpage [29][2].

Our results are based on statistics taken from 100 independent runs each of which was stopped whenever one of the following criteria was met: (i) 10^6 function evaluations have been performed, or (ii) the solution value is less than or equal to 10^{-15}. In each run, most benchmark functions were randomly shifted within the specified range. The exceptions are Schwefel's and Step problems because their optima is not at the origin of the coordinate system. In all cases, we used the functions 100-dimensional versions, that is, $n = 100$. Candidate solutions beyond the allowed search range were forced to stay within the search region by putting them on the boundary.

3.2 Results

In Figure 1, we show the solution value obtained with the compared algorithms after 10^6 function evaluations. The plots shown in this figure correspond to the case in which all PSO-based algorithms were run with a fully-connected topology. Asterisks on the top of the boxplots indicate the cases in which statistically significant differences exist between the indicated algorithm and IPSOLS (upper row or red asterisks) or IPSO (lower row or blue asterisks).[3] Figure 2 shows the same kind of results as Figure 1 but with PSO-based algorithms using a ring topology.

[2] At this same address, the reader can find the complete set of plots and data that, for the sake of conciseness, are not included in the paper. Nevertheless, the main results are discussed in the text.

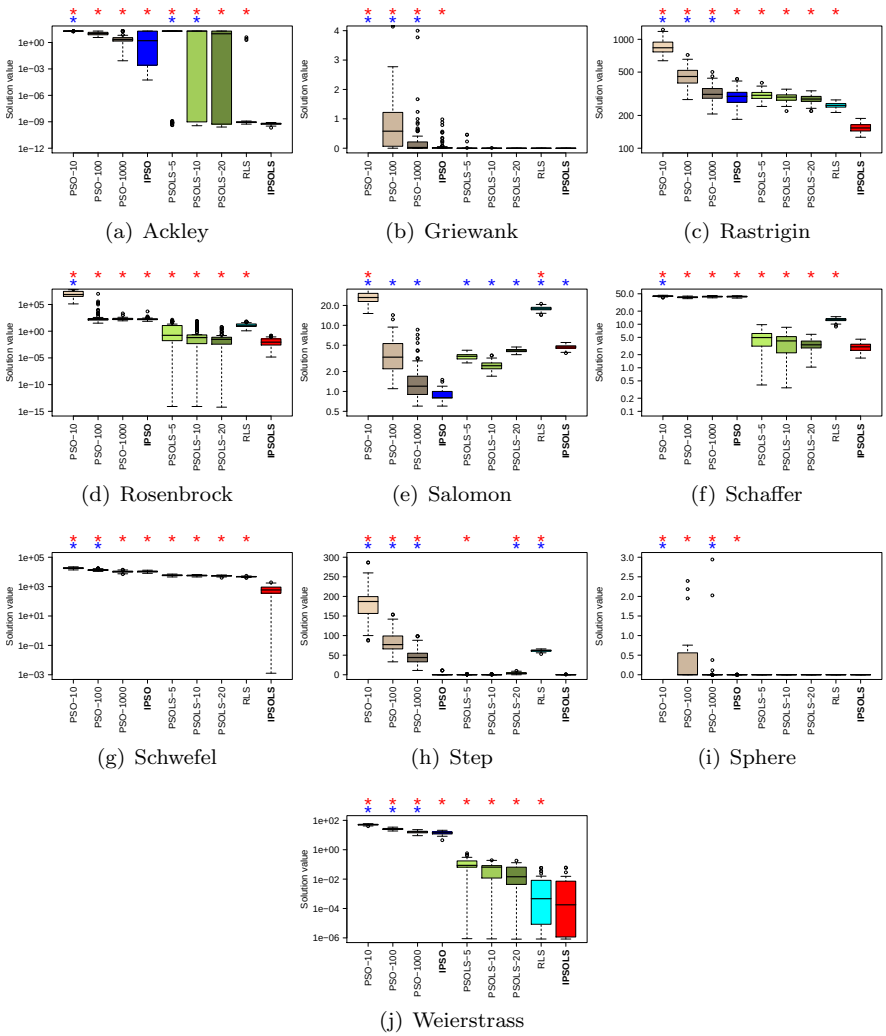
[3] Significance determined at the 5% level using a Wilcoxon test using Holm's correction method for multiple comparisons.

If the results of the fixed population size PSO algorithms are compared with those obtained with IPSO, it is possible to observe how IPSO obtains solutions of comparable (if not better) quality. For example, IPSO typically obtains better results than a PSO algorithm with 10 particles. In our results, the only exception can be seen in Figure 2(f) where the difference is not statistically significant. In 8 out of the 20 cases shown in Figures 1 and 2, IPSO obtains better results than any of the fixed population size PSO algorithms. In most of the other cases, IPSO obtains solutions of similar quality to that of solutions obtained with fixed population size PSO algorithms. This trend is also observed in the results that correspond to shorter runs (the supporting data can be found in [29]).

The local search component plays a major role on the performance of the algorithms that include it. Figures 1 and 2 show that the quality of the solutions obtained with the algorithms that include a local search procedure is usually higher than the one of the solutions obtained with the algorithms that do not. The difference in performance due to the use of a local search procedure is particularly contrasting in the case of IPSO and IPSOLS. This last algorithm outperformed IPSO in 16 out of the 20 possible cases. In fact, IPSOLS outperformed all other algorithms in 9 cases. As it was the case with IPSO and the constant population size PSO algorithms without local search, IPSOLS obtains solutions of comparable, if not better, quality to those obtained with the best hybrid PSO–local search algorithm with constant population size. However, for very short runs (in our case, runs of some hundreds of function evaluations), IPSOLS performs worse than IPSO and the hybrid PSO–local search algorithms (data not shown). In the first case, this is because Powell's method has to perform at least n line searches (n is the number of dimensions of the problem) before making any significant improvement. In the second case, this is due to the fact that PSOLS takes advantage of the random initialization of the population.

IPSOLS is effectively an algorithm that calls repeatedly a local search procedure from different initial solutions. In this respect, IPSOLS works in the same way as a random restart local search algorithm (RLS). However, the main difference between these two algorithms resides in the way the new initial solutions are chosen. In RLS this choice is made at random; in IPSOLS it is the result of the application of the PSO rules. Thus, a comparison of the results obtained with these two algorithms can give us an indication of the impact of the PSO component in IPSOLS. The results presented in Figures 1 and 2 indicate that IPSOLS outperforms RLS in all problems except on Griewank, Sphere and Weierstrass. In the two first cases, the local search procedure alone is able to make such a significant progress that indeed it is able to find the minimum of the Sphere function (with a solution value that is less than or equal to 10^{-15}, one of our stopping criteria). In the case of the Griewank function, IPSO solves the problem with a population of around 3 particles (data shown in [29]). Thus, IPSOLS's behavior is essentially equivalent to that of RLS when its population does not grow significantly (see, for example, Figure 4). This behavior is not exhibited by PSOLS which is outperformed by RLS in eight occasions.

The new version of IPSOLS outperforms the original one [30] in 13 out of the 20 possible cases (data shown in [29]). Among the cases in which the new version cannot find better solutions than the original one, we find the Griewank and Sphere functions. Only in one case, the Rosenbrock function, the original version performs better than the new one.



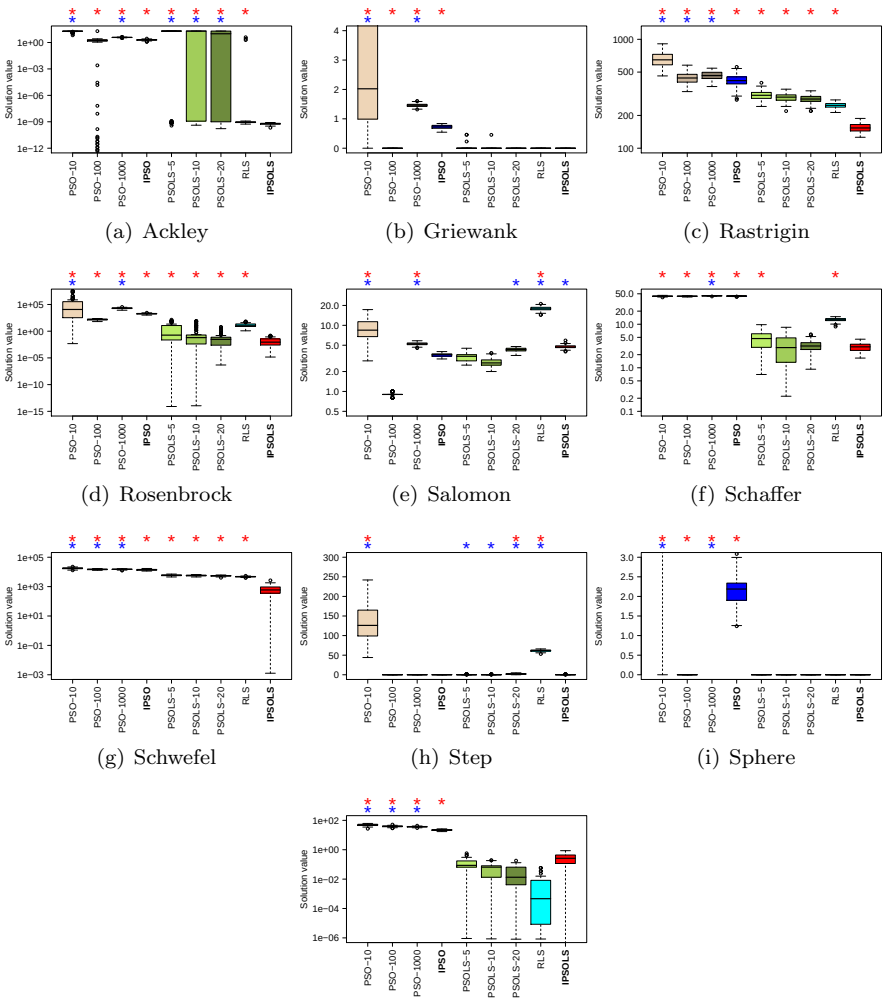
(a) Ackley (b) Griewank (c) Rastrigin

(d) Rosenbrock (e) Salomon (f) Schaffer

(g) Schwefel (h) Step (i) Sphere

(j) Weierstrass

Figure 1: The boxplots show the distribution of the solution quality obtained with the compared algorithms after 10^6 function evaluations. These results correspond to the case in which a fully-connected topology is used with all particle swarm-based algorithms. In the Griewank and Sphere functions, the solution values obtained with the traditional PSO algorithm with 10 particles are so much higher than those obtained with other algorithms that its boxplot does not appear. An asterisk on top of a boxplot denotes a statistically significant difference at the 5% level between the results obtained with the indicated algorithm and those obtained with IPSOLS (red asterisks or upper row) or IPSO (blue asterisks or lower row).



(a) Ackley (b) Griewank (c) Rastrigin

(d) Rosenbrock (e) Salomon (f) Schaffer

(g) Schwefel (h) Step (i) Sphere

(j) Weierstrass

Figure 2: The boxplots show the distribution of the solution quality obtained with the compared algorithms after 10^6 function evaluations. These results correspond to the case in which a ring topology is used with all particle swarm-based algorithms. In the Griewank and Sphere functions, the solution values obtained with the traditional PSO algorithm with 10 particles are so much higher than those obtained with other algorithms that its boxplot is not depicted completely. An asterisk on top of a boxplot denotes a statistically significant difference at the 5% level between the results obtained with the indicated algorithm and those obtained with IPSOLS (red asterisks or upper row) or IPSO (blue asterisks or lower row).

As examples of the behavior of the algorithms over time, consider the results shown in Figures 3 and 4, which correspond to the solution development over time obtained by a selection of the compared algorithms on the Rastrigin and Sphere functions. In these figures, we also show the average population size growth over time in IPSO and IPSOLS.

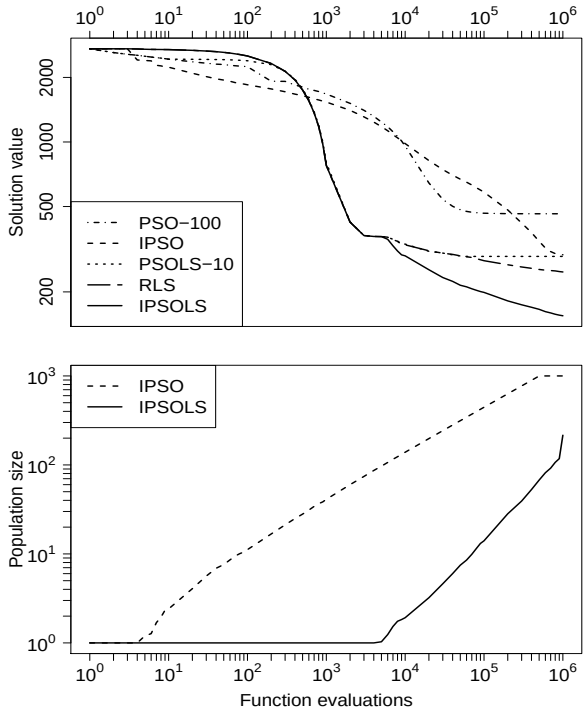


Figure 3: Solution development over time obtained by a selection of the compared algorithms (PSO-based algorithms using a fully-connected topology) on the Rastrigin function (upper figure). Average population size growth in IPSO and IPSOLS (lower figure).

In some cases, as was noted before, IPSOLS is outperformed by other algorithms for short runs (in our case, runs between 10^2 and 10^3 function evaluations). However, IPSOLS improves dramatically once the population size starts growing, as exemplified by the plots in Figure 3 in which IPSOLS starts differentiating from RLS and PSOLS after approximately 5,000 function evaluations. IPSOLS improves rapidly once the local search procedure begins to make progress as seen in Figure 4. In this last figure, it can be seen how the population in IPSOLS, when it is applied to the Sphere function, does not grow. This explains the equivalence of IPSOLS and RLS on problems solvable by local search alone. In most cases, the population growth in IPSOLS is independent of the population topology used (data shown in [29]).

The results obtained with the traditional PSO algorithm for shorter runs confirm the tradeoff between solution quality and speed that we mentioned in the introduction. Swarms of 10 particles find better solutions than larger swarms after up to around 10^4 function evaluations. Then, the swarms of 100

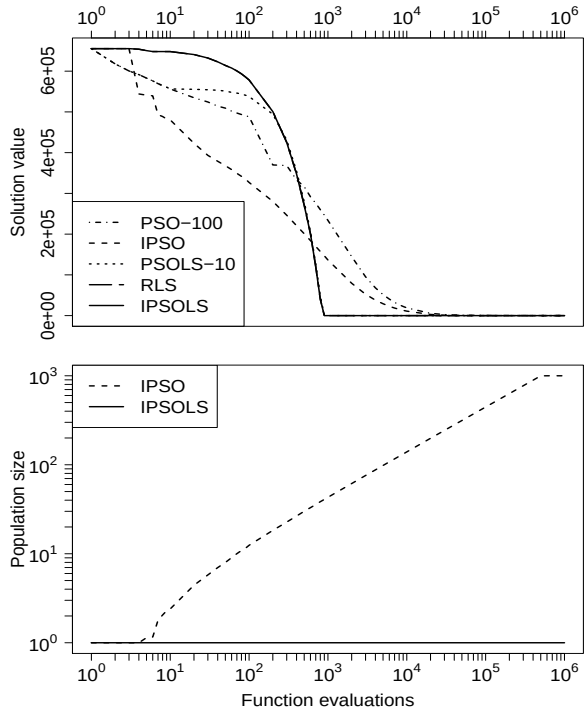


Figure 4: Solution development over time obtained by a selection of the compared algorithms (PSO-based algorithms using a fully-connected topology) on the Sphere function (upper figure). Average population size growth in IPSO and IPSOLS (lower figure).

particles are the best performing after 10^5 function evaluations, and finally, after 10^6 function evaluations, the swarms with 1000 particles usually return the best solutions. The solution development over time obtained with IPSO follows closely the one that could be obtained by a constant population size PSO that is able to get the best solution quality for a given number of function evaluations (data shown in [29]).

From a practitioner's point of view, the advantage of using IPSOLS over a hybrid PSO algorithm are at least two: (i) IPSOLS does not require the practitioner to fix the population size in advance hoping to have chosen the right size for his/her problem, and (ii) IPSOLS is more robust to the choice of the population's topology. The difference between the results obtained with IPSOLS with a fully-connected and with a ring topology are smaller than the differences observed in the results obtained with the hybrid algorithms.

4 Analysis of the new particles' initialization rule

In IPSO and IPSOLS, the initial position of a new particle is not generated randomly but by using an initialization rule whose goal is to simulate the phenomenon of vertical social learning. This rule biases the position of the newly created particle towards the position of a particle that serves as an attractor or "model". In Section 4.1, we present the exact probability density function induced by the initialization rule and describe its behavior and some of its properties. In Section 4.2, we measure the impact of the initialization rule on the performance of IPSO and IPSOLS.

4.1 Probability density function

IPSO and IPSOLS initialization rule (Eq. 3) is a function of two random variables: the uniformly distributed original position and a uniformly distributed random number in the range $[0, 1)$ that determines the strength of the attraction towards the position of the particle used as a model (the best particle in the swarm in our case). The model's position is, strictly speaking, also a random variable due to the fact that it is the result of a number of iterations of the PSO position-update mechanism. However, when the initialization rule is invoked, it can be taken as a constant.

It is of interest to know the probability density function that is induced by Eq. 3 over the initialization range in order to better understand the effects of the new particles' initialization rule, and ultimately of the incremental social learning approach applied to optimization.

The probability density function induced by the new particles' initialization rule for dimension j is the following (its derivation is shown in Appendix A):

$$f_{X_j}(x_j) = \frac{1}{x_{max,j} - x_{min,j}} \cdot \begin{cases} \ln \left| \frac{p_{model,j} - x_{min,j}}{p_{model,j} - x_j} \right| & \text{if } x_j < p_{model,j} \\ 0 & \text{if } x_j = p_{model,j} \\ \ln \left| \frac{p_{model,j} - x_{max,j}}{p_{model,j} - x_j} \right| & \text{if } x_j > p_{model,j} \end{cases}, \quad (4)$$

where $x_{min,j}$ and $x_{max,j}$ are the minimum and maximum limits of the initialization range over the problem's jth dimension and $x_{min,j} \leq x_j < x_{max,j}$.

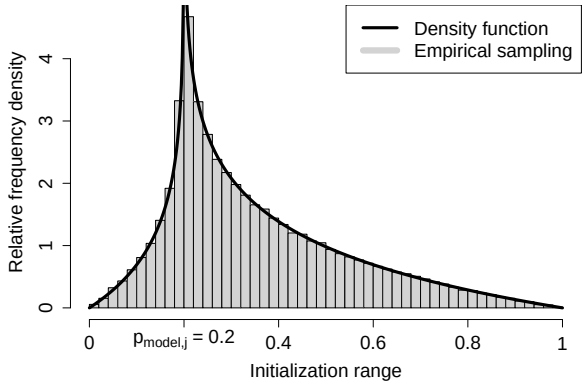


Figure 5: Probability density function induced by the initialization rule of new particles. In the figure, the attractor $p_{model,j} = 0.2$. The initialization range is $[0, 1)$. The figure shows both the analytical density function and the histogram obtained using a Monte Carlo simulation (10^5 samples).

Figure 5 shows the exact density function and a histogram obtained using a Monte Carlo simulation when the initialization range is $[0, 1)$ and $p_{model,j} = 0.2$.

Most of the samples concentrate around the model's position as desired. Note, however, that there is a nonzero probability of sampling regions far away from the model. This behavior ensures that there is a certain level of exploration by initialization. The probability density function is not symmetric except in the case $p_{model,j} = (x_{max,j} + x_{min,j})/2$. The expected value of a new particle's position is the following:

$$\begin{aligned} E(x'_{new,j}) &= E(x_{new,j}) + E(U)\left(p_{model,j} - E(x_{new,j})\right) \\ &= \frac{x_{max,j} + x_{min,j}}{2} + \frac{1}{2}\left(p_{model,j} - \frac{x_{max,j} + x_{min,j}}{2}\right) \\ &= \frac{x_{max,j} + x_{min,j}}{4} + \frac{p_{model,j}}{2}\,. \end{aligned} \tag{5}$$

The analysis presented above is valid only if the attractor particle's position is within the range $[x_{min,j}, x_{max,j})$. If it is outside the initialization range, the probability density function remains the same within the range but it turns into a uniform distribution outside of it as shown in Figure 6.

Under these conditions, a new particle will follow the model only from one of its sides. The initialization rule is not able to position a new particle beyond the location of the attractor particle if this particle is outside the original initialization range. This is not a drawback because, in a practical scenario, the initialization region is usually the region within which the optimum is expected to be.

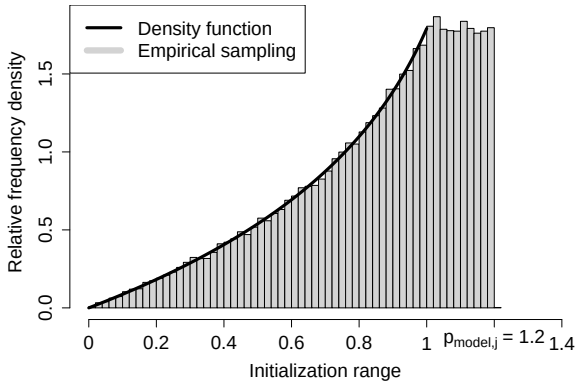


Figure 6: Probability density function induced by the initialization rule of new particles when the attractor lies outside the original initialization range. In the figure, the attractor $p_{model,j} = 1.2$. The initialization range is $[0, 1)$. The figure shows that the density function follows the analytical density function up to the limit of the original initialization range. The histogram obtained using a Monte Carlo simulation (10^5 samples) shows the actual density function.

4.2 Effect size analysis of the bias strength toward the best-so-far solution

In this section, we measure the extent to which the new particles' initialization rule affects the quality of the solution obtained after a certain number of function evaluations with respect to a random initialization. For this purpose, IPSO and IPSOLS are run with initialization mechanisms that induce a bias of different strength towards the best particle of the swarm. These mechanisms are (in increasing order of bias strength): (i) random initialization, (ii) the initialization rule as defined in Eq. 3 (labeled as "weak bias"), and (iii) the same as in (ii) but with the random number U drawn from a uniform distribution in the range $[0.95, 1)$ (labeled as "strong bias").

The experiments are carried out on problems derived from the Rastrigin function, each of which has different fitness distance correlation (FDC) [21]. Since the initialization rule used in IPSO and IPSOLS implicitly assumes that good solutions are close to each other, the hypothesis is that the performance of the algorithms degrades as the problem's FDC approaches zero and that the rate of performance degradation is faster with stronger initialization bias.

The Rastrigin function, whose n-dimensional formulation is $nA+\sum_{i=1}^{n}(x_i^2 - A\cos(\omega x_i))$, can be thought of as a parabola with a superimposed (co)sine wave with an amplitude and frequency controlled by parameters A and ω respectively. By changing the values of A and ω one can obtain a whole family of problems. In our experiments, we set $\omega = 2\pi$ as is usually done, and tuned the amplitude A to obtain functions with specific FDCs. Other settings are the search range and the dimensionality of the problem which we set to $[-5.12, 5.12]^n$ and $n = 100$, respectively. The amplitude and the resulting FDCs (estimated using 10^4 random samples spread over the whole search range) are shown in Table 1.

Table 1: Amplitudes used in the Rastrigin function to obtain specific fitness distance correlations (FDCs).

Amplitude	FDC	Amplitude	FDC
155.9111	≈ 0.001	13.56625	≈ 0.6
64.56054	≈ 0.1	10.60171	≈ 0.7
40.09456	≈ 0.2	7.938842	≈ 0.8
28.56419	≈ 0.3	5.21887	≈ 0.9
21.67512	≈ 0.4	0.0	≈ 0.999
16.95023	≈ 0.5	-	-

IPSO and IPSOLS with the three initialization rules described above, were run 100 times on each problem for up to 10^6 function evaluations. To measure the magnitude of the effect of using one or another initialization scheme, we use Cohen's d statistic [10] which for the case of two samples is defined as follows:

$$d = \frac{\hat{\mu_1} - \hat{\mu_2}}{\sigma_{pooled}}\,, \tag{6}$$

with

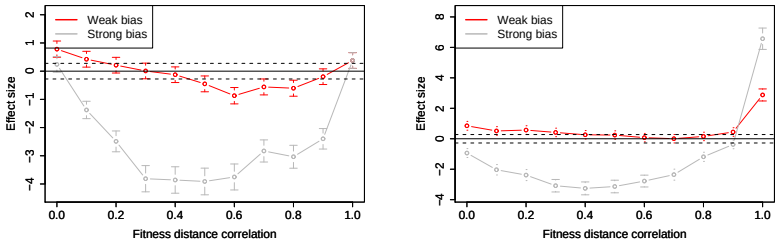
$$\sigma_{pooled} = \sqrt{\frac{(n_1 - 1)\hat{\sigma_1}^2 + (n_2 - 1)\hat{\sigma_2}^2}{n_1 + n_2 - 2}}\,, \tag{7}$$

where $\hat{\mu_i}$ and $\hat{\sigma_i}$ are the mean and standard deviation of sample i, respectively [31].

As an effect size index, Cohen's d statistic measures the difference between the mean responses of a treatment and control groups expressed in standard deviation units [39]. The treatment group is, in our case, the set of solutions obtained with IPSO and IPSOLS using the initialization rule that biases the position of a new particle toward the one of the best particle of the swarm. The control group is the set of solutions obtained with IPSO and IPSOLS when new particles are initialized completely at random. (Since in our case the lower the solution value the better, the order of the operands in the subtraction is reversed.) An effect size value of 0.8, for example, means that the average solution found using the particles' initialization rule is better than 79% of the solutions found without using it. The practical significance that the value associated to an effect has depends, of course, on the situation at hand; however, a value of 0.8 can already be considered a large effect [10].

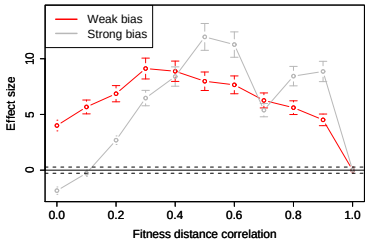
The observed effect sizes with 95% confidence intervals on the solution quality obtained with IPSO and IPSOLS after 10^6 function evaluations are shown in Figure 7.

In IPSO, the effects of using the new particles' initialization rule are very different from the ones in IPSOLS. In IPSO, the weak bias initialization rule produces better results than random initialization only in two cases: (i) when the problem's FDC is almost equal to one and the algorithm is run with a ring topology, and (ii) when the problem's FDC is close to zero irrespective of the population topology used. In all other cases, the weak bias initialization rule produces results similar to those obtained with a random initialization. The strong bias initialization rule reports benefits only in the case of a high FDC and a ring topology. In all other cases, it results in significantly worse solutions than

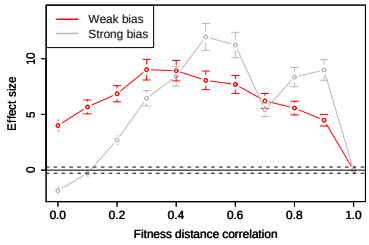


(a) IPSO, Fully-connected topology

(b) IPSO, Ring topology



(c) IPSOLS, Fully-connected topology



(d) IPSOLS, Ring topology

Figure 7: Effect size of the new particles initialization rule, as measured using Cohen's d statistic with 95% confidence intervals (indicated by either error bars or dashed lines), on the solution quality obtained with IPSO and IPSOLS after 10^6 function evaluations. Two bias strengths are tested: (i) weak bias and (ii) strong bias. The reference results (line at zero) are obtained with a random initialization.

the ones obtained with a random initialization. The worst performance of IPSO with the strong bias initialization rule is obtained when the problem's FDC is in the range (0.3,0.6). This behavior is a consequence of the new particle's velocity being equal to zero, which effectively reduces the particle's initial exploratory behavior. Setting the new particle's initial velocity to a value different from zero reduces the effect of the initialization bias because it would immediately make the particle move to a quasi-random position right after the first iteration of the algorithm's PSO component. The performance observed when the problem's FDC is close to zero is the result of the fact that with a fixed search range and a high amplitude, the parabolic component of the Rastrigin function has a much lower influence and many of the locally optimal solutions are of the same quality, thus moving close to or away from already good solutions has no major impact on the solution quality.

While the effect in IPSO is only positive in a few cases, in IPSOLS the effect size is not only positive in almost all cases but it is also large. IPSOLS with the weak bias initialization rule produces significantly better solutions than with a random initialization in all but one case, which corresponds to the situation where the problem's FDC is close to one. When the strong bias initialization rule is used, IPSOLS produces better solutions than with random initialization when the problem's FDC is in the range (0.1, 1.0). In the range (0.4,1.0), the solutions obtained with a strong bias initialization rule are better than or equal to those obtained with a weak bias initialization rule.

The fact that in IPSOLS using a random initialization of new particles is effectively the same as initializing them with a bias toward the location of the best particle of the swarm when the problem's FDC is almost one can be easily explained: under these circumstances IPSOLS is a local search algorithm that starts from a single solution. Since a local search algorithm alone can solve a problem with an FDC close to one, no population growth occurs and the initialization rule is never used. The degradation of the effect size as the problem's FDC decreases can be observed in the range (0.0,0.5) for the strong bias initialization rule, and in the range (0.0, 0.3) for the weak bias initialization rule. As hypothesized, the rate of degradation is faster when using a strong bias.

In summary, the use of the weak bias initialization rule in IPSOLS, which is the originally proposed vertical social learning rule, provides significant benefits over random initialization on problems with a fitness distance correlation in the range (0,1).

5 Related Work and Discussion

In the context of multiagent systems, the idea of letting the size of the population grow over time has been applied by Noble and Franks [32]. They note that newborn animals may benefit from the observation of elder individuals and implemented a simulation where the population of agents grows over time. This idea is also exploited in the definition of the incremental social learning framework presented in [27]. By making an analogy with a multiagent system, we have devised two population-based optimization algorithms, IPSO and IPSOLS, which can be seen as a population of agents with the capability of learning from others, and in the case of IPSOLS, also by themselves. However, the evaluation of the performance of these algorithms lets us conclude that IPSOLS is more

promising than IPSO.

In IPSOLS, the population size is increased if local search alone cannot solve the problem at hand satisfactorily, either because it has converged to a local optimum or because the maximum number of iterations of the local search procedure has been reached. When a new particle is added to the main population, a sort of "vertical social learning" is simulated. We have shown that not only increasing the population size is beneficial but also that the biased initialization of new particles results in improved performance. This may be an indication that the analogy between a multiagent system and a population of particles in a PSO algorithm is stronger than it appears at first sight. Further research needs to be done in order to establish the strength of the relationship between the problem faced by multiple coexisting learning agents and a population of agents searching for an optimum in a swarm-based optimization algorithm.

Two of the main algorithmic components integrated in IPSOLS, namely the time-varying population size and the subsidiary local search procedure, have been a topic of research in the past. Population (re)sizing has been studied within the field of evolutionary computation for many years. From that experience, it is now usually accepted that the population size in evolutionary algorithms should be proportional to the problem's difficulty [26]. The issue is that we usually know little about a problem's difficulty *a priori*. The approach taken in the design of IPSOLS (i.e., to start with a small population and let it grow over time) makes sense because if the problem is not so difficult, it may be solved by local search alone. If the problem is difficult, a growing population size will offer a better tradeoff between solution quality and speed than a constant population size PSO-based algorithm.

Practically all resizing strategies found in the literature consider the possibility of reducing the size of the population during an algorithm's run (see e.g. [1, 20, 3, 14, 16, 9, 25, 12]). An exception to this approach is the work of Auger and Hansen [2] in which the population size of a CMA-ES algorithm is doubled each time it is restarted. The feature that IPSOLS shares with Auger and Hansen's approach is that no population reduction mechanism is implemented; however, IPSOLS does not use restarts and has a slower population growth rate. Nevertheless, it is possible that the performance of IPSOLS can improved by decreasing the population size from time to time because this may save many function evaluations. Such a change is not expected to affect negatively the quality of the solutions found by the algorithm if the removed particles are part of a group that has converged to a local minimum. This topic is left as future work.

The idea of combining local search techniques with particle swarm optimization algorithms comes from the observation that particles are attracted by their previous best positions. It is usually thought that the better the attractors are, the higher the chances to find even better solutions. The goal of most hybrid algorithms is thus to accelerate the placement of the particles' previous best positions in good spots. For example, Chen et al. [6] combined a particle swarm algorithm with a hill-climbing local search procedure; Gimmler et al. [18] experiment with Nelder and Mead's simplex method as well as with Powell's method; Das et al. [11] also use Nelder and Mead's simplex method and propose the inclusion of an estimate of the local gradient into the particles' velocity update rule. Petalas et al. [34] report experiments with several local search-particle swarm combination schemes. All these previous approaches try to enhance the perfor-

mance of constant population size PSO algorithms but without considering the features of the objective function. In IPSOLS, the incremental population size approach can adapt to the features of the objective function (as exemplified by the Sphere or Griewank functions). Future work related to these issues include testing whether invoking the local search procedure on the particles' current positions can improve the results obtained with the algorithms presented in this and other papers. Also of interest is to use other local search procedures such as Powell's NEWUOA algorithm [37].

The incremental social learning framework is not tied to any particular implementation of the learning mechanisms used. However, more research is needed in order to know whether the framework extends succesfully to other population-based optimization techniques.

6 Conclusions

We have presented a study of two recently proposed algorithms that are based on social learning and swarm intelligence ideas: an incremental particle swarm optimization (IPSO) and a particle swarm-guided local search algorithm (IPSOLS). In the first part of the paper, we have described an extension to one of these algorithms (IPSOLS) that reduces the number of calls to the local search procedure without affecting the algorithm's performance. In the second part of the paper, we have evaluated the performance of the two algorithms on a suite of benchmark problems and measured the contribution of the algorithms' main components to their performance. It has been shown that the effects of simulating the phenomenon of vertical social learning are positive on problems of positive fitness distance correlation.

Acknowledgments

This work was supported by the META-X project, an *Action de Recherche Concertée* funded by the Scientific Research Directorate of the French Community of Belgium. M. A. Montes de Oca acknowledges partial support from the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E05D054889MX. Thomas Stützle and Marco Dorigo acknowledge support from the F.R.S-FNRS of the French Community of Belgium of which they are a Research Associate and a Research Director, respectively.

Appendix A. New particles initialization rule exact probability density function

The position of a newly added particle in IPSO and IPSOLS can be seen as a random variable Z which is a function of two independent continuous random variables X and Y. X is a uniformly distributed random variable in the complete initialization range $[x_{min}, x_{max})$, while Y is a uniformly distributed random variable in the range $[0, 1)$. Z is defined as follows:

$$Z = X + Y(c - X)\,, \tag{A.1}$$

where $x_{min} \leq c < x_{max}$ is a constant representing the location of the attractor particle.

The distribution function F_Z of Z is given by

$$F_Z(a) = P(Z \leq a) = \iint\limits_{(x,y):x+y(c-x)\leq a} f(x,y)\,dx\,dy\,, \tag{A.2}$$

where $f(x,y)$ is the joint probability distribution of X and Y.

Since X and Y are independent, we have that

$$f(x,y) = f_X(x)f_Y(y) = \frac{1}{x_{max} - x_{min}}\,, \tag{A.3}$$

where f_X and f_Y are the marginal probability functions of X and Y respectively. This holds for $x_{min} \leq x < x_{max}$ and $0 \leq y < 1$.

Using A.3 and considering that $y = \frac{a-x}{c-x}$, we can rewrite A.2 as follows

$$\begin{aligned} F_Z(a) &= \frac{1}{x_{max} - x_{min}} \int_{x_{min}}^{x_{max}} y\,dx \\ &= \frac{1}{x_{max} - x_{min}} \int_{x_{min}}^{x_{max}} \frac{a-x}{c-x}\,dx\,. \end{aligned} \tag{A.4}$$

Eq. A.4 must be solved in two parts: when $x_{min} \leq x \leq a < c$ and when $c < a \leq x < x_{max}$. In the special case when $x = c$, $F_Z(a) = c/(x_{max} - x_{min})$ (see Eq. A.1).

When $x_{min} \leq x \leq a < c$, we obtain

$$\begin{aligned} F_Z(a) &= \frac{1}{x_{max} - x_{min}} \int_{x_{min}}^{a} \frac{a-x}{c-x}\,dx \\ &= \frac{1}{x_{max} - x_{min}} \left[a + (a-c)\ln\left|\frac{c - x_{min}}{c-a}\right|\right]\,. \end{aligned} \tag{A.5}$$

When $c < a \leq x < x_{max}$, we obtain

$$\begin{aligned} F_Z(a) &= \frac{1}{x_{max} - x_{min}} \left[1 - \int_{a}^{x_{max}} \frac{a-x}{c-x}\,dx\right] \\ &= \frac{1}{x_{max} - x_{min}} \left[a + (a-c)\ln\left|\frac{c - x_{max}}{c-a}\right|\right]\,. \end{aligned} \tag{A.6}$$

Hence the probability density function f_Z of Z is given by

$$f_Z(z) = \frac{d}{dz}F_Z(z) = \frac{1}{x_{max} - x_{min}} \begin{cases} \ln\left|\frac{c-x_{min}}{c-z}\right| & \text{if } z < c \\ 0 & \text{if } z = c \\ \ln\left|\frac{c-x_{max}}{c-z}\right| & \text{if } z > c \end{cases}\,. \tag{A.7}$$

References

[1] Jaroslaw Arabas, Zbigniew Michalewicz, and Jan J. Mulawka. GAVaPS – A genetic algorithm with varying population size. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 73–78, Piscataway, NJ, 1994. IEEE Press.

[2] Anne Auger and Nikolaus Hansen. A restart CMA evolution strategy with increasing population size. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 1769–1776, Piscataway, NJ, 2005. IEEE Press.

[3] Thomas Bäck, A. E. Eiben, and Nikolai A. L. van der Vaart. An empirical study on GAs "without parameters". In *LNCS 1917. Parallel Problem Solving from Nature - PPSN VI, 6th International Conference*, pages 315–324, Berlin, Germany, 2000. Springer-Verlag.

[4] Richard P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, Englewood Cliffs, NJ, 1973.

[5] Luigi Luca Cavalli-Sforza and Marcus W. Feldman. *Cultural Transmission and Evolution. A Quantitative Approach*. Princeton University Press, Princeton, NJ, 1981.

[6] Junying Chen, Zheng Qin, Yu Liu, and Jiang Lu. Particle swarm optimization with local search. In *Proceedings of the International Conference on Neural Networks and Brain (ICNN&B 2005)*, pages 481–484, Piscataway, NJ, 2005. IEEE Press.

[7] Maurice Clerc. *Particle Swarm Optimization*. ISTE, London, UK, 2006.

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

[9] André L. V. Coelho and Daniel G. de Oliveira. Dynamically tuning the population size in particle swarm optimization. In *Proceedings of the ACM Symposium on Applied Computing (SAC'08)*, pages 1782–1787, New York, NY, 2008. ACM Press.

[10] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Earlbaum Associates, Hillsdale, NJ, 1988.

[11] Sanjoy Das, Praveen Koduru, Min Gui, Michael Cochran, Austin Wareing, Stephen M. Welch, and Bruce R. Babin. Adding local search to particle swarm optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, pages 428–433, Piscataway, NJ, 2006. IEEE Press.

[12] Chen DeBao and Zhao ChunXia. Particle swarm optimization with adaptive population size and its application. *Applied Soft Computing*, 9(1):39–48, 2009.

[13] Marco Dorigo, Marco A. Montes de Oca, and Andries P. Engelbrecht. Particle swarm optimization. *Scholarpedia*, 3(11):1486, 2008.

[14] A. E. Eiben, E. Marchiori, and V. A. Valkó. Evolutionary algorithms with on-the-fly population size adjustment. In *LNCS 3242. Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference*, pages 41–50, Berlin, Germany, 2004. Springer-Verlag.

[15] Andries P. Engelbrecht. *Fundamentals of Computational Swarm Intelligence*. John Wiley & Sons, Chichester, UK, 2005.

[16] Carlos Fernandes and Agostinho Rosa. Self-regulated population size in evolutionary algorithms. In *LNCS 4193. Parallel Problem Solving from Nature - PPSN IX, 9th International Conference*, pages 920–929, Berlin, Germany, 2006. Springer-Verlag.

[17] Bennett G. Galef Jr. and Kevin N. Laland. Social learning in animals: Empirical studies and theoretical models. *BioScience*, 55(6):489–499, 2005.

[18] Jens Gimmler, Thomas Stützle, and Thomas E. Exner. Hybrid particle swarm optimization: An examination of the influence of iterative improvement algorithms on performance. In Marco Dorigo et al., editors, *LNCS 4150. Ant Colony Optimization and Swarm Intelligence. 5th International Workshop, ANTS 2006*, pages 436–443, Berlin, Germany, 2006. Springer-Verlag.

[19] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored MDPs. In Thomas G. Dietterich et al., editors, *Advances in Neural Information Processing Systems 14. Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, pages 1523–1530, Cambridge, MA, 2001. MIT Press.

[20] Georges R. Harik and Fernando G. Lobo. A parameter-less genetic algorithm. In Wolfgang Banzhaf et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999)*, pages 258–265, San Francisco, CA, 1999. Morgan Kaufmann.

[21] Terry Jones and Stephanie Forrest. Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 184–192, San Francisco, CA, 1995. Morgan Kaufmann.

[22] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, 1995. IEEE Press.

[23] James Kennedy and Russell Eberhart. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA, 2001.

[24] Kevin N. Laland. Social learning strategies. *Learning & Behavior*, 32(1):4–14, 2004.

[25] Laura Lanzarini, Victoria Leza, and Armando De Giusti. Particle swarm optimization with variable population size. In R. Goebel et al., editors, *LNAI 5097. Proceedings of the International Conference on Artificial Intelligence and Soft Computing. ICAISC 2008*, pages 438–449, Berlin, Germany, 2008. Springer-Verlag.

[26] Fernando G. Lobo and Claudio F. Lima. *Parameter Setting in Evolutionary Algorithms*, volume 54/2007 of *Studies in Computational Intelligence*, chapter Adaptive Population Sizing Schemes in Genetic Algorithms, pages 185–204. Springer, Berlin, Germany, 2007.

[27] Marco A. Montes de Oca and Thomas Stützle. Towards incremental social learning in optimization and multiagent systems. In W. Rand et al., editors, *Workshop on Evolutionary Computation and Multiagent Systems Simulation of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, pages 1939–1944, New York, NY, 2008. ACM Press.

[28] Marco A. Montes de Oca, Thomas Stützle, Mauro Birattari, and Marco Dorigo. Frankenstein's PSO: A composite particle swarm optimization algorithm. Technical Report TR/IRIDIA/2007-006, IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium, 2007.

[29] Marco A. Montes de Oca, Thomas Stützle, Ken Van den Enden, and Marco Dorigo. Incremental social learning in particle swarms: Complete results, 2009. Supplementary information page at `http://iridia.ulb.ac.be/supp/IridiaSupp2009-001/`.

[30] Marco A. Montes de Oca, Ken Van den Enden, and Thomas Stützle. Incremental particle swarm-guided local search for continuous optimization. In M. J. Blesa et al., editors, *LNCS 5296. Proceedings of the International Workshop on Hybrid Metaheuristics. HM 2008*, pages 72–86, Berlin, Germany, 2008. Springer-Verlag.

[31] Shinichi Nakagawa and Innes C. Cuthill. Effect size, confidence interval and statistical significance: a practical guide for biologists. *Biological Reviews*, 82(4):591–605, 2007.

[32] Jason Noble and Daniel W. Franks. Social learning in a multi-agent system. *Computers and Informatics*, 22(6):561–574, 2003.

[33] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.

[34] Y. G. Petalas, K. E. Parsopoulos, and M. N. Vrahatis. Memetic particle swarm optimization. *Annals of Operations Research*, 156(1):99–127, 2007.

[35] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. An overview. *Swarm Intelligence*, 1(1):33–57, 2007.

[36] M. J. D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162, 1964.

[37] M. J. D. Powell. *Large-Scale Nonlinear Optimization*, volume 83 of *Nonconvex Optimization and Its Applications*, chapter The NEWUOA software for unconstrained optimization, pages 255–297. Springer-Verlag, Berlin, Germany, 2006.

[38] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C. The Art of Scientific Computing*. Cambridge University Press, New York , NY, second edition, 1992.

[39] David J. Sheskin. *Handbook of parametric and nonparametric statistical procedures*. Chapman & Hall/CRC, Boca Raton, FL, second edition, 2000.

[40] Frans van den Bergh and Andries P. Engelbrecht. Effects of swarm size on cooperative particle swarm optimisers. In Lee Spector et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 469–476, San Francisco, CA, 2001. Morgan Kaufmann.