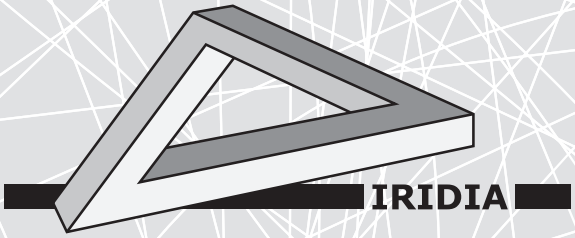


Université Libre de Bruxelles

Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Towards Incremental Social Learning in Optimization and Multiagent Systems

Marco A. Montes de Oca and Thomas Stützle

IRIDIA – Technical Report Series

Technical Report No.
TR/IRIDIA/2008-012

April 2008

To appear in the Proceedings of the Evolutionary Computation and Multi-Agent Systems and Simulation (ECoMASS) Workshop. GECCO 2008

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/2008-012

Revision history:

TR/IRIDIA/2008-012.001 April 2008

Towards Incremental Social Learning in Optimization and Multiagent Systems

Marco A. Montes de Oca mmontes@ulb.ac.be
Thomas Stützle stuetzle@ulb.ac.be
IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

April 2008

Abstract

Social learning is a mechanism that allows individuals to acquire knowledge from others without incurring the costs of acquiring it individually. Individuals that learn socially can thus spend their time and energy exploiting their knowledge or learning new things. In this paper, we adapt these ideas for their application to both optimization and multiagent learning. The approach consists of a growing population of agents that learn socially as they become part of the main population. We find that learning socially in an incremental way can speed up the optimization and learning processes, as well as improve the quality of the solutions and strategies found.

1 Introduction

The design of systems composed of numerous autonomous entities that exhibit, at a collective level, some desired behaviors remains an outstanding problem in various fields including multiagent systems and population-based stochastic optimization algorithms.

Broadly speaking, there are two alternative approaches to tackle this problem. The first one consists in using a traditional top-down approach whereby behaviors at the individual level are carefully crafted. The main issue with this approach is that its effectiveness depends ultimately on the designer's ingenuity and his/her application-specific experience. In some cases, it is possible to resort to the observation of some collective behavior of interest exhibited by a natural system in order to identify the individual behaviors that make it possible. Examples of systems that have been designed using this approach are ant colony optimization [6] and particle swarm optimization [12, 7]. The second approach consists in using automatic learning techniques in order to find an appropriate mapping from the agents' states to their actions that leads to the observation of the desired collective level behaviors; however, learning in a multiagent environment is usually a difficult task. The first difficulty comes from the exponential growth of the search space (defined as the space of the combined sensor-action mappings of all the participating agents) as a function of the number of agents [10]. To overcome this problem, agents can be equipped with their own learning algorithm, but in this situation, the main difficulties stem from

the interference caused by the co-existence of multiple learning agents whose rewards depend on the group's performance [18].

In this paper, we look at the effects of taking an incremental approach to the multiagent learning problem in which agents are added to the system one at a time. Starting with a small number of agents provides two advantages: (i) it enables fast learning for the initial population of agents due to the reduced interference effect that a large population provokes, and (ii) it may allow the optimal allocation of agents to solve a particular task. This incremental approach is complemented by the implementation of a social learning strategy whereby the newly added agents acquire knowledge about their environment from more experienced agents. The resulting mechanism, that we call incremental social learning, is aimed at facilitating scalability in the number of agents as well as at accelerating and improving learning.

This paper is organized in two parts. In the first part, we present a study on the effects of applying incremental social learning on a population-based optimization algorithm. The results are compared with those obtained with the standalone algorithm. In the second part, we present a similar study but this time using a multiagent system in which agents are capable of learning individually by means of embedded Q-learning algorithms. Different implementations of the incremental social learning framework are explored.

2 Incremental social learning

The term social learning is used to identify a class of mechanisms for knowledge transmission between agents without the use of genetic material [15, 5]. Social learning is attractive for the design of large multiagent systems because it allows individuals to acquire knowledge from other more experienced agents without incurring the costs of acquiring it individually [14]. However, theoretical models and empirical studies conclude that relying only on socially acquired knowledge is not always advantageous [9]. For social learning to be useful to a group, individuals must devote some of their time and energy to learn individually or to innovate [14].

The approach, that we call incremental social learning, consists of a growing population of agents which learn socially when they become part of the main group and learn individually when they are already part of it. The growing population strategy is based on the observation 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 [8]. The algorithmic structure of the incremental social learning framework is outlined below.

After initializing the environment and the initial population of agents (that we call primogenial), the main learning loop begins. If no agents are to be added, the agents in the current population learn individually. An agent addition schedule or criterion is used to control the rate at which agents are added to the main group. When a new agent is created and before it becomes part of the population, it learns socially from a subset of the already experienced agents. In Algorithm 1 above, the environment update state is made explicit in order to note the fact that the environment may be dynamic. In a real implementation, the environment can change at any time and not necessarily at the end of a

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
```

training round.

A small number of agents at the beginning of the learning process should reduce the interference caused by the co-existence of many learning agents. The agent addition schedule is used to create time delays that allow agents to learn individually from the interaction with the environment and with other agents. To make a parallel with natural systems, this time delays can correspond, for example, to the time that exists between the birth of an individual and the birth of its offspring. When a new agent is created, the agents that are part of the main population already acquired some updated knowledge of the world. Much of the effort spent by these agents in learning by themselves can be saved for the new agents by means of social learning. Incrementally growing the population size may also serve to allocate the minimum number of agents required to solve a particular problem.

The actual implementations of the individual and social learning mechanisms are independent from the incremental social learning framework outlined above. Both generic or application-specific mechanisms may be used. In the two case studies presented in the following sections, we explore different implementations of the incremental social learning framework.

3 Incremental social learning in optimization

In this section we evaluate the effectiveness of the incremental social learning strategy when applied to a population-based optimization algorithm. We first describe the optimization algorithm employed and the implementation details of the incremental social learning framework. Finally, we present the results of a series of experiments run on a number of benchmark optimization problems.

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
Schwefel	$418.9829n + \sum_{i=1}^{n} -x_i \sin(\sqrt{\lvert x_i \rvert})$	[-500,500]^n
Step	$6n + \sum_{i=1}^{n} \lfloor x_i \rfloor$	[-5.12,5.12]^n

3.1 Particle Swarm Optimization

The optimization algorithm we use is the particle swarm optimization (PSO) algorithm. It was inspired by the behavior of birds while flocking [12, 7]. In a PSO algorithm, a population of agents (called particles), whose positions in a multidimensional space represent potential solutions to an optimization problem, move by updating their velocity according to the information gathered by the group (called swarm). Every iteration, 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 defined 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

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

and

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

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 [4] 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.

Rather than learning individually, particles engage in some form of horizontal social learning whereby particles learn from the individuals in the main group. Indeed, the PSO algorithm can be seen as a simple model of social learning where agents try to imitate the "behaviors" of the individuals in their neighborhood that receive the greatest rewards [11, 13].

3.2 Experimental Setup

The particle addition criterion used in our experiments is solution quality stagnation. Whenever the solution improvement stagnates for a certain number of consecutive iterations k, a new particle is added. In our experiments, $k \in \{1, 5, 10\}$. Social learning is implemented using a simple rule that moves the new particle's previous best position from its initial random location in the search

space to a location 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 and is defined as

$$p'_{new,j} = p_{new,j} + U \cdot (p_{model,j} - p_{new,j}), \tag{1}$$

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 the range $[0, 1)$. Two strategies are used in order to select the model particle: (i) at random, or (ii) the best particle of the swarm. Note that the new particle does not copy the model particle but only moves closer to it. The new particle's velocity is randomly initialized.

The parameter settings for the PSO algorithm are the most commonly found in the literature, that is, the constriction factor χ is set to 0.729 and the acceleration coefficients φ_1 and φ_2 are both set to 2.05. Two population topologies are used: A fully connected one in which each particle is neighbor to all other particles in the swarm, and a ring topology, in which each particle is neighbor to two other particles.

The benchmark problems that are used to evaluate the effectiveness of using an incremental social learning strategy over a PSO algorithm are listed in Table 1. In all cases, their 100-dimensional form is used (i.e., $n = 100$). The location of the problems' global optimum is randomly shifted within the search range (except for Schwefel's function) in each of the 100 independent runs that were used.

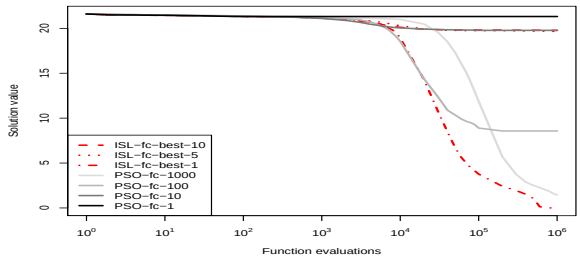
3.3 Results

The results obtained with the two strategies for selecting the model particle (i.e., one particle at random or the best one) do not show a significant difference in most of the studied cases[1]. This result may come from the fact that selecting the best particle or a random one from a group of particles that are close to each other in the search space is, in effect, the same.

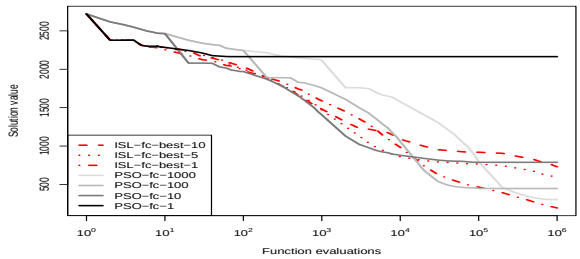
The relative difference between the results obtained with the fully connected and ring topologies is dependent on the problem being solved. The solution improvement during the first hundreds of function evaluations is usually faster with a fully connected topology, while the final best solutions are found with a ring topology. The factor that produces the most significant differences is the use of the incremental social learning approach. As an example, consider the results shown in Figure 1. These results correspond to the median solution quality development over time obtained with the constant population PSO algorithm and the incremental social learning PSO algorithm, both using a fully connected topology, on the five benchmark problems used in our experiments. In the case of the incremental social learning PSO algorithm, the best particle of the swarm is used as model.

The results show that with a constant population PSO algorithm there is usually a trade-off between solution quality and speed. The incremental social learning PSO algorithm does not have this problem as it seems to benefit from

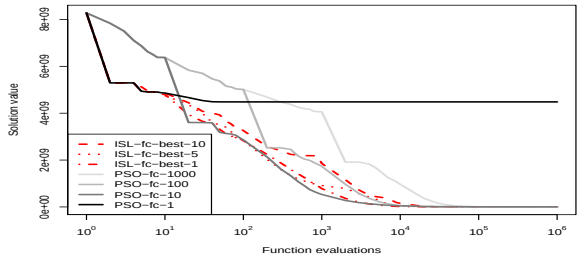
[1] Due to space constraints, we refer the interested reader to `http://iridia.ulb.ac.be/supp/IridiaSupp2008-009/` for access to all supporting data.



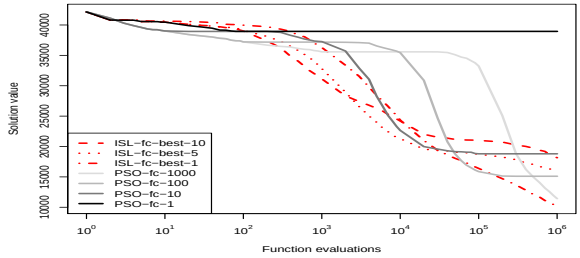
(a) Ackley



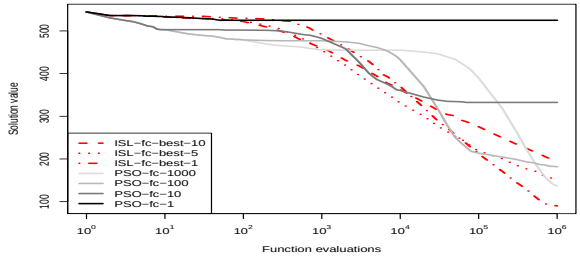
(b) Rastrigin



(c) Rosenbrock



(d) Schwefel



(e) Step

Figure 1: Median solution quality development over time. Different population sizes are used for the traditional PSO algorithm and different stagnation thresholds are used for the incremental social learning approach. In all cases a fully connected topology is used.

starting with a minimal population size. It finds solutions of the same, or even better, quality than a constant population PSO algorithm without the need of setting the population size in advance (although there is another parameter: the particle addition rate). There are cases (e.g., with Schwefel's and the Step functions) in which starting with one particle seems not to be the best strategy. However, the behavior of the algorithm in these cases is as if it waited for the right population size to then proceed with the optimization process. In spite of this "waiting time", the incremental social learning PSO algorithm obtained the best results at the end of the allocated number of function evaluations. The results also show that the role of the stagnation threshold which controls the rate at which particles are added to the main swarm is that of an exploration-exploitation control parameter. Faster rates encourage exploration while slower rates encourage exploitation. This is clearer when the results obtained with a ring topology (which encourages exploration) are compared with those obtained with a fully connected topology (which encourages exploitation). With a fully connected topology, the fastest particle addition rate ($k = 1$) produced the best results while with a ring topology, the slowest rates ($k = \{10, 5\}$) did.

4 Incremental social learning in multiagent systems

In this section we present the effects of using the incremental social learning framework on a multiagent system composed of learning agents. We describe the algorithms used, the experimental setup and the results obtained.

4.1 Individual Learners in a Multiagent System

By letting agents learn by themselves, the computational complexity of the multiagent learning problem can be reduced [2]. In this paper we consider a multiagent system composed of individually learning agents each of which uses a Q-learning algorithm for that purpose.

Q-learning allows an agent i to learn an action-value function $Q_i(a, s)$ that represents the value of taking a particular action a in a particular state s. The goal is to find a policy that maximizes the rewards received for executing a series of actions starting from an initial state. The so-called Q-values are updated using the rule

$$Q_i(a, s) = Q_i(a, s) + \alpha \cdot (r_i(s) + \gamma \cdot \max_{a'} Q_i(a', s') - Q_i(a, s)),$$

which is applied whenever action a is taken in state s leading to state s'. $r_i(s)$ is the reward received by the agent in state s, α is parameter known as the learning rate and γ is another parameter known as the discount factor.

The effectiveness of this approach depends on the constraints imposed by the problem being tackled. If agents can act more or less independently from others, this method can provide good results in a reasonable amount of time [1].

4.2 Experimental Setup

We use the multiagent grid world problem as described by Agogino and Tumer [1]. In this problem, a square grid populated by agents that can move, one patch

at a time, in any of four directions (up, down, left and right). An agent's state is its location on the grid. Each grid patch can contain a token with a value within the range $[0, 1]$ that represents the reward given to the agent that first moves to that patch. If no token is present, the reward is zero. The agents' aim is to maximize the total reward received after a fixed number of time steps. In our experiments, 100 agents are used. The whole learning process lasts for 1000 rounds. In each one of these rounds agents start from an initial state, move for 50 time steps, and then are moved back to their initial state.

Two configurations of a 20×20 environment are used. In the first one, only one fourth of the world is populated with tokens with values that decrease inversely with the distance to the center of the world. In this configuration, the token-value landscape resembles a cliff. In the second configuration tokens are clustered. The location of the clusters is selected uniformly at random. In our experiments, 33 clusters are used. Their width and the values associated with their tokens are set according to a Gaussian function with standard deviation $\sigma = 1$. In both configurations, all token values are normalized so that the maximum collective reward is equal to one.

In these experiments, agents are added according to a predefined schedule. We experiment with three schedules, adding agents every 1, 5, and 10 rounds until the maximum number of agents is reached. Social learning is implemented using a subset of the system's current population of agents. Three different agents are selected at random to update the new agent's Q-values using the rule

$$\begin{aligned} Q'_{new}(a,s) = & \; Q_{new}(a,s) \\ & + \; U_1 \cdot (Q_{model_1}(a,s) - Q_{new}(a,s)) \\ & + \; U_2 \cdot (Q_{model_2}(a,s) - Q_{model_3}(a,s)), \end{aligned} \tag{2}$$

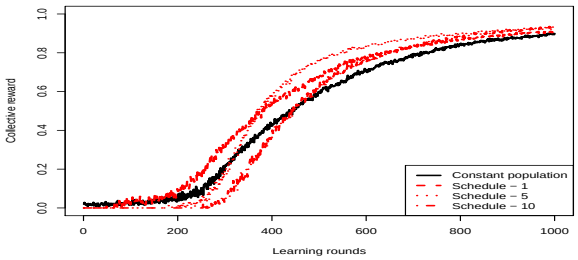
where $Q'_{new}(a,s)$ is the new agent's new Q-value, $Q_{new}(a,s)$ is the new agent's original Q-value, and $Q_{model_{1,2,3}}(a,s)$ are the models' Q-values. The random numbers U_1 and U_2 are drawn from a uniform distribution in the range $[0, 1)$. In order to use this rule, at least three agents must compose the primogenial population. The rule is inspired by the update rule used in differential evolution [19].

In all the experiments, the learning rate α is set to 0.5 and the discount factor γ is set to 0.9. The action selection is controlled by an ϵ-greedy strategy with ϵ set to 0.2. The results are based on 100 independent runs.

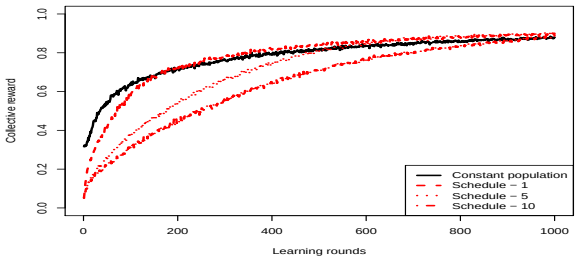
4.3 Results

The median of the collective reward (i.e., the sum of the agents' accumulated rewards) distribution over time is shown in Figure 2.

In the cliff environment, several learning rounds are needed before a sustained improvement of the collective reward is observed. This contrasts with the results obtained in the clustered environment, in which even random movements make the system obtain a collective reward greater than zero. This result is expected given the greater difficulty of the cliff environment in which the highest rewards are located at the center of the environment. Since rewards are given only to the agent that first visits a given patch, there is some competition among the agents to arrive first to the patches where the highest-valued tokens are located. The effects of the incremental social learning approach are



(a) Cliff environment



(b) Clustered environment

Figure 2: Median of the collective reward distribution over time. Different agent addition schedules are used for the implementation of the incremental social learning framework.

more evident in the results obtained in the cliff environment. The improvement of the collective reward is faster if an incremental social learning approach is used. This suggests that the competition for rewards among 100 agents plays an important role in this problem. The incremental approach seems to benefit from the faster learning that happens when no major interference among agents occurs, which is the case at the beginning when just a few agents populate the environment. The social learning rule allows agents to avoid conflict and to use their time in exploiting the acquired knowledge and to contribute to the collective reward. Different agent addition schedules influence differently the results obtained. With the fastest schedule, the improvement is faster but the final reward is slightly lower that the one obtained with slower schedules.

In the clustered environment, the benefits of using an incremental social learning approach are less evident. In fact, the approach delays the improvement of the collective reward. This might be the result of the reduced competition among agents as a result of the token distribution in this environment.

5 Related work and discussion

In Section 1 we mentioned that one of the main problems that arises when applying learning techniques in a multiagent scenario is that there is some interference due to the co-existence of agents that are learning at the same time. From an agent's perspective, the problem is that the appropriate behavior is difficult to learn because the consequences of its actions depend on what other

agents are doing, which in turn depends on what others are doing and so on. A further complication comes from the fact that, while learning, agents change their behavior which may render other agents' learned behaviors obsolete [18]. A common way to deal with this problem is to reduce the number of learning agents in the system. For example, Guestrin et al. [10] use coordination graphs in order to reduce the complexity of the multiagent coordination problem. The idea comes from the observation that not all agents' actions must be tightly coupled in order to solve a problem. The incremental social learning framework tackles this problem by actually reducing the number of agents in the system; however, the size of the primogenial population should be aligned with the complexity of the problem at hand.

The idea of letting the size of the population grow over time has been applied by Noble and Franks [16], who already pointed out the fact that newborn animals may benefit from the observation of elder individuals and implemented a simulation where the population of agents grows over time. In their work, they study the effects of using different social learning mechanisms and not on the utility of learning socially in an incremental way as we do.

A possible alternative to the individual learners approach is to try to control the learning process from a global level through a performance measure, common to all agents. Obtaining results with this approach, though possible in theory, is very difficult in practice because small changes in the agents' behavior can lead to very different collective level behaviors [3]. In fact, designing performance measures that allow agents to learn appropriate behaviors is a field of study on its own [17].

We believe that the independence of the incremental social learning framework from the actual social and individual learning algorithms makes it attractive for its application to a wide range of situations. However, the problem must have certain features if the incremental social learning approach is to provide any advantage over other approaches. One such feature is that it should be possible for an individual to know how well it is performing at any given time. This is very important because it is the basis for individual learning. Moreover, the agents must be capable of communicating relatively complex messages, a capability that is not always available, for example, in small robots.

6 Conclusions and Future Work

Social learning allows agents to acquire knowledge from others without incurring the costs of acquiring it individually. For naive individuals, it is particularly useful as it effectively provides them a shortcut to knowledge that otherwise may take them a long time to acquire.

In this paper we have defined an incremental social learning framework that consists of a growing population of agents that learn socially as they become part of the main population and learn individually once they are already part of it. The framework is independent of the underlying social and individual learning algorithms. By increasing gradually the size of the population, it is possible to accelerate the first learning phase in domains where inter-agent conflicts exist. Furthermore, it enables the possibility of allocating the minimum number of agents required to solve a particular problem.

To evaluate the framework's potential, we applied it on a population-based

optimization algorithm as well as on a multiagent learning system. The results show that by using an incremental social learning approach we can accelerate the optimization/learning process as well as improve the quality of the solutions found.

Future work should be focused on determining the best suited implementation of the framework for particular learning/optimization algorithms and problems. Another line of research is the application of the framework in online learning scenarios.

7 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] A. Agogino and K. Tumer. Reinforcement learning in large multi-agent systems. In *AAMAS'05 Workshop on Coordination of Large Scale Multi-agent Systems*. ACM Press, 2005.

[2] A. Agogino and K. Tumer. QUICR-learning for multi-agent coordination. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.

[3] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ, USA, 2001.

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

[5] D. Curran and C. O'Riordan. Increasing population diversity through cultural learning. *Adaptive Behavior*, 14(4):315–338, 2006.

[6] M. Dorigo and T. Stützle. *Ant Colony Optimization*. Bradford Books. MIT Press, Cambridge, MA, USA, 2004.

[7] R. Eberhart and J. Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, pages 39–43, Piscataway, NJ, USA, 1995. IEEE Press.

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

[9] L.-A. Giraldeau, T. J. Valone, and J. J. Templeton. Potential disadvantages of using socially acquired information. *Philosophical Transactions of*

the Royal Society of London. Series B: Biological Sciences, 357:1559–1566, 2002.

[10] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14. Proceedings of the 2001 Neural Information Processing Systems (NIPS) Conference*, pages 1523–1530, Cambridge, MA, USA, 2001. MIT Press.

[11] J. Kennedy. The particle swarm: Social adaptation of knowledge. In *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, pages 303–308, Piscataway, NJ, USA, 1997. IEEE Press.

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

[13] J. Kennedy, R. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann, San Francisco, CA, USA, 2001.

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

[15] C. L. Nehaniv and K. Dautenhahn, editors. *Imitation and Social Learning in Robots, Humans and Animals: Behavioral, Social and Communicative Dimensions*. Cambridge University Press, Cambridge, United Kingdom, 2007.

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

[17] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Bradford Books. MIT Press, Cambridge, MA, USA, 2000.

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

[19] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*. Natural Computing Series. Springer-Verlag, Berlin, Germany, 2005.