Incremental Particle Swarm-Guided Local Search for Continuous Optimization

Marco A. Montes de Oca[1], Ken Van den Enden[2] and Thomas Stützle[1]

[1] IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{mmontes,stuetzle}@ulb.ac.be
[2] Vrije Universiteit Brussel, Brussels, Belgium
kvdenden@vub.ac.be

Abstract. We present an algorithm that is inspired by theoretical and empirical results in social learning and swarm intelligence research. The algorithm is based on a framework that we call *incremental social learning*. In practical terms, the algorithm is a hybrid between a local search procedure and a particle swarm optimization algorithm with growing population size. The local search procedure provides rapid convergence to good solutions while the particle swarm algorithm enables a comprehensive exploration of the search space. We provide experimental evidence that shows that the algorithm can find good solutions very rapidly without compromising its global search capabilities.

1 Introduction

Many algorithms that have been successfully used for solving optimization problems can be thought of as being a collection of agents that interact with each other and with the environment. In fact, this mental imagery has been used to design some of them. Some examples are genetic algorithms [1], ant colony optimization algorithms [2], and particle swarm optimization algorithms [3]. Following this same approach, we present an algorithm that is inspired by results in social learning and swarm intelligence research. The algorithm is based on a multiagent learning framework that we call *incremental social learning* [4]. The multiagent scenario used for the design of the proposed algorithm is the following: A growing population of agents that can learn both individually and socially explores its environment in order to maximize the group's well-being. The strategy used by the agents is to use their social learning skills when they become part of the population and when learning individually is either too costly or deemed unproductive.

Technically speaking, the algorithm presented in this paper is a hybrid between the well-known Powell's direction set method [5] and a particle swarm optimization algorithm with growing population size [4]. Powell's method is used as an agent's individual learning mechanism. If the local topography of the objective function (from an agent's perspective) allows it, this mechanism improves candidate solutions without any information exchange between agents. The particle swarm velocity- and position-update rules, which allow the global

exploration of the search space, play the role of the agents' social learning mechanism. Together, these components form an incremental particle swarm-guided local search algorithm for solving continuous optimization problems (Section 3).

We compare the performance of the proposed algorithm with that of other four algorithms (Section 4). The proposed algorithm exhibits the fast convergence of Powell's method with good global search capabilities (Section 5).

2 Incremental Social Learning

In previous work, we introduced a multiagent learning framework, called *incremental social learning*, that combines elements of social and individual learning in order to speed up learning and to allow scalability [4]. Social learning is a term that refers to the class of mechanisms that allow the transmission of knowledge between individuals without the use of genetic material [6, 7]. Individual learning, on the other hand, is a process that lets an individual acquire knowledge about its environment by interacting directly with it and without any social influence [8]. Applying social learning ideas to the multiagent learning problem is appealing because learning from others effectively gives agents a shortcut to adaptive information that otherwise would be expensive to acquire [9]. The framework is called incremental because it is based on a growing population of agents that adapts incrementally to the final multiagent environment. When a new agent is added to the population, it learns socially from more experienced agents and learns individually once it is part of it. This strategy is inspired by the fact that, in nature, newborn individuals are particularly favored by social learning because it allows them to learn many skills very rapidly from the adult individuals that surround them [10]. The algorithmic structure of the incremental social learning framework is outlined in Algorithm 1.

At the beginning, the environment and the primogenial population of agents are initialized. An agent addition schedule is used to control the rate at which agents are added to the environment. If no agents are to be added, the agents in the current population learn individually by directly interacting with the environment. If a new agent is scheduled to be part of the population, it learns socially from a subset of the agents in the population before it is added. The environment is then updated (if needed) and the cycle is repeated until a stopping criterion is satisfied.

The learning process starts with a small number of agents because this reduces the interference caused by the co-existence of multiple learning agents. Agents are added according to a problem-dependent schedule in order to create time delays that allow agents that are already part of the population to learn from the environment and to have something to teach to newcomers. When a new agent learns socially, it saves itself the effort required to learn individually what others already have. Incrementally growing the population size is also useful for allocating the minimum number of agents required to solve a particular problem.

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 learning */
  else
    Create new agent a_new
    slearn(a_new, X^t) /* Social learning */
    X^{t+1} ← X^t ∪ {a_new}
  end if
  E^{t+1} ← update(E^t) /* Update environment */
  t ← t + 1
end while
```

3 Incremental Particle Swarm-Guided Local Search

The incremental social learning framework, although conceived for tackling multiagent learning problems, can also be used for designing population-based optimization algorithms. This is because, as we said in Section 1, the multiagent metaphor is usually useful for thinking about how this kind of algorithms work. In [4], a particle swarm optimization algorithm with growing population size (IPSO) was designed as an instantiation of the incremental social learning framework. In IPSO, individual learning is not implemented; instead, the algorithm comprises vertical and horizontal social learning mechanisms. Vertical social learning is a pattern of knowledge transmission between individuals of different generations while horizontal social learning happens between individuals of the same generation [11]. In IPSO, vertical social learning occurs when a new particle is added to the swarm and horizontal social learning is used instead of individual learning. IPSO exhibits a good solution quality vs. time trade-off since it finds solutions of at least the same quality than a particle swarm optimization algorithm with constant population size in fewer objective function evaluations.

Here we present an extension of IPSO that includes an individual learning mechanism. Individual learning is implemented as a local search procedure to simulate a particle's self-improvement process. The addition of an individual learning mechanism is justified not only from a practical point of view but also from a theoretical one. Indeed, since social learning is cheaper than individual learning, it is usually assumed to be always advantageous [9]. However, it has been argued that relying only on socially acquired knowledge is not always the best strategy [12]; instead, individuals should devote some of their time and energy to learn individually or to innovate [9].

As IPSO, the proposed algorithm is based on the particle swarm optimization algorithm in which particles (i.e., potential solutions to an optimization problem) move in the search space with a certain velocity. Each particle is attracted toward its own previous best position (with respect to an objective function) and toward the best position found by the particles in its neighborhood. Neighborhood relations are usually given in advance through a population topology which can be defined by 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 velocity and position updates of a particle i over dimension j are as follows

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

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, U_1 and U_2 are two uniformly distributed random numbers in the range $[0, 1)$, and χ is a constriction factor that is used in order to avoid an "explosion" of the particles' velocity. Clerc and Kennedy [13] found the relation $\chi = 2k / \left| 2 - \varphi - \sqrt{\varphi^2 - 4\varphi} \right|$, where $k \in [0, 1]$, and $\varphi = \varphi_1 + \varphi_2 > 4$, to compute it.

The local search procedure employed in the proposed algorithm is the well-known Powell's direction set method [5] using Brent's technique [14] as the auxiliary line minimization algorithm. The decision of using Powell's method as the local search procedure is based on performance considerations. The proposed algorithm calls repeatedly the local search procedure and thus an efficient one was needed. In this sense, Powell's method quadratic convergence can be very advantageous if the objective function is locally quadratic [15].

The last component of the proposed algorithm is the vertical social learning mechanism which is implemented as a rule that moves the new particle's previous best position from its initial random location in the search space to one that is closer to the previous best position of a particle that serves as a "model" to imitate. The rule is applied in a component-wise fashion as follows

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

where $p'_{new,j}$ is the new particle's updated previous best position, $p_{new,j}$ is the new particle's original previous best position, $p_{model,j}$ is the model's previous best position and U is a uniformly distributed random number in $[0, 1)$.

The combination of the local search procedure and the particle swarm optimization algorithm with growing population size, as sketched in Algorithm 2, produces an incremental particle swarm-guided local search algorithm whose operation is illustrated in Figure 1.

The proposed algorithm is designed for taking advantage of features that the objective function may have. For example, if the objective function is convex

Algorithm 2 Incremental Particle Swarm-Guided Local Search

Input: Objective function f and maximum number of iterations t_{max}

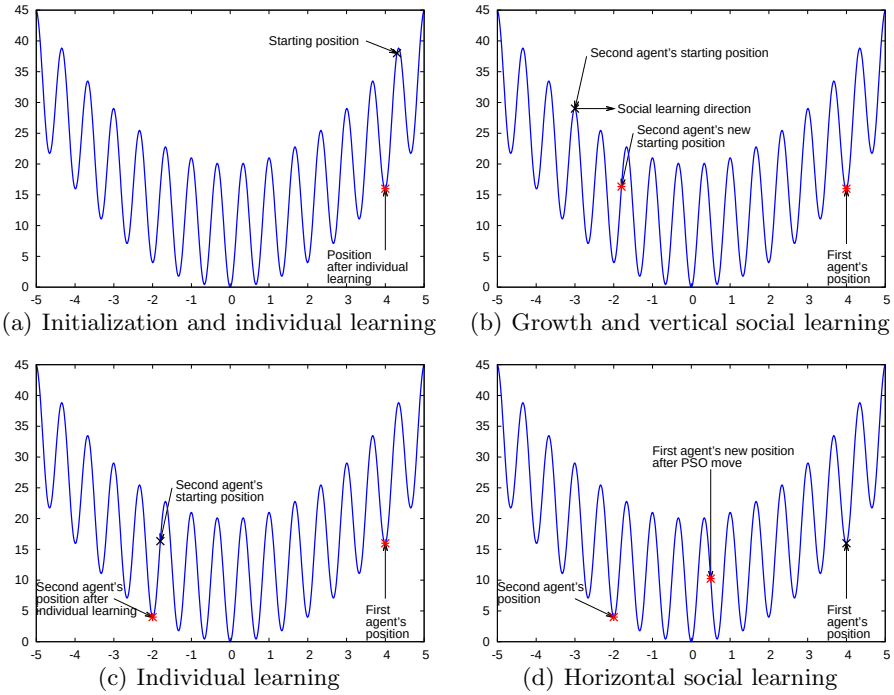
```
/* Initialization */
Create initial particle p_1 and add it to the set of particles P which is initially empty
Initialize position vector p_1.x to random values within the search range
Initialize velocity vector p_1.v to zero
Set p_1.pb = p_1.x

/* Main Loop */
Set t = 0 and k = 1
repeat
  /* Individual learning */
  for i = 1 to k do
    Improve p_i.pb through a local search procedure
  end for

  /* Horizontal social learning */
  for i = 1 to k do
    Move p_i.x using Eqs. 1 and 2
    if f(p_i.x) is better than f(p_i.pb) then
      Set p_i.pb = p_i.x
    end if
  end for

  /* Population growth and vertical social learning */
  if Agent addition criterion is met then
    Create particle p_{k+1} and add it to the set of particles P
    Initialize position vector p_{k+1}.x using Eq. 3
    Initialize velocity vector p_{k+1}.v to zero
    Set p_{k+1}.pb = p_{k+1}.x
    Set k = k + 1
  end if
  Set t = t + 1
  Set sol = argmin_{p_i ∈ P} f(p_i.pb)
until f(sol) is good enough or t = t_max
```

and separable, it should be optimized in a single local search run without the need of using the exploration capabilities provided by the particle swarm algorithm. Similarly, if the objective function has a few local optima, starting with only one particle ensures the minimal use of function evaluations for the exploration of the space. However, the objective function may also have features for which local search and small initial populations are ineffective. For example, if an objective function had large plateaus, the local search procedure would return with no solution improvement after having wasted several function evaluations. Under these circumstances, the algorithm would start making progress (thanks



(a) Initialization and individual learning (b) Growth and vertical social learning

(c) Individual learning (d) Horizontal social learning

Fig. 1. The incremental particle swarm-guided local search algorithm at work. In Figure (a), a particle is randomly placed in the search space. Immediately after, the first round of individual learning (local search) is run, moving the particle to, or close to, a local minimum. This is followed by a horizontal learning round (a normal PSO move) that does not have any effect in a swarm of one particle (the particles' velocity are initialized to zero). In Figure (b), the size of the population is increased. The vertical social learning rule (Eq. 3) is used to place the new particle. In Figure (c), the second iteration of the algorithm begins by running a round of individual learning. In Figure (d), through a round of horizontal learning, particles can move and discover more promising areas of the search space. The algorithm continues intertwining individual and social learning procedures until a stopping criterion is satisfied.

to the exploration capabilities of the particle swarm component) until a critical population size is reached.

4 Experimental Setup

The performance of the incremental particle swarm-guided local search algorithm (labeled IPSOLS) is compared to that of the following algorithms:

1. A traditional particle swarm optimization algorithm with constant population size (labeled PSO). Three population sizes of 10, 100 and 1000 particles are used.

2. An incremental particle swarm optimization algorithm as described in Section 3 (labeled IPSO). In [4], IPSO proved to work well with a fast agent addition schedule. Accordingly, we use a fast particle addition schedule in which a new particle is added every iteration (this schedule is also used in IPSOLS). The maximum size of the swarm is set to 1000 particles. Whenever a new particle is added, the particle that serves as model to imitate is the best of the swarm.
3. 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. Three population sizes of 10, 100 and 1000 particles are used.
4. 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.

All particle swarm-based algorithms (PSO, IPSO, PSOLS and IPSOLS) are 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. Other parameter settings are $\varphi_1 = \varphi_2 = 2.05$ and $\chi = 0.7298$.

Powell's method has also a number of parameters to be defined. The so-called tolerance, used to stop the procedure once a very small difference between two solutions is detected, is set to 0.01. In case such a difference is not found, the procedure is stopped after a maximum number of iterations. In our experiments, this parameter is set to 10. Other settings were explored but no significant differences in the results were found. Finally, different step sizes (used for the line minimization procedure) were explored. The values tried for this parameter were 0.1, 1, 10, 20, and 33% of the length of the corresponding search range.

A set of six benchmark functions are used in our experiments. Their mathematical formulation and search ranges are listed in Table 1. In all cases, we used their 100-dimensional instantiations (i.e., $n = 100$).

Our data is based on 100 independent runs of up to 10^6 function evaluations each. In each run, all benchmark functions (except Schwefel's) were randomly shifted within the specified range. Particles or trial points outside the search range were forced to stay within by putting them on the boundary.

5 Results

The distribution of the solution quality obtained by the compared algorithms after 10^6 function evaluations is shown in Figure 2. These results correspond to the case in which all particle swarm-based algorithms use a ring topology and the local search step size is equal to 20% the length of the search range. In [16], the reader can find the complete set of results, including those from the statistical

Table 1. Benchmark optimization problems

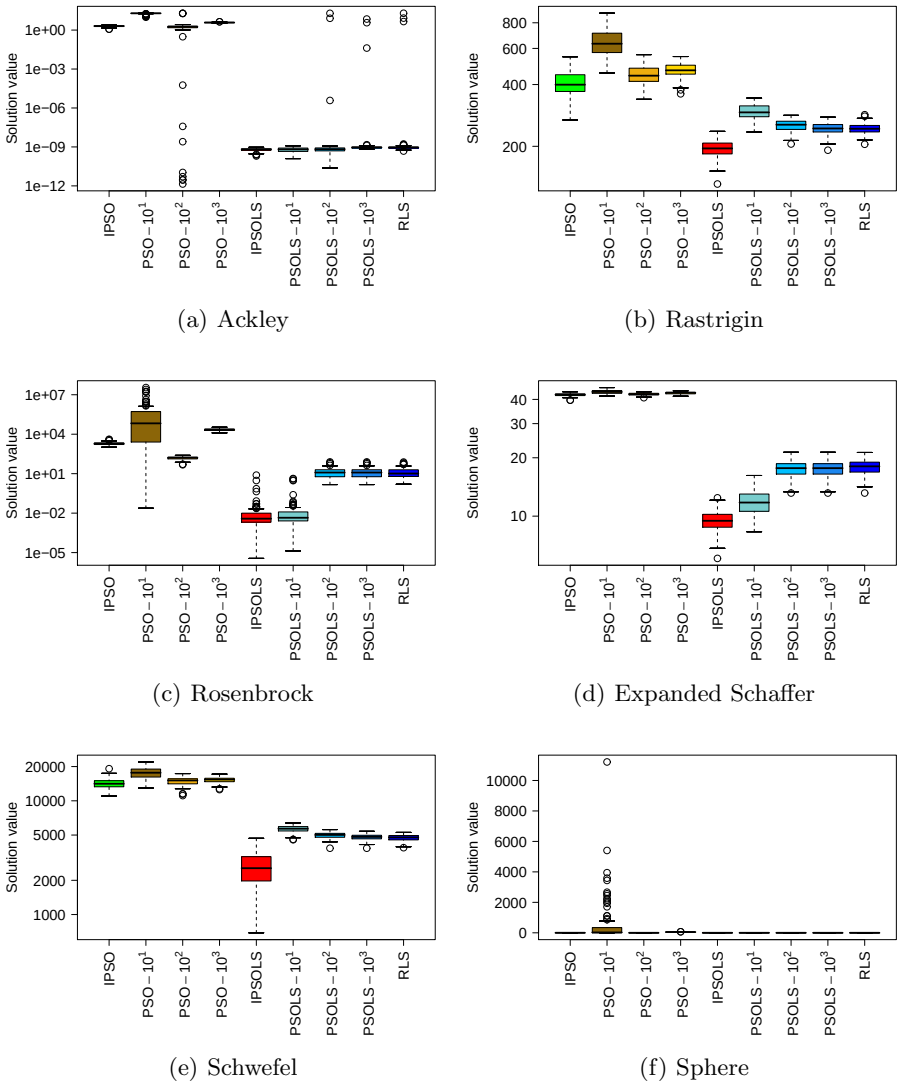
Name	Definition	Range
Ackley	$-20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e$	[-32,32]^n
Rastrigin	$10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$	[-5.12,5.12]^n
Rosenbrock	$\sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	[-30,30]^n
Expanded Schaffer	$ES(\boldsymbol{x}) = \sum_{i=1}^{n-1} S(x_i, x_{i+1}) + S(x_n, x_1)$, where $S(x,y) = 0.5 + \frac{\sin^2(\sqrt{x^2+y^2}) - 0.5}{(1+0.001(x^2+y^2))^2}$	[-100,100]^n
Schwefel	$418.9829n + \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	[-500,500]^n
Sphere	$\sum_{i=1}^{n} x_i^2$	[-100,100]^n

significance tests performed on the algorithms' output data[3]. The algorithms without a local search component are placed on the left side of each plot. This arrangement reveals that the algorithms with a local search component find solutions of equal or better quality than the algorithms without it. The use of a local search component also makes the algorithms more robust to changes in the population topology, that is, the algorithms with a local search component do not show a significant change in their performance if a different topology is used. Only PSO and IPSO are affected by a change in the population topology. For example, the PSO algorithm with 10 particles finds better solutions with the ring topology than with the fully connected topology while the opposite happens if larger populations are used. In any case, regardless of the topology used, the algorithms with local search outperform those without it.

IPSOLS (at the center of all plots) performs at least as well as the other particle swarm-based algorithms with local search. In three problems (Expanded Schaffer, Schwefel, and Rastrigin), IPSOLS obtains better solutions than all other algorithms. With Ackley's function, IPSOLS, PSOLS and RLS obtain comparable results. With Rosenbrock's function, IPSOLS and PSOLS with a population of 10 particles obtain the best results. It is important to note that even after 10^5 function evaluations, the only algorithm that consistently finds high-quality solutions to all our benchmark problems is IPSOLS.

Figure 3 shows the development over time of the median solution quality obtained by the compared algorithms. In order to make the plots more readable, we consider only the algorithms that obtained the best results after 10^6 function evaluations. These plots show more clearly the effects of the incremental and socially guided use of local search procedures. As expected, IPSOLS exhibits the same performance as RLS during the first 10^4 function evaluations. This happens because IPSOLS starts, just as RLS does, performing a local search from a random initial solution. Once the first local search procedure stops making progress, IPSOLS and RLS start differentiating. The beneficial effects of relying

[3] Unless otherwise noted, all our statements are supported by statistical evidence.



(a) Ackley (b) Rastrigin

(c) Rosenbrock (d) Expanded Schaffer

(e) Schwefel (f) Sphere

Fig. 2. Boxplots showing the distribution of the solution quality obtained by the compared algorithms after 10^6 function evaluations. These results correspond to the case in which all particle swarm-based algorithms used a ring topology. The numbers next to the labels indicate the population size used.

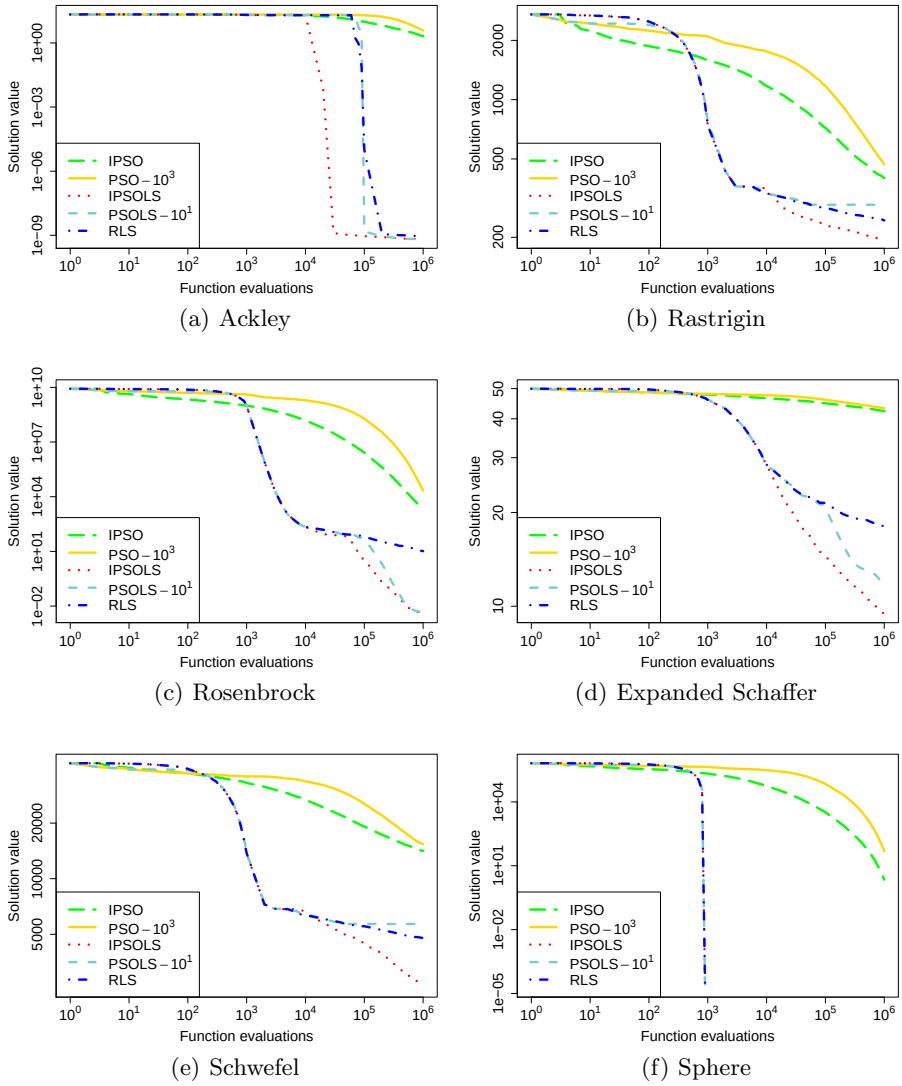


Fig. 3. Development of the solution quality over time (median values). These results correspond to the case in which all particle swarm-based algorithms used a ring topology. The numbers next to the labels indicate the population size used.

on socially acquired information are specially clear in the results obtained for Ackley, Rastrigin, Expanded Schaffer, and Schwefel's problems.

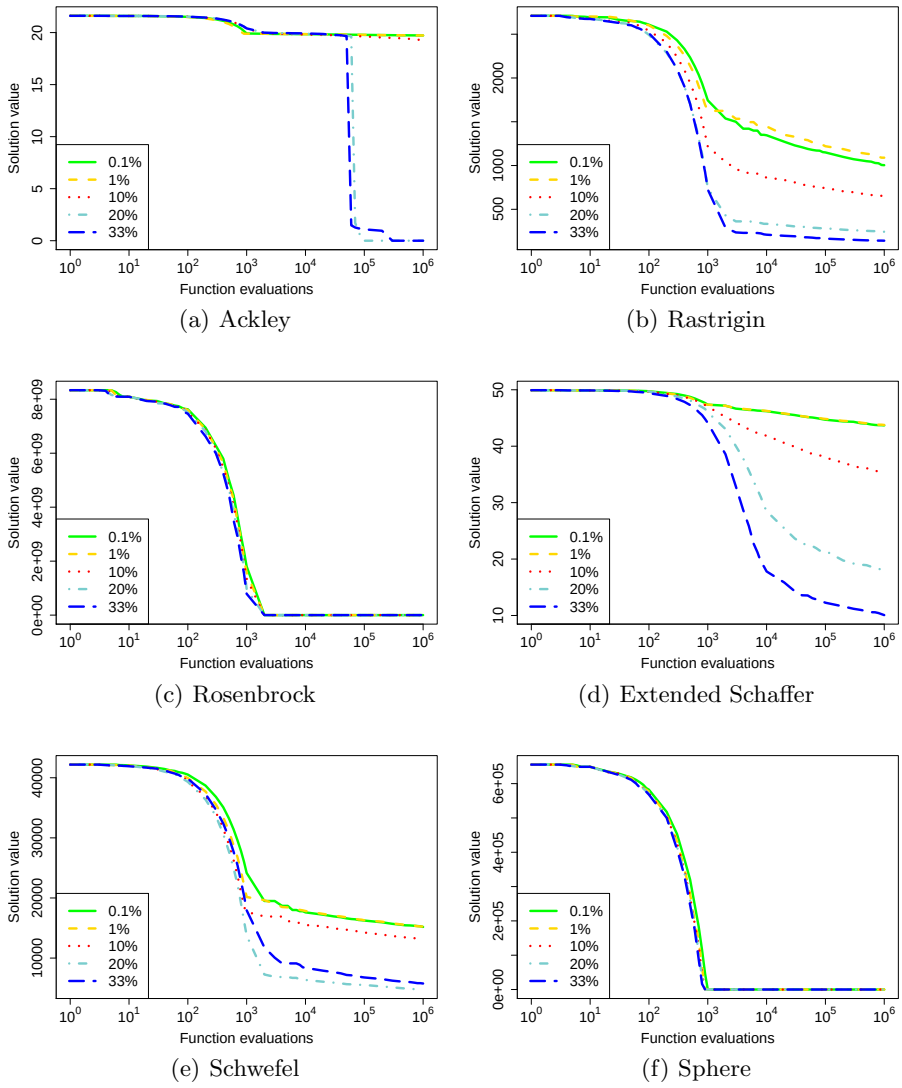
The results obtained while solving the Sphere problem show clearly that starting with one single particle is beneficial if the objective function is separable and convex. In this case, as can be seen in the plots, IPSOLS and RLS have exactly the same performance. This is so because a single run of the local search procedure is able to find the optimal solution with no extra algorithmic overhead. The difference between IPSOLS and RLS is that IPSOLS is able to adapt and cope with the difficulties posed by the features of other objective functions.

All the algorithms that contain Powell's method as an algorithmic component are affected by the step size of the subsidiary line minimization algorithm (in our case, Brent's algorithm). Figure 4 shows the median solution quality development over time for different step sizes using RLS. As can be seen in the plots, the actual effect of the step size depends on the objective function. In our study, the greatest differences are seen on multimodal test problems (Ackley, Rastrigin, Expanded Schaffer, and Schwefel) while there is practically no difference on unimodal functions (Rosenbrock and Sphere). This result suggests that, on multimodal problems, large step sizes allow the local search procedure to "jump" over local optima, which effectively increases the chance of finding good quality local optima. The downside of this behavior is that if the step size parameter is set to the wrong value, a poor performance can be obtained.

6 Related Work and Discussion

IPSOLS can be seen as a population of agents with the capability of learning by themselves and from others. In order to obtain the greatest reward, individuals must use a strategy that allows them to intertwine their two modes of learning in a productive way. In IPSOLS, individual learning is preferred over social learning. Agents learn socially only when individual learning is either too costly or deemed unproductive. Practically speaking, this means that local search is always tried first. If local search alone cannot solve the problem at hand satisfactorily, either because it has converged to a local optimum (i.e., individual learning is deemed unproductive because no further improvement can be made by local search) or because the maximum number of iterations of the local search procedure has been reached (i.e., individual learning is deemed too costly), then social learning is used. Of course, it is possible to intertwine the two modes of learning in a different way. A study on the utility of different learning strategies seems a fruitful avenue for future research.

IPSOLS shares a number of features with other algorithms. In particular, a time-varying population size and a subsidiary local search algorithm. Research on population (re)sizing has been a topic of study 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 [17]. The problem is that we usually know little about the real-world problem's difficulty. The approach taken in the design of the



(a) Ackley (b) Rastrigin

(c) Rosenbrock (d) Extended Schaffer

(e) Schwefel (f) Sphere

Fig. 4. Development of the solution quality over time (median values) for different local search step sizes using RLS.

incremental particle swarm-guided local search algorithm (i.e., to start with small populations) makes sense because if the problem is not so difficult, a growing population will offer the best solution quality vs. time trade-off. If the problem is difficult, there will be a time penalty to pay for reaching very high quality solutions; however, acceptable solutions may be found early in a run.

Practically all resizing strategies consider the possibility of reducing the size of the population during an algorithm's run (see e.g. [18–23]). An exception to this approach is the work of Auger and Hansen [24] 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 may be that the performance of IPSOLS is improved by decreasing the population size from time to time. This is because decreasing the population size may save many function evaluations. In terms of solution quality, however, it is not expected that this may impact negatively on the quality of the solutions found since the choice of removal would be done on converged particles (particles very close to each other with very small velocity and already in a local minimum).

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. [25] combined a particle swarm algorithm with a hill-climbing local search procedure; Gimmler et al [26] experiment with Nelder and Mead's simplex method as well as with Powell's method; Das et al. [27] 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. [28] report experiments with several local search-particle swarm combination schemes. All these previous approaches try to enhance the performance of the particle swarm algorithm; IPSOLS, on the contrary, guides a local search procedure in order to better explore the search space. By doing it incrementally, IPSOLS can exploit the features of the objective function.

7 Conclusions

A hybrid algorithm for solving continuous optimization problems, called incremental particle swarm-guided local search, has been presented. The algorithm has two main components: (i) a particle swarm optimization algorithm with growing population size and (ii) a subsidiary local search procedure that operates on the particles' previous best positions. We presented the experimental evidence that allows us to conclude that the algorithm exhibits a very good exploitation behavior that does not compromise its global search capabilities.

The design of the proposed algorithm is inspired by results in social learning and swarm intelligence research. Using a multiagent systems metaphor, the

algorithm can be thought of as a growing population of agents that can learn both socially and individually. The strategy used by the agents to intertwine these two modes of learning is the following: when a naive agent becomes part of the main population, it learns from more experienced agents. Once it is part of the main population, an agent learns socially only when learning individually is considered to be too costly or when doing it is deemed unproductive.

Acknowledgments Marco A. Montes de Oca is funded by the Programme Alβan, the European Union Programme of High Level Scholarships for Latin America, scholarship No. E05D054889MX, and by the *SWARMANOID* project funded by the Future and Emerging Technologies programme (IST-FET) of the European Commission (grant IST-022888). Thomas Stützle acknowledges support from the F.R.S-FNRS of the French Community of Belgium of which he is a Research Associate.

References

1. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading, MA, USA (1989)
2. Dorigo, M., Stützle, T.: Ant Colony Optimization. Bradford Books. MIT Press, Cambridge, MA, USA (2004)
3. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ, USA, IEEE Press (1995) 1942–1948
4. Montes de Oca, M.A., Stützle, T.: Towards incremental social learning in optimization and multiagent systems. In: Workshop on Evolutionary Computation and Multiagent Systems Simulation of the Genetic and Evolutionary Computation Conference (GECCO 2008), New York, NY, USA, ACM Press (2008) In press.
5. Powell, M.J.D.: An efficient method for finding the minimum of a function of several variables without calculating derivatives. The Computer Journal **7**(2) (1964) 155–162
6. Nehaniv, C.L., Dautenhahn, K., eds.: Imitation and Social Learning in Robots, Humans and Animals: Behavioral, Social and Communicative Dimensions. Cambridge University Press, Cambridge, United Kingdom (2007)
7. Curran, D., O'Riordan, C.: Increasing population diversity through cultural learning. Adaptive Behavior **14**(4) (2006) 315–338
8. Aoki, K., Wakano, Y., Feldman, M.W.: The emergence of social learning in a temporally changing environment: A theoretical model. Current Anthropology **46**(2) (2005) 334–340
9. Laland, K.N.: Social learning strategies. Learning & Behavior **32**(1) (2004) 4–14
10. Galef Jr., B.G., Laland, K.N.: Social learning in animals: Empirical studies and theoretical models. BioScience **55**(6) (2005) 489–499
11. Cavalli-Sforza, L.L., Feldman, M.W.: Cultural Transmission and Evolution. A Quantitative Approach. Princeton University Press, Princeton, NJ, USA (1981)
12. Giraldeau, L.A., Valone, T.J., Templeton, J.J.: Potential disadvantages of using socially acquired information. Philosophical Transactions of the Royal Society of London. Series B: Biological Sciences **357** (2002) 1559–1566
13. Clerc, M., Kennedy, J.: The particle swarm–explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation **6**(1) (2002) 58–73

14. Brent, R.P.: Algorithms for Minimization Without Derivatives. Prentice-Hall, Englewood Cliffs, NJ, USA (1973)
15. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. The Art of Scientific Computing. Second edn. Cambridge University Press (1992)
16. Montes de Oca, M.A., Van den Enden, K., Stützle, T.: Incremental particle swarm-guided local search for continuous optimization: Complete results (2008) Supplementary information page at `http://iridia.ulb.ac.be/supp/IridiaSupp2008-013/`.
17. Lobo, F.G., Lima, C.F.: Adaptive Population Sizing Schemes in Genetic Algorithms. In: Parameter Setting in Evolutionary Algorithms. Volume 54/2007 of Studies in Computational Intelligence. Springer, Berlin, Germany (2007) 185–204
18. Arabas, J., Michalewicz, Z., Mulawka, J.J.: GAVaPS – A genetic algorithm with varying population size. In: Proceedings of the IEEE Conference on Evolutionary Computation, Piscataway, NJ, USA, IEEE Press (1994) 73–78
19. Harik, G.R., Lobo, F.G.: A parameter-less genetic algorithm. In Banzhaf, W., et al., eds.: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), San Francisco, CA, USA, Morgan Kaufmann (1999) 258–265
20. Bäck, T., Eiben, A.E., van der Vaart, N.A.L.: An empirical study on GAs "without parameters". In: Parallel Problem Solving from Nature, PPSN VI, Berlin, Germany, Springer-Verlag (2000) 315–324
21. Eiben, A.E., Marchiori, E., Valkó, V.A.: Evolutionary algorithms with on-the-fly population size adjustment. In: Parallel Problem Solving from Nature, PPSN VIII, Berlin, Germany, Springer-Verlag (2004) 41–50
22. Fernandes, C., Rosa, A.: Self-regulated population size in evolutionary algorithms. In: Parallel Problem Solving from Nature, PPSN IX, Berlin, Germany, Springer-Verlag (2006) 920–929
23. Coelho, A.L.V., de Oliveira, D.G.: Dynamically tuning the population size in particle swarm optimization. In: Proceedings of the ACM Symposium on Applied Computing (SAC'08), New York, NY, USA, ACM Press (2008) 1782–1787
24. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), Piscataway, NJ, USA, IEEE Press (2005) 1769–1776
25. Chen, J., Qin, Z., Liu, Y., Lu, J.: Particle swarm optimization with local search. In: Proceedings of the International Conference on Neural Networks and Brain (ICNN&B 2005), Piscataway, NJ, USA, IEEE Press (2005) 481–484
26. Gimmler, J., Stützle, T., Exner, T.E.: Hybrid particle swarm optimization: An examination of the influence of iterative improvement algorithms on performance. In Dorigo, M., et al., eds.: LNCS 4150. Ant Colony Optimization and Swarm Intelligence. 5th International Workshop, ANTS 2006, Berlin, Germany, Springer-Verlag (2006) 436–443
27. Das, S., Koduru, P., Gui, M., Cochran, M., Wareing, A., Welch, S.M., Babin, B.R.: Adding local search to particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005), Piscataway, NJ, USA, IEEE Press (2006) 428–433
28. Petalas, Y.G., Parsopoulos, K.E., Vrahatis, M.N.: Memetic particle swarm optimization. Annals of Operations Research **156**(1) (2007) 99–127